INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO

Campus São João da Boa Vista

Trabalho Final de Curso

4º ano – Curso Técnico em Informática

Prof. Breno Lisi Romano e Prof. Luiz Angelo Valota Francisco

A IMPORTÂNCIA DO MVC NO PROJETO GERAÇÕES

Aluno: Matheus Gregio Herrera

Prontuário: bv 1620291

São João da Boa Vista – SP

2019

Resumo

O intuito deste relatório não é ensinar o uso das linguagens de programação ou o uso da arquitetura MVC, mas sim apresentar conceitos básicos da programação na qual nos interessa é ressaltar o porque o MVC é uma ferramenta indispensável na hora da elaboracao e execucao de um projeto. E claro mantendo sempre um entendimento básico do assunto para qual não se disperse as informações.

Sumário

Sumário Figuras

1 Introdução

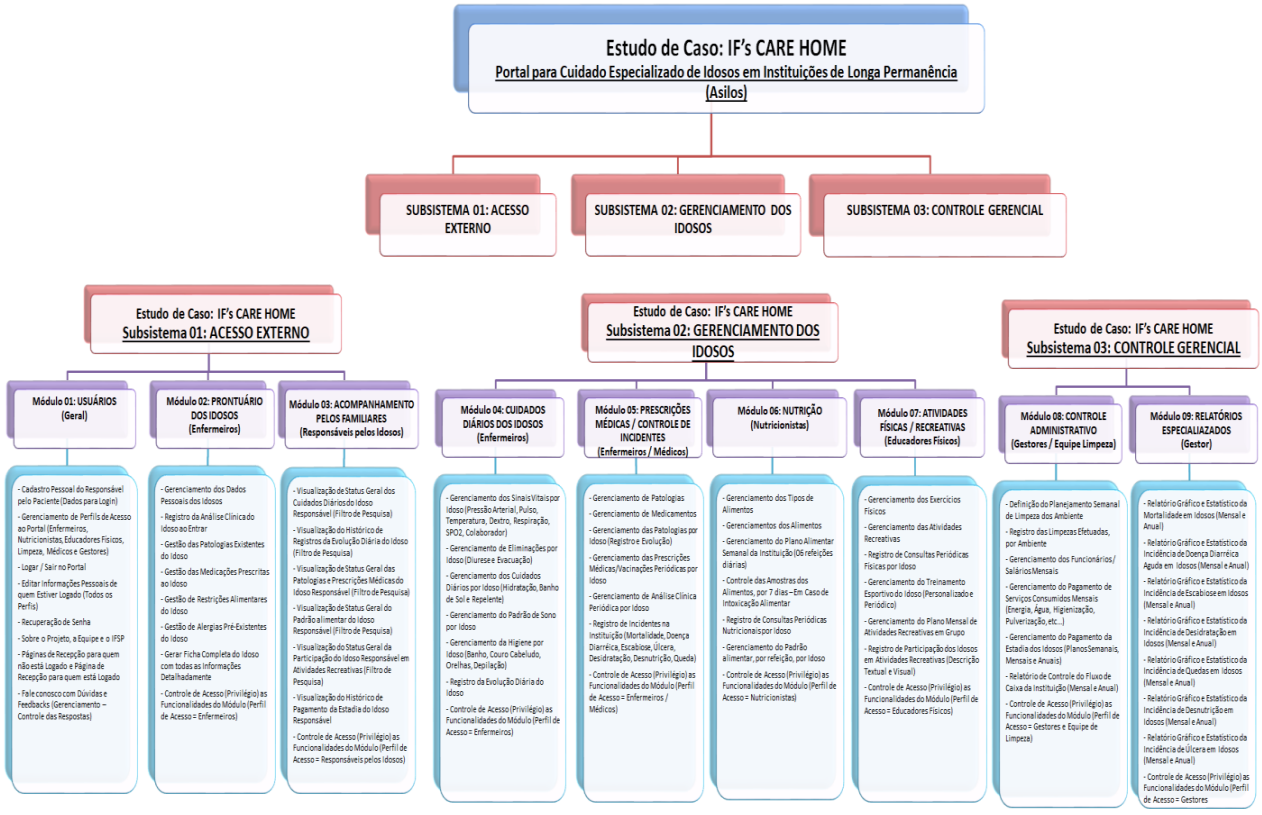
1.1 Contextualização / Motivação

São João da Boa Vista é um município situado no estado de São Paulo que, conta com aproximadamente 90.000 habitantes. O município também conta com um Instituto Federal de Ciência e Tecnologia, uma instituição federal que oferece ensino profissionalizante totalmente gratuita. Entre os diversos cursos oferecidos, os que são juntos ao ensino médio são, Técnico em Eletrônica e Informática, os dois com duração de quatro anos.[1][2][3]

O Curso técnico em Informática visa qualificar seu aluno de modo que estejam preparados para o mercado de trabalho em qualquer área da Informática. Neste curso, mais especificamente no quarto ano do ensino médio, está a matéria de Prática e Desenvolvimento de Sistemas (PDS) que e ministrada pelo professor Breno Lisi Romano, e no ano de 2019 conta com a participação do professor Luiz Angelo Valota Francisco. A matéria em questão visa colocar em prática todas as habilidades de programação dos alunos em um projeto final, que neste ano está sendo criado para uma instituição de longa permanência, construindo uma plataforma online onde não apenas os funcionários do local terão acesso, mas também a família do idoso. Podendo acessar diversas informações, desde a ficha do idoso, rotinas, rotinas de limpezas, atividade, alimentação e etc. Porém, cada usuário tem acesso restrito, para manter a segurança e organização da plataforma e de seus utilizadores.[4]

Este projeto conta com nove módulos diferentes, cada um responsável por uma parte do projeto. Entre estes grupos estão divididos em Programadores, Analistas e Dbas, cada módulo contanto com, pelo menos, dois alunos de cada função.

Figura 1 Divisão dos módulos do projeto Gerações.[4]



1.2 Objetivo Geral da Pesquisa

Considerando o projeto e sua divisão de equipes de trabalho em módulos, é preciso ter uma padronização para a codificação do projeto para que haja conformidade no código e evite bagunças e eventuais bugs. Perante a esta padronização do trabalho de desenvolvimento que a utilização da arquitetura Model-View-Controller (MVC).

Portanto o objetivo geral e entender a importância do MVC, para que haja conformidade e entendimento entre os códigos, páginas e desenvolvedores.

1.3 Objetivos Específicos

A partir do Objetivo Geral citado na seção anterior, destacam-se como objetivos específicos deste trabalho:

- Introduzir o projeto Gerações dos alunos do Instituto Federal de São João da Boa Vista.
- Definir o Uso do MVC no Projeto Gerações
- Ilustrar o MVC no Projeto Gerações
- Detalhar o Model

- Detalhar o View
- Detalhar o Controller
- Protótipo Final.

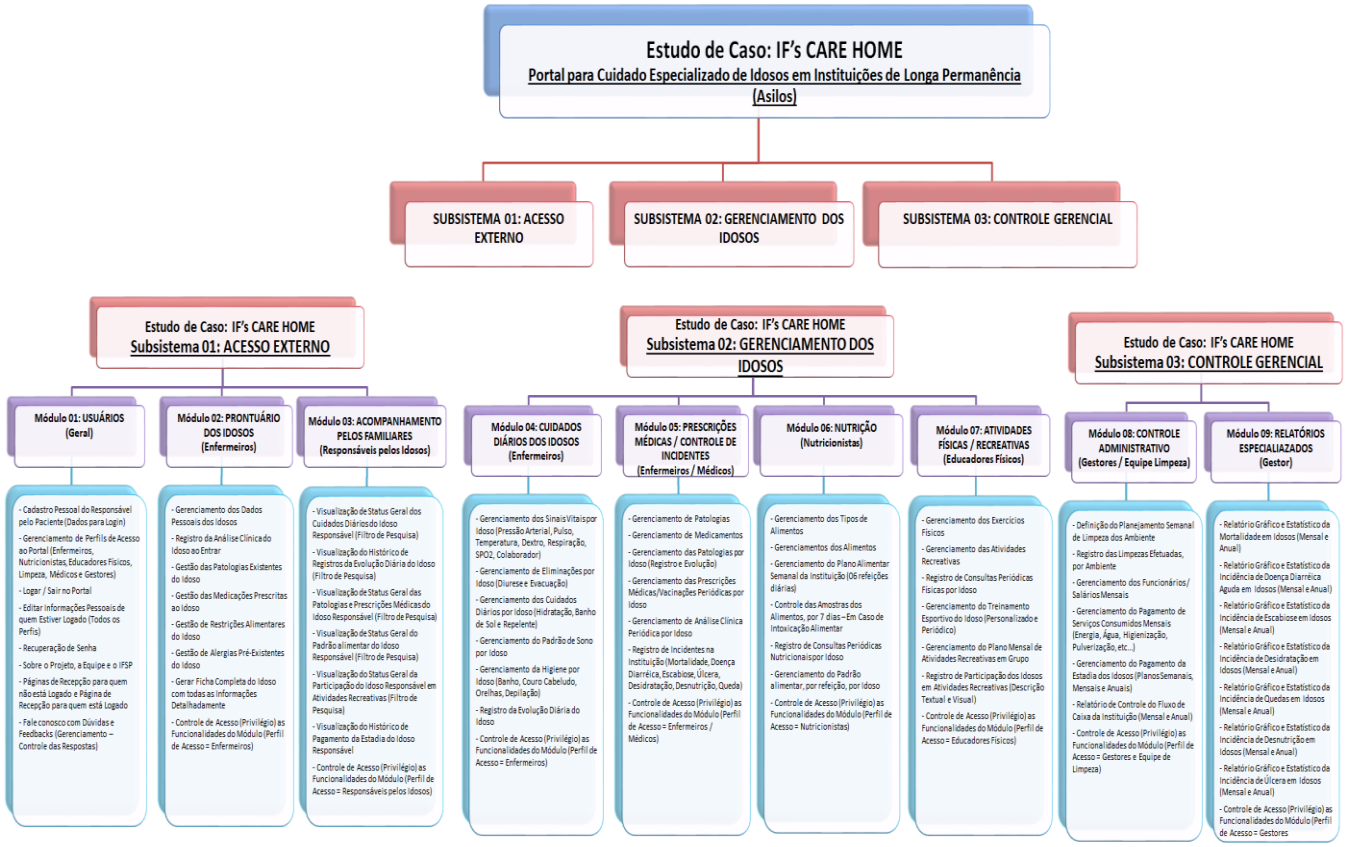
2 Desenvolvimento

2.1 Levantamento Bibliográfico

2.2.1 Apresentar o Projeto Gerações

O projeto Gerações, está sendo desenvolvido por alunos do quarto ano do ensino médio do Instituto federal de ciência e tecnologia no ano de 2019, ministrado pelo professor Breno Lisi Romano na Prática e Desenvolvimento de Sistemas (PDS). Atualmente conta-se com 60 integrantes. O objetivo do projeto é automatizar e facilitar a organização de uma casa de idosos, dando a eles um sistema onde poderão fazer rotinas, cardápios, cadastrar idosos, cadastrar medicamentos de cada idoso e muito mais. O projeto está separado em nove módulos contendo em cada um 2 Analistas, 2 Desenvolvedores, 2 Dbas.

Figura 2 Divisão dos módulos do projeto Gerações.[4]



2.2.2 Definir o Uso do MVC no Projeto Gerações

Para entendermos o uso do MVC no projeto, é necessário ter-se uma noção básica de alguns conceitos e também mais especificamente programação em PHP.

- **Linguagem de programação**

  A linguagem de programação nada mais é que um conjunto de comandos, identificadores, caracteres ASCII e etc, também contendo suas regras de sintaxe que permitem a construção lógica de ações que serão executadas pelo computador.[4]

Figura 3 linguagem de programação. [5]

```
LINGUAGEM DE PROGRAMAÇÃO = SÍMBOLOS + REGRAS DE SINTAXE
```

Compõem-se de dois componentes, Sintaxe e Semântica, a Sintaxe sendo o conjunto de regras que define letras dígitos e símbolos. Já a Semântica expressa o significado do programa. [4]

Na programação existem várias línguas diferentes como: Java, C, C++, PHP, JavaScript, Python, Swift e etc.

- **Variáveis e Operadores**

Variáveis estão presentes em todos tipos de programas e são fundamentais para que os mesmos funcionem. São declarados e inicializados antes da execução do programa. A seguir a sintaxe da variável.[6]

Figura 4 Exemplo de pseudocódigo.[6]

Sintaxe: <tipo_da_variável> <nome_da_variável>;
Exemplos: *double* salario_liquido, salario_bruto;

Devemos nos atentar também que a variável possui o seu "tipo" e dependendo do seu "tipo" define o que ela poderá armazenar, como por exemplo, uma variável de tipo char, só poderá armazenar caracteres.

Figura 5 Tipos de variáveis.[6]

Tipo primitivo	Valores a serem armazenados	Tamanho em *bits*
char	Permite armazenar um caractere. Exemplo: 'a'.	16
byte	Permite armazenar um número inteiro de -128 até +127. Exemplo: 10.	8
short	Permite armazenar um número inteiro de -32.768 até +32.767. Exemplo: 10.	16
int	Permite armazenar um número inteiro de -2.147.483.648 até +2.147.483.647. Exemplo: 10.	32
long	Permite armazenar um número inteiro de -9.223.372.036.854.775.808 até +9.223.372.036.854.775.807. Exemplo: 10.	64
float	Permite armazenar um **ponto flutuante** de -3,40292347E+38 até +3,40292347E+38. Exemplo: 3.1416.	32
double	Permite armazenar um ponto flutuante de -1,79769313486231570E+308 até +1,79769313486231570E+308. Exemplo: 3.1416.	64
boolean	Permite armazenar apenas o valor *true* ou o valor *false*.	8

Já os operadores são divididos em três categorias, aritméticos, relacionais e lógicos. A diferença vem na forma em qual se utiliza os operadores. São também usados em todos os programas e os mesmos são utilizados de diversas formas diferentes para fazer diversas coisas, mas tudo depende da língua na qual está utilizando, pois dependendo da mesma há vários jeitos de usar os operadores de modos diferentes.[6]

Tabela 1. Operadores aritméticos. [Desenvolvida pelo autor]

Operador	Descrição
+	Soma
-	Subtração

*	Multiplicação
/	Divisão
--	Decremento Unário
++	Incremento Unário
%	Resto de Divisão

Tabela 2. Operadores Relacionais. [Desenvolvida pelo autor]

Operadores	Descrição
==	Igualdade
>	Maior
<	Menor
>=	Maior ou Igual
<=	Menor ou Igual
!=	Diferente de

Tabela 3. Operadores Lógicos. [Desenvolvida pelo autor]

Operador	Descrição
&&	E
||	Ou
!	Não

- **Algoritmo**

Algoritmo e um conjunto finito de regras e passos que um determinado programa poderá executar após sua inicialização.[7][8]

Podemos ilustrar um o Funcionamento de um algoritmo através de Fluxograma utilizando de base o pseudocódigo a seguir:[7][8]

Figura 6 Exemplo de pseudocódigo.[8]

```
ALGORITMO
  DECLARE nota1, nota2, M : NUMÉRICO
  LEIA nota1
  LEIA nota2
  M ← (nota1 + nota2) / 2
  SE M >= 7.0 ENTÃO
    ESCREVA "Aprovado"
  SENÃO
    ESCREVA "Reprovado"
  FIM-SE
FIM_ALGORITMO.
```

Neste exemplo estamos fazendo um "programa" na qual pega duas notas e sem seguida realiza a média de ambas para saber se o aluno foi aprovado ou reprovado. Logo em seguida vemos a representação do "programa" em um fluxograma.

Figura 7 Legenda do Fluxograma.[8]

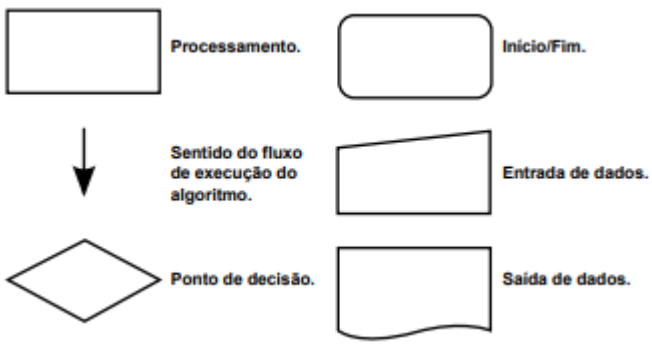
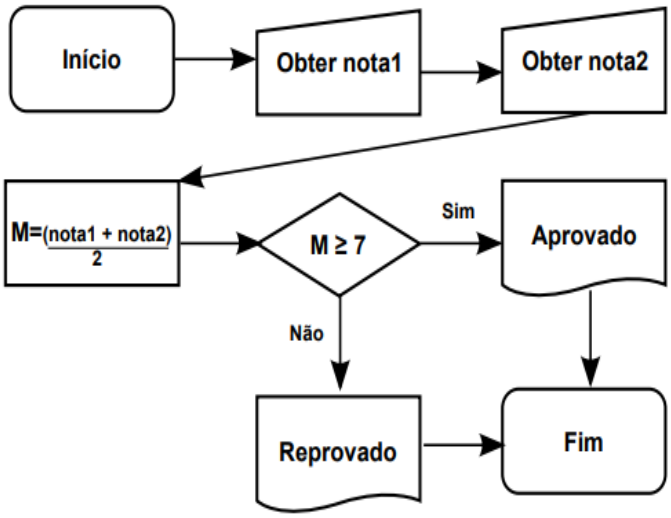


Figura 8 Fluxograma calculando a média por duas notas inseridas.[8]



- **Programação Orientada a Objetos**

  A programação orientada a objetos nada mais é que um programa que utiliza vários objetos, com características diferentes que realizam funções durante a execução do programa. Quando você está trabalhando com este tipo de programação você não propriamente criar esses objetos, mas sim suas classes que irão criar tais objetos, a classe em si agira como um modelo.[6]

Figura 9 Exemplo de classe no Projeto Gerações.

```php
namespace App\Model;

class Cadastro_tarefas {

    private $id;
    private $nome;
    private $data;
    private $setor;
    private $funcionario;
    private $descricao;
    private $observacao;

    public function __get($atributo) {
        return $this->$atributo;
    }

    public function __set($atributo, $valor) {
        $this->$atributo = $valor;
    }

}
```

Neste exemplo vemos que os atributos da classe estão com Encapsulamento na frente, que nada mais é que uma técnica para restringir o uso de métodos de outra classe. No caso, como os atributos desta classe estão com private, significa que apenas a classe tem acesso a essas informações, porém logo abaixo vemos dois métodos, que são basicamente funções que serão executadas pelo sistema, estão como público, isso significa que qualquer classe teria acesso a esses métodos. Ainda nesse mesmo tópico a opção Protected, que dá acesso a classe em si, mas todas as suas classes conectadas pela hierarquia (Herança) que será explicado mais a frente neste tópico.[6]

- **Herança**

Uma das vantagens de línguas que utilizam a orientação a objetos e a reutilização de códigos, mas isso não quer dizer um simples copy paste, e sim utilizando o recurso de Herança.

A partir desta ferramenta podemos criar uma classe a partir de outra já existente, e como a classe tivesse um filho, o "filho" herda os atributos e métodos da classe "pai". Para utilizar este recurso utilizamos a palavra ***extends*** na criação da classe [6][7][8]

Figura 10 Exemplo de classe com Herança, classe "Filho".

```php
class Cadastro_tarefasController extends Action{

    public function cadastrar(){

        if(isset($_POST['nometarefa'])&& isset($_POST['setor'])&& isset($_POST['funcionario'])&& isset($_POST['descricao'])&& i

            $cadastrotarefas = new Cadastro_tarefas();

            $cadastrotarefas->__set('nome', $_POST['nometarefa']);
            $cadastrotarefas->__set('setor', $_POST['setor']);
            $cadastrotarefas->__set('funcionario', $_POST['funcionario']);
            $cadastrotarefas->__set('descricao', $_POST['descricao']);
            $cadastrotarefas->__set('observacao', $_POST['observacao']);

            $cadastrotarefaDAO = new Cadastro_tarefasDAO();
            $cadastrotarefaDAO->inserir($cadastrotarefas);
        }

        header('Location: /md8/lista_tarefas');
        die();
    }
```

Figura 11 Exemplo de classe com Herança, "Pai"

```php
abstract class Action {

    private $view;

    function __construct() {
        $this->view = new \stdClass();
    }

    protected function getView() {
        return $this->view;
    }

    protected function render($view, $layout, $include=''){
        $this->view->page = $view;
        $this->view->include = $include;
        if(file_exists("App/View/$layout.php")){
            require_once "App/View/$layout.php";
        } else {
            $this->content();
        }
    }

    protected function content(){
        $classeAtual = get_class($this);
        $classeAtual = str_replace('App\\Controller\\', '', $classeAtual);
        $classeAtual = strtolower(str_replace('Controller', '', $classeAtual));

        require_once "App/View/$classeAtual/".$this->view->page.".php";
    }
```

Percebemos nesse exemplo que a classe "Filha" que tá recebendo o ***extends***, no caso a classe "Cadastrar" está herdando os métodos de sua classe "mãe" que no caso e o "Action".

- **Programação em PHP**

PHP (Hyper Text Processor), é uma linguagem de script open source para desenvolvimento web, porém é também usada para programação geral por ser uma língua vasta com propósitos diversos. Diferente de outras linguagens script, o PHP é uma língua server side, ou seja, precisamos de um.[10][11]

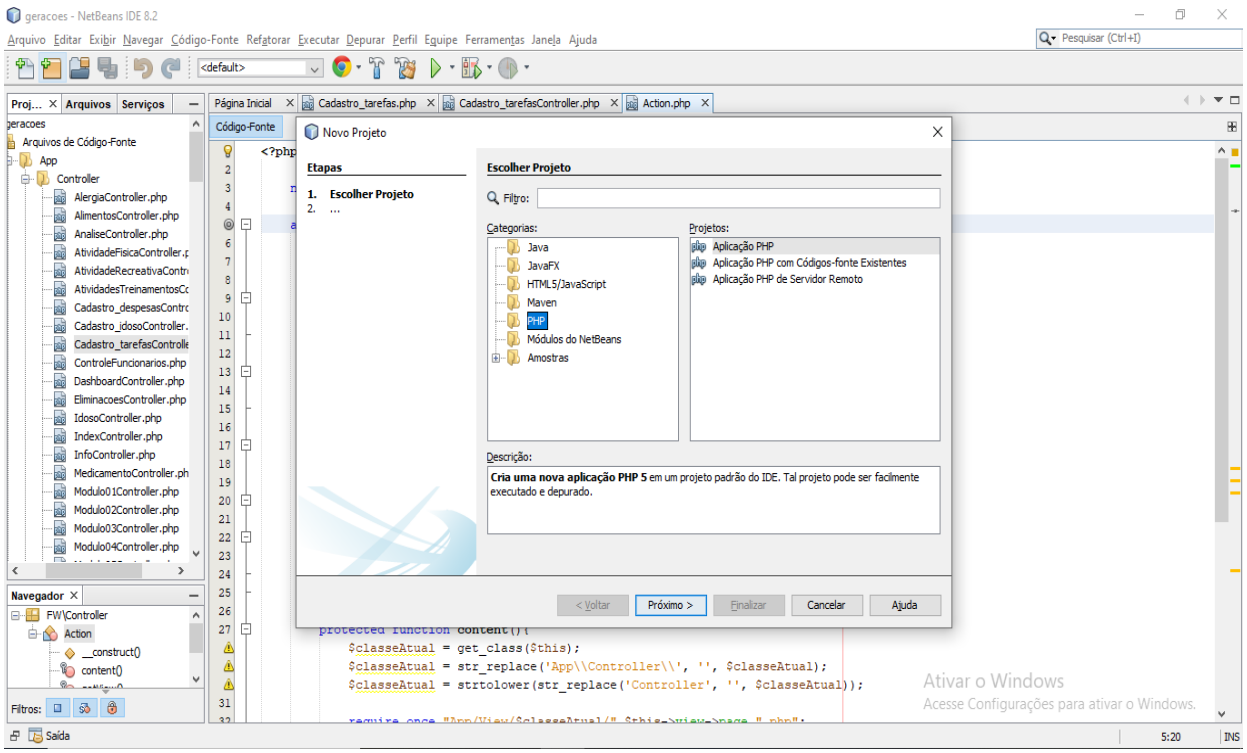
Figura 12 Logo Php.[9]

Diferente de algumas linguagens de programação, o PHP é gratuita e pode ser utilizado em várias plataformas diferentes como Windows, Linux e suas variantes, Mac OS e etc. Também é suportado pelos servidores web mais atuais como Apache, Microsoft Internet Information Server, Personal Web Server e etc. [10][11]

Antes de começar a usar o PHP precisamos de um servidor HTTP(WEB), como o Apache e um banco de dados como o Mysql ou MariaDB. E para ter essas duas funções, podemos baixá-las individualmente e configura-los a parte, ou podemos usar um programa chamado XAMPP que já faz a parte do servidor e banco de dados juntos.[10][11]

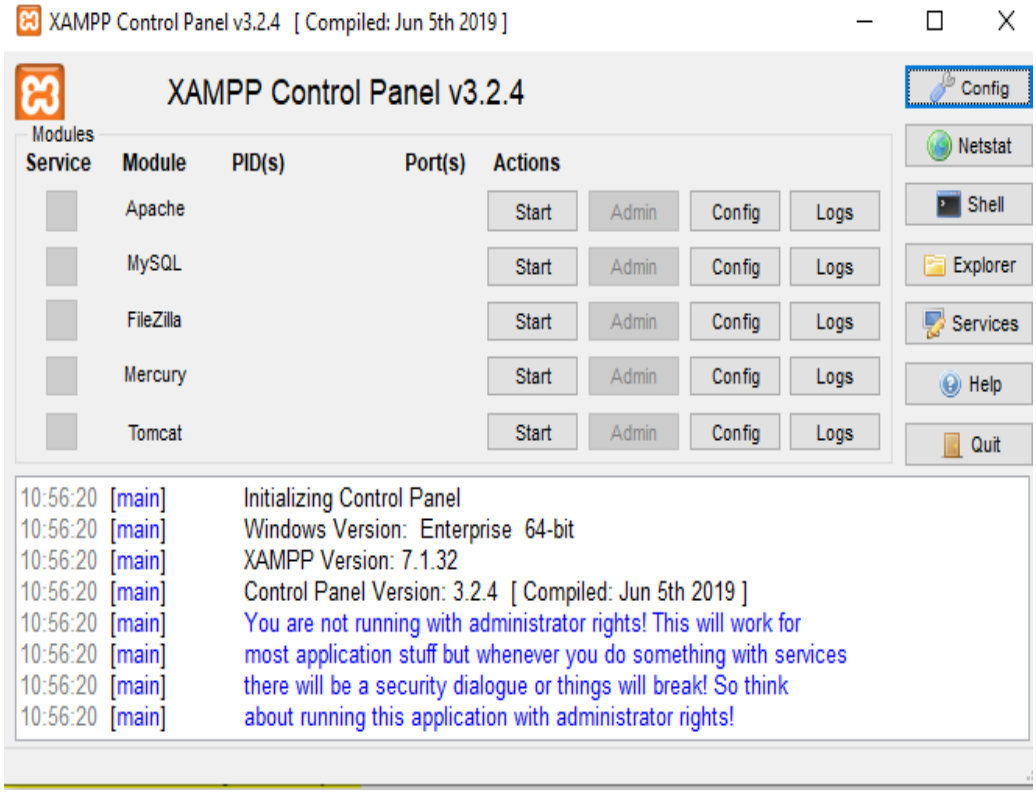
O PHP pode ser instalado de várias formas, com seu instalador ou manualmente e podemos encontrá-lo no site oficial do PHP www.php.net, ou podemos utilizar ele em alguma ferramenta que já tenha sua língua embutida. No caso utilizaremos a ferramenta NetBeans que já tem o PHP embutido em sua ferramenta. O Netbeans pode ser baixado no seguinte site https://www.oracle.com/technetwork/pt/java/javase/downloads/jdk-netbeans-jsp-3413153-ptb.html.[10][11]

Figura 13 Interface NetBeans. [Fornecido pelo Autor]



O XAMPP já vem com um instalador automático, podendo ser adquirido no site https://www.apachefriends.org/pt_br/index.html. Assim que o processo de instalação estiver concluído o XAMPP e o PHP estarão prontos para serem usados.[10][11]

Figura 14 Interface do XAMPP [Fornecida pelo autor]



Agora com a utilização do XAMPP e do PHP podemos começar a programar em PHP.[10][11]

- **Básicos do PHP**

  Todos os arquivos php terminam com a extensão .php, e geralmente o php é usado ao lado do HTML, porém existem vários pares de tags diferentes para abrir o mesmo.

Figura 15 Tags PHP.[10]

```php
<?php
  echo 'Primeiro par de tags PHP';
?>

<script language="php">
  echo 'Segundo par de tags PHP';
</script>

<?= $expressao = "Terceiro par de tags PHP" ?>
  Atalho <? echo $expressao; ?>

<%
  echo 'Quarto par de tags PHP - estilo ASP';
%>
```

Figura 16 PHP no HTML.[10]

```php
<html>
  <body>
    <h1> Exemplo de PHP </h1>
    <?php
      $a = 10;
      echo $a;
    ?>
  </body>
</html>
```

Assim como na linguagem C, todos os comandos precisam terminar com ponto e vírgula (;), porém o comando para fechar a tag php não necessita do mesmo, como vistos nos exemplos acima.[10][11]

O PHP assim como outras línguas de programação usa também variáveis, operadores e etc., porém há algumas coisas que não fora falado ainda que serão mostrados nesse tópico que também são essenciais a todas as línguas de programação, como por exemplo, Variáveis, Estruturas de controle e formulários.

- **Estruturas de Controle,Funções,Formulários**
  - Estruturas de controle

    Uma estrutura de controle, e um bloco lógico ou relacional de condições que podem ou não ser executadas. Para iniciar uma

estrutura de controle, você precisa utilizar if ,if else, for, while, do while, foreach.[10][11]

If e else if, são blocos de comando executados quando a condição precisa ser verdadeira ou falsa. No caso o if seria interpretado com SE, então SE x for verdadeiro então retorne TRUE, o else if ou só else agira como SE NÃO, então SE x for verdadeiro faça algo, SE NÃO faça isso.[10][11]

Figura 17 If else no projeto Gerações.

```php
<?php
  if ($this->getView()->idoso->__get('sexo') == 0) {
    echo '<input type="text" class="form-control w-100" name="sexo" id="sexo" value="Feminino" readonly>';
  }
  else{
    echo '<input type="text" class="form-control w-100" name="sexo" id="sexo" value="Masculino" readonly>';
  }
?>
```

While, e um bloco que executa comandos enquanto uma condição for verdadeira, já o do while e um pouco diferente, pois o while testa a condição logo no começo do código, já o mesmo testa no final do código.

Figura 18 Exemplo While[10]

```php
<?php
  $i=0;
  while ($i < 10) {
    echo "i = ".$i++;
  }
?>
```

Figura 19 Exemplo Do While[10]

```php
<?php
  $i=0;
  do{
    echo "i = ".$i++;
  }while ($i < 10);
?>
```

For e uma estrutura de controle na qual age como um contador e um loop, na qual ele vai verificar e executar quantas vezes fora posto até que a contagem acabe ou a condição seja falsa. O mesmo para ser inicializado precisa de três expressões, a inicialização, condição e incremento/decremento.[10][11]

Figura 20 Exemplo do For. [10]

```php
<?php
  for($i=1; $i<10; $i++){
    echo "\n i = ".$i;
  }
?>
```

Já o foreach percorre cada item de um array (uma variável com mais de um dado inserido nela, como por exemplo um vetor). Armazenando todos os seus respectivos valores.[10][11]

Figura 21 Exemplo declaração do foreach.[10]

```php
<?php
  $cursos = array('ADS','Automação','Matemática');

  foreach($cursos as $chave => $nome){
    echo "<br> curso $chave = $nome";
  }
?>
```

Figura 22 Exemplo de Foreach no projeto Gerações

```
<?php
foreach ($this->getView()->dados as $funcs) {
    ?>
<tr>


<td><?php echo $funcs->__get('USU_ID');?><input type="hidden" name="id" value="<?php echo $funcs->__get('USU_ID');?>"></td>
<td><?php echo $funcs->__get('USU_NOME');?></td>
<td><?php echo $funcs->__get('USU_PRONTUARIO');?></td>
<td><?php echo $funcs->__get('USU_CPF');?> </td>
<td><?php echo $funcs->__get('USU_FUNCAO');?></td>

<td><a href="/md8/lista_FuncionariosPorID?id=<?php echo $funcs->__get('USU_ID');?>" class="btn btn-outline-dark">Ficha</a></td>
```

- Funções

  Uma função, similar às estruturas de controle são divididas por bloco de comandos que pode ou não ser usada várias vezes no programa "chamando" a função. a função na sua criação recebe um nome, mas não e por que ela foi criada que ela vai ser executada diretamente, como dito antes, você terá que chama lá no código para que a mesma seja executada. Também devemos ficar atento, pois a mesma retorna um valor ao usuário, esse valor tem que ser de acordo com que o mesmo deseja que retorne.

Figura 23 Exemplo declaração de função [10]

```
function nome_da_função($arg_1, $arg_2, /* ..., */ $arg_n){
  //Bloco de comandos
  echo "Exemplo de função.\n";
  return $valor_retornado;
}
```

Figura 24 Exemplo de função projeto Gerações

```php
public function listar() {

    $uDAO = new UsuarioDAO();
    $uDAO->verify();

    $tipos = array();
    $sql = "SELECT * FROM tarefas_limpeza";
    $stmt = $this->getConn()->prepare($sql);
    $stmt->execute();
    $results = $stmt->fetchAll(\PDO::FETCH_ASSOC);

    foreach ($results as $row) {
        $tipo = new Cadastro_tarefas();
        $tipo->__set('id', $row['TAR_ID']);
        $tipo->__set('nome', $row['TAR_NOME']);
        $tipo->__set('descricao', $row['TAR_DESCRICAO']);
        $tipo->__set('observacao', $row['TAR_OBSERVACAO']);
        $tipo->__set('setor', $row['FK_SETORES_INSTITUICAO_SET_ID']);
        $tipo->__set('funcionario', $row['FK_SETOR_LIMPEZA_USU_ID']);
        array_push($tipos, $tipo);
    }

    return $tipos;
}
```

No caso a função já que a função e public (público), voltando a alguns conceitos anteriores, vemos que a mesma pode ser acessada por qualquer classe.

- Formulários

  Um dos motivos do por que o PHP é usado mais comumente com a linguagem de programação web HTML, e o jeito com que o mesmo trata os formulários em HTML. Ademais todos os formulários submetidos para um script PHP será automaticamente suas variáveis salvas nesse script e para isso existem dois meios de conseguir pegar os dados, por GET e POST.

Figura 25 Exemplo de formulário.[10]

```
<html>
  <body>
    <form action="recebe_dados.php" method="POST">
      Nome: <input type="text" name="nome"/>
      Idade: <input type="text" name="idade"/>

      <input type="submit" value="Enviar" name="enviar"/>
      <input type="reset" name="limpar" value="Limpar"/>
    </form>
  </body>
</html>
```

Figura 26 Exemplo formulário no projeto Gerações .

```
    <form method="post" action="/md8/cadastro_tarefas/cadastrar">
        <div class="text-center">
            <i class="far fa-calendar-alt fa-5x"></i></p>
        </div>
        <br>
        <div class="form-group">
            <div class="row">
                <div class="col-4 ml-auto">
                    <label>Data que deve ser realizada:</label>
                    <input type="date" class="form-control" name="datarealizar" id="datarealizar">
                </div>
                <div class="col-4 mr-auto">
                    <label>Nome:</label>
                    <input type="text" class="form-control" name="nometarefa" id="nometarefa" placeholder="Nome da
                </div>
            </div>
        </div>
</div>
</form>
```

- **Método Get**

Get e um padrão de envio de dados por formulário do html. No caso ele age como um distribuidor, ele passa as informações do formulário para outra página. No entanto o mal do método GET e que o mesmo envia suas informações pela URL tornando explícito as informações que estão passando pelas páginas.[10][11]

Figura 27 Exemplo de formulário com GET.[10]

```
<form action="recebe_dados.php" method="get">
  Nome: <input type="text" name="nome" />
  Idade: <input type="text" name="idade"/>

  <input type="submit" value="Enviar" name="enviar">
  <input type="reset" name="limpar" value="Limpar"/>
</form>
```

Figura 28 Exemplo de dados passando pela URL.[10]

```
http://www.meusite.com.br/recebe_dados.php?nome=Aluno&idade=19
```

Passados os dados para outra página, independente se por GET ou por POST (Método que será explicado a seguir), o método também serve para pegar as informações que foram enviados para a próxima página.[10][11]

Figura 29 Exemplo de captura de dados.[10]

```
<html>
  <body>
    Olá <?php echo $_GET["nome"]; ?><br>
    você tem<?php echo $_GET["idade"]; ?> anos.
  </body>
</html>
```

Saída:

```
Olá Aluno.
Você tem 19 anos.
```

Figura 30 Exemplo do método GET no projeto Gerações .

```php
public function lista_FuncionariosPorID() {
    $tipoDAO = new \App\DAO\FuncionariosDAO();

    $id = $_GET['id'];

    $funcs = $tipoDAO->listarPorID($id);

    $this->getView()->dados = $funcs;
    $this->render('ficha_Funcionario', 'dashboard', '../');
}
```

Apesar do GET ser eficiente não é seguro então sua utilização está sendo para capturas de dados que chegam de outras páginas, como no exemplo da imagem, utilização de captura do ID do funcionário que por ele será pego suas outras informações.

- **Método POST**

  POST similar ao GET e um método de formulário também utilizado em HTML para mandar informações. Porém seu grande diferencial é que os dados enviados pelo formulário são "escondidos" e não mostram na URL, isso tornando o mais seguro. Contudo ele também tem o benefício de conseguir enviar mais dados, ou seja, abrimos um leque de possibilidades, podendo ser mandado até imagens ou imagens.

Figura 31 Exemplo de utilização do método POST.[10]

```html
<form name="informacoes" method="POST" action="curso.php">

  <select name="curso">
    <option value="ADS" selected>
      Tecnologia em Análise e Desenvolvimento de Sistemas
    </option>
    <option value="Automação">
      Tecnologia em Automação Industrial
    </option>
```

Figura 32 Exemplo do POST no projeto Gerações .

```
public function alterar(){

    if(isset($_POST['idtarefa']))
    {
        $tarefa = new Cadastro_tarefas();
        $tarefa->__set('id',$_POST['idtarefa']);
        $tarefa->__set('nome',$_POST['nome']);
        $tarefa->__set('setor',$_POST['setor']);
        $tarefa->__set('funcionario',$_POST['funcionario']);
        $tarefa->__set('descricao',$_POST['descricao']);
        $tarefa->__set('observacao',$_POST['observacao']);

        $cDAO = new Cadastro_tarefasDAO();
        $cDAO->alterar($tarefa);
    }

    ?>
        <script>
            window.location.replace("/md8/lista_tarefas");
        </script>
    <?php
}
```

Por ser um método mais seguro, sua utilização foi para todos os formulários de cadastro para que possa tudo ser discreto sem que o usuário saiba as informações sendo passadas.

- **Definindo o MVC**

  O MVC (Model-View-Controller) e um padrão de arquitetura/projetos utilizados em programas que divide o projeto em camadas bem definidas. cada uma delas, no caso o Model, View e Controller executa o que lhe fora definido e nada mais. Com essas separações podemos definir o que cada cada irá fazer, no caso o Model e onde terá leitura e escrita de dados e também validações. O View será a camada de interação com o usuário e exibição de dados para o mesmo, normalmente esta página é feita em HTML juntamente com PHP. O Controller é responsável por receber as requisições e ter função de cada página que está atrelada a ele.[12][13][14].

  O uso do MVC para um projeto traz muitos benefícios como a organização de camadas, padronização na comunicação entre os objetos, e uma facilidade na hora de aplicar, porém temos que tomar cuidado pois o mesmo também traz males se aplicado de maneira incorreta como a

dificuldade de manutenção e se não for controlado pode desfazer todo o padrão e o projeto vira um caos.

Figura 33 Ilustração do MVC .[12]

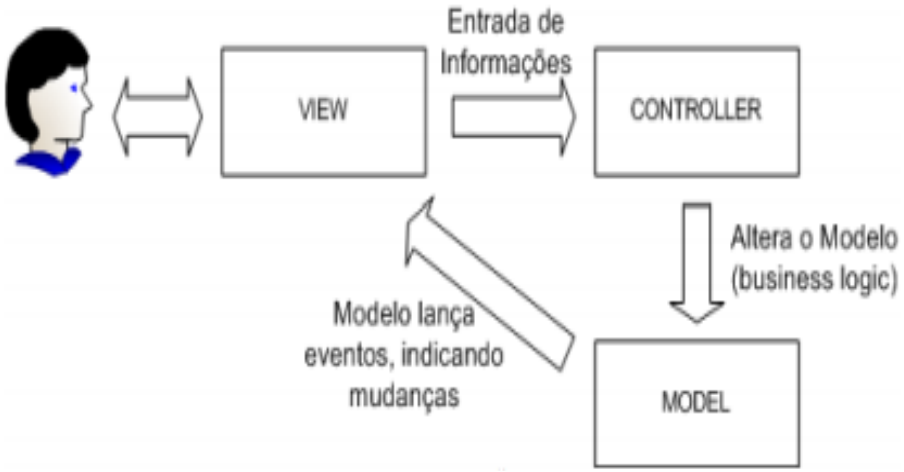
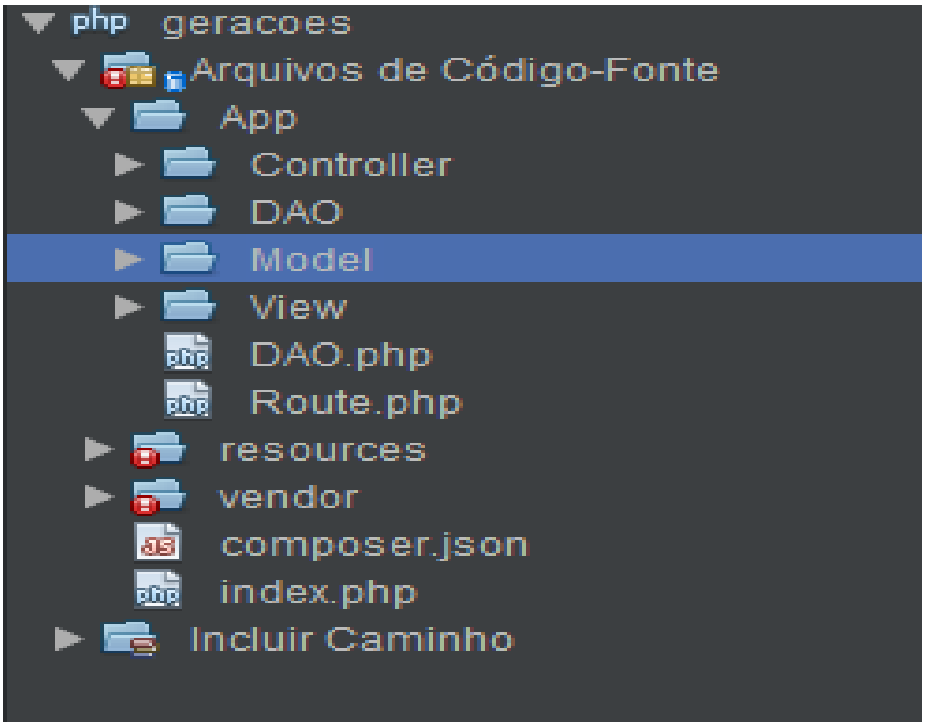


Figura 34 Exemplo dessa camada no projeto Gerações.

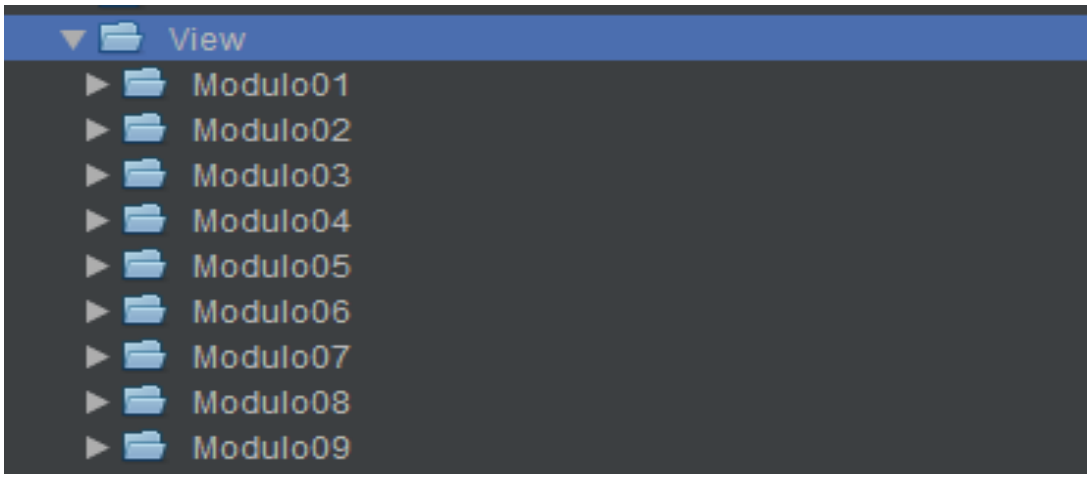


2.2.3 Ilustrar o MVC no Projeto Gerações

O MVC no projeto Gerações e uma ferramenta crucial para que todos os nove módulos consigam ter seu “espaço” para trabalhar, tendo a liberdade de trabalhar quando

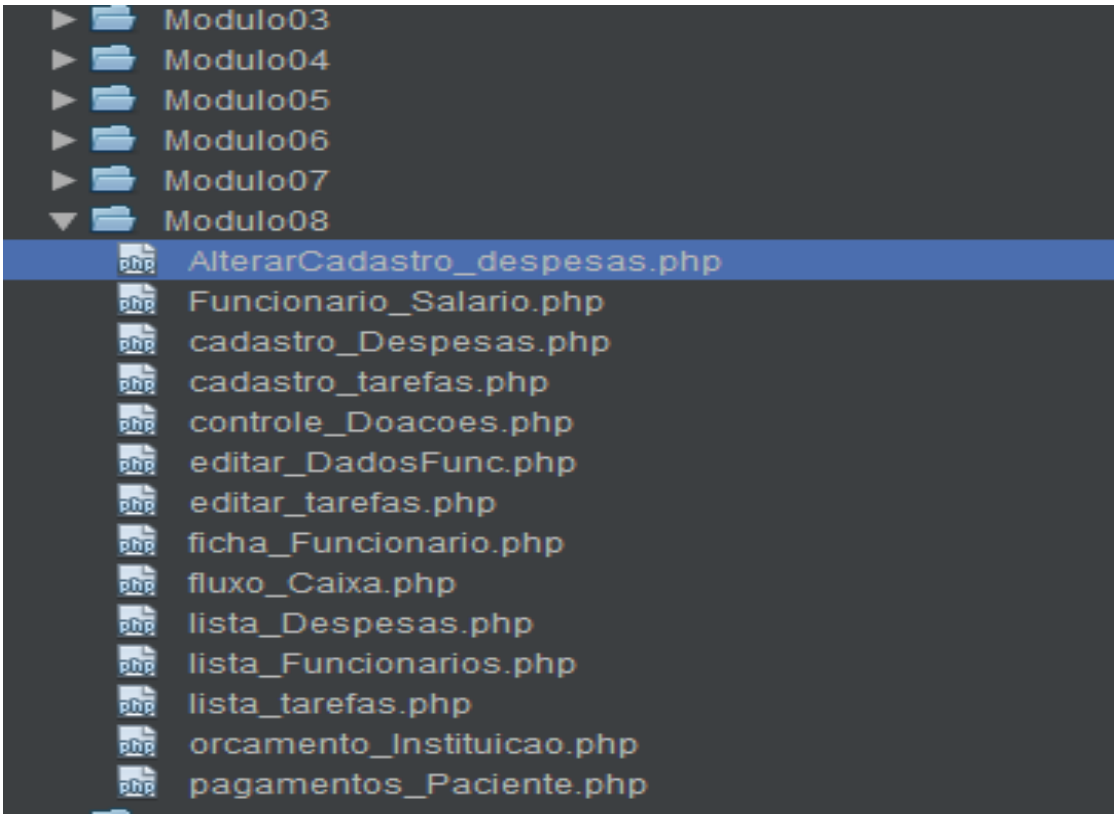
quiser sem atrapalhar os outros módulos. E como isso foi feito, primeiro o View é separado pelas páginas que serão necessárias de cada módulo como ilustrado no exemplo abaixo.

Figura 35 View Projeto Gerações.



Deste jeito os grupos têm liberdade para mexer sem suas páginas sem atrapalhar outros grupos

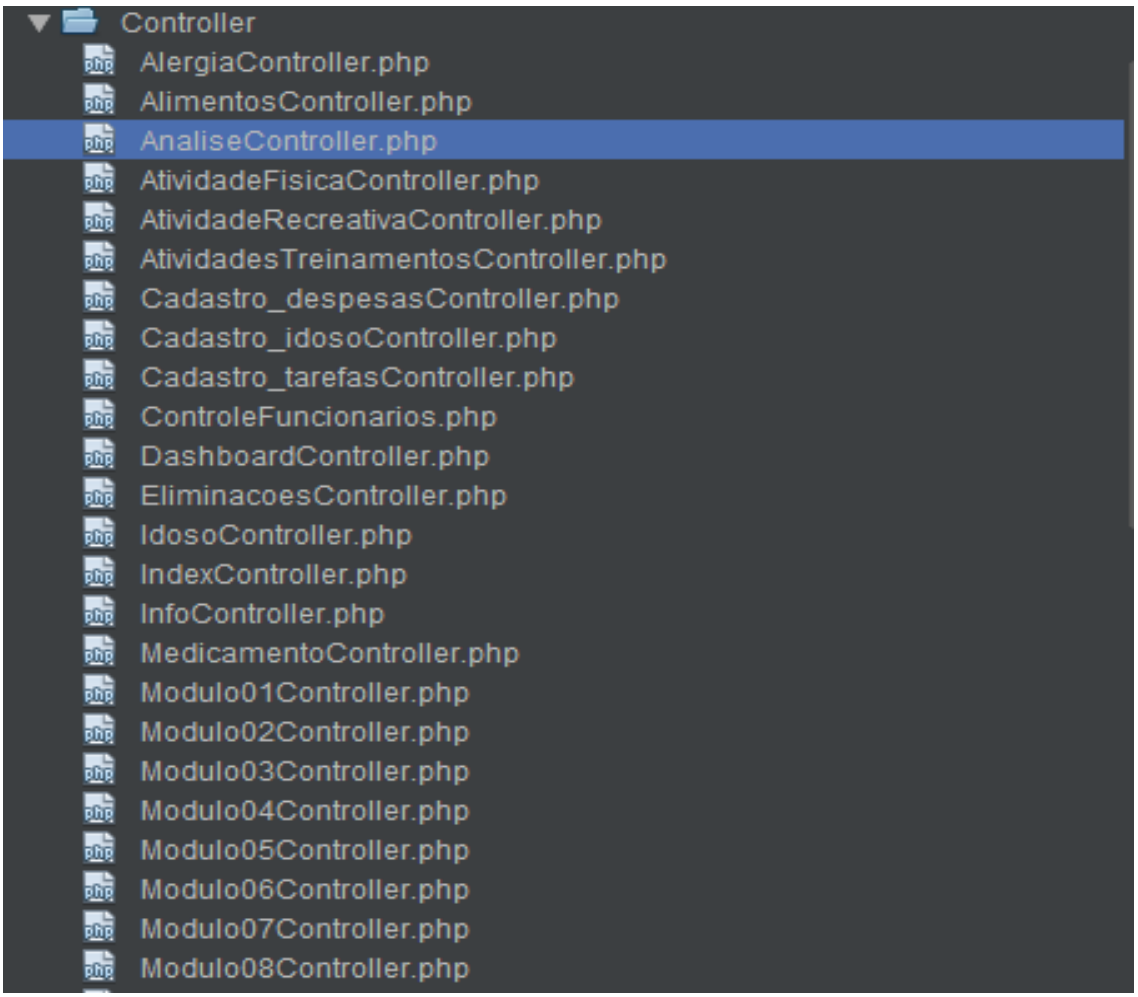
Figura 36 Páginas do View Projeto Gerações



.

O Controller segue um padrão similar ao View porem o mesmo possui um Controller geral para cada um dos módulos, e depois temos controladores diversos de todos os módulos.

Figura 37 Controller Projeto Gerações

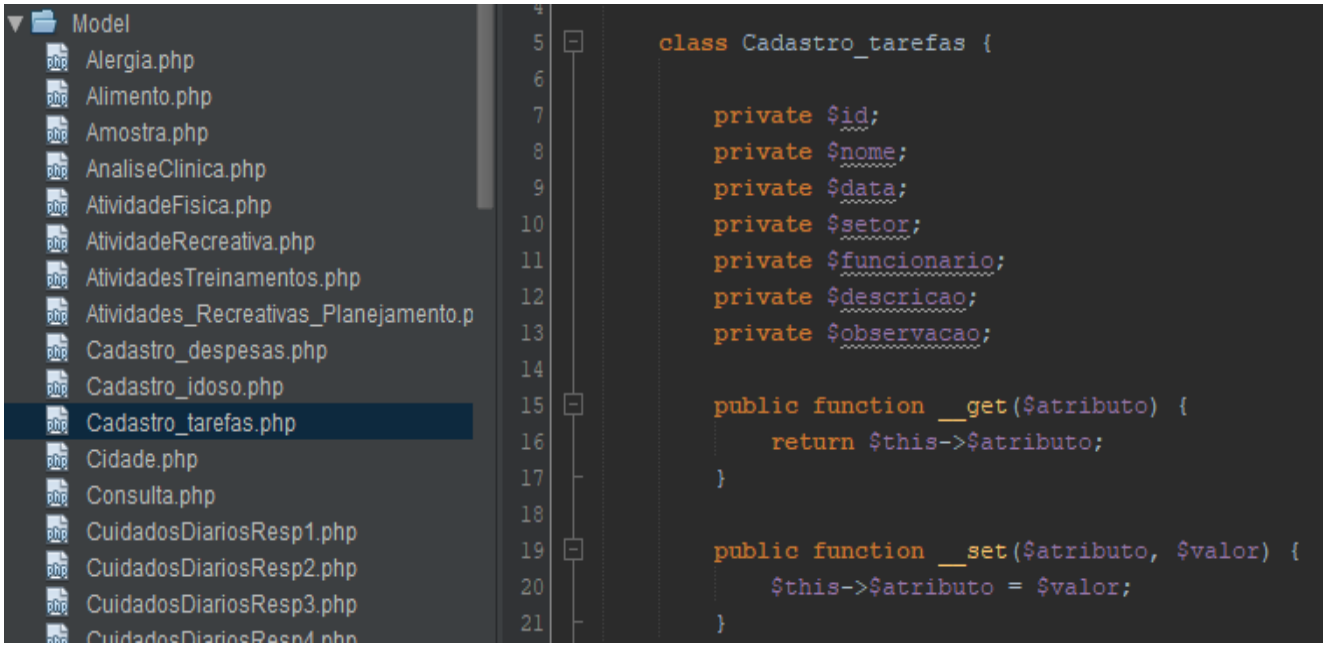


.

2.2.3.1 Detalhar o Model

O Model e onde vamos separar a lógica/dados ou melhor a lógica de negócio do projeto para que a mesma não fique confusa. A proposta é usar o Model como uma classe mãe, para que classes como por exemplo, uma classe de cadastro, consiga pegar os dados e inserir na tabela do banco de dados.

Figura 38 Exemplo Model projeto Gerações.



Como podemos ver na imagem, estamos no arquivo Cadastro_tarefas.php. Essa classe tem os dados de uma tarefa, no caso a tarefa em questão terá um Nome, Data, Sertor, Funcionário e etc. Note que no próximo exemplo teremos outra classe que irá herdar esses atributos, e essa classe que vai herdar será a classe que vai concluir o cadastro da inserção no banco.

Figura 39 Exemplo herança do modelo, projeto Gerações.

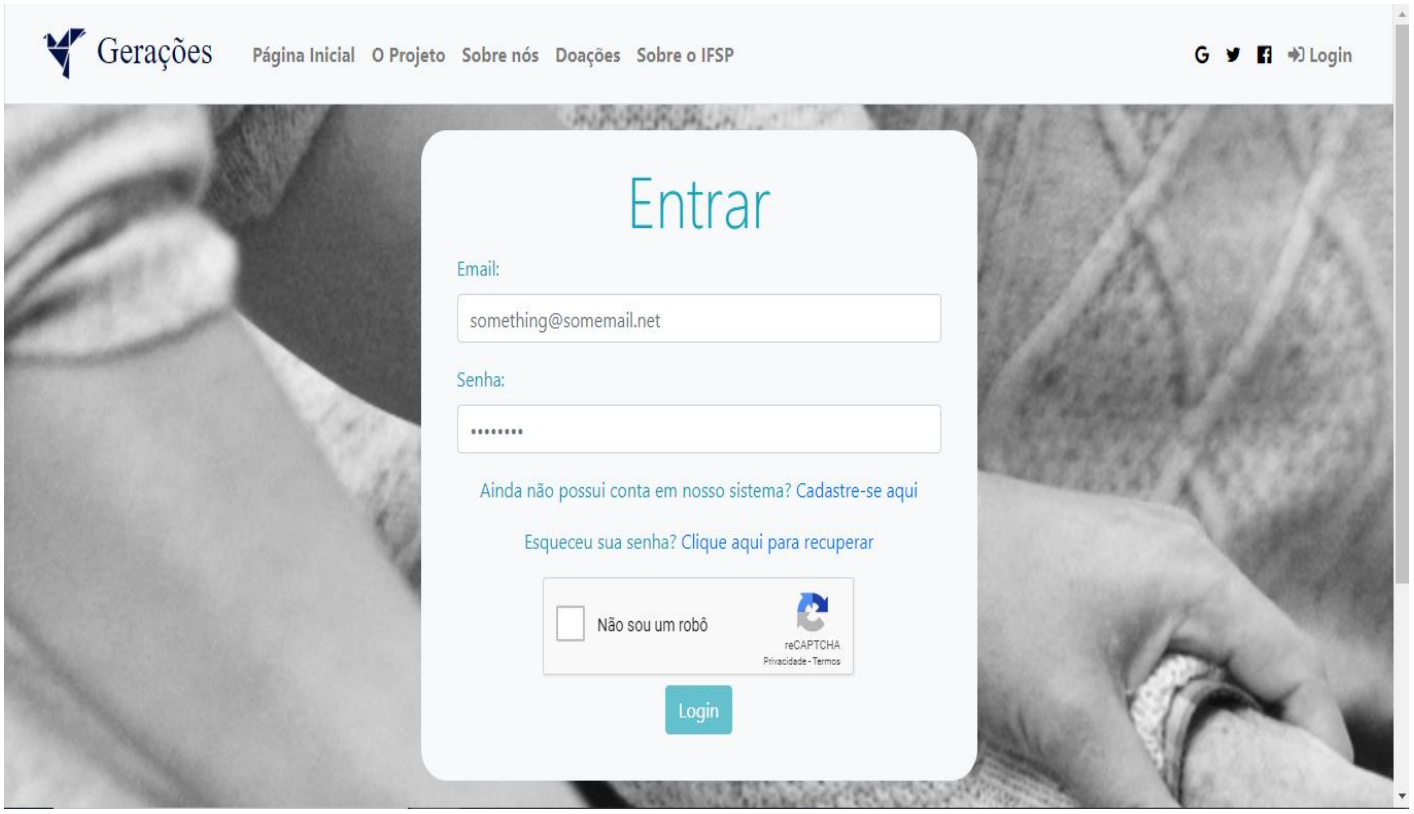
```
use App\DAO;
use App\Model\Cadastro_tarefas;

class Cadastro_tarefasDAO extends DAO {

    public function alterar($cadastrotarefa) {

        $uDAO = new UsuarioDAO();
        $uDAO->verify();

        $sql = "UPDATE tarefas_limpeza SET TAR_NOME=:NOME,FK_S

        $stmt = $this->getConn()->prepare($sql);

        $id = $cadastrotarefa->__get('id');
        $nome = $cadastrotarefa->__get('nome');
        $setor = $cadastrotarefa->__get('setor');
        $funcionario = $cadastrotarefa->__get('funcionario');
        $descricao = $cadastrotarefa->__get('descricao');
        $observacao = $cadastrotarefa->__get('observacao');

        $stmt->bindParam(":ID",$id);
        $stmt->bindParam(":NOME", $nome);
        $stmt->bindParam(":SETOR", $setor);
        $stmt->bindParam(":FUNCIONARIO", $funcionario);
        $stmt->bindParam(":DESCRICAO", $descricao);
        $stmt->bindParam(":OBSERVACAO", $observacao);
        $stmt->execute();
    }
```

Essa classe que chamamos de Cadastro_TarefasDAO, e a classe que vai pegar todos os dados da classe model para que possa ser feita a realização da inserção desses dados ao banco integrado.

2.2.3.2 Detalhar o View

O View ao similarmente ao Model, realiza “exatamente” o que seu nome diz, enquanto o Model serve como um modelo para as outras classes, o View vai ser a parte “estética” do projeto, vai ser a página que terá a interação do usuário, para que o usuário possa realizar o login, cadastrar algo e etc.

Figura 40 Tela de login do projeto Gerações.



Essa página, perceba que por ela ser de interação com o usuário, o usuário terá que informar o login e senha para que o mesmo possa entrar no sistema do projeto. Mas claro, por mais que seja uma parte de interação com o usuário, não significa que ela não tenha nem uma função por trás dela, esta página provavelmente se não tem a função para realizar o login, ela chama essa função para verificar se os dados são verdadeiros ou não.

2.2.3.3 Detalhar o Controller

O Controller, como o próprio nome diz, ele controla e diz quando as coisas devem ou não acontecer. Usado para intermediar a Model e a View. Por exemplo, para pegar dados da Model (guardados em um banco) e exibir na View (em uma página HTML), ou pegar os dados de um formulário (View) e enviar para alguém (Model). No exemplo a seguir, veremos o Controller no exemplo do Cadastro_tarefas citadas acima.

Figura 41 Controller Cadastro_tarefas projeto Gerações

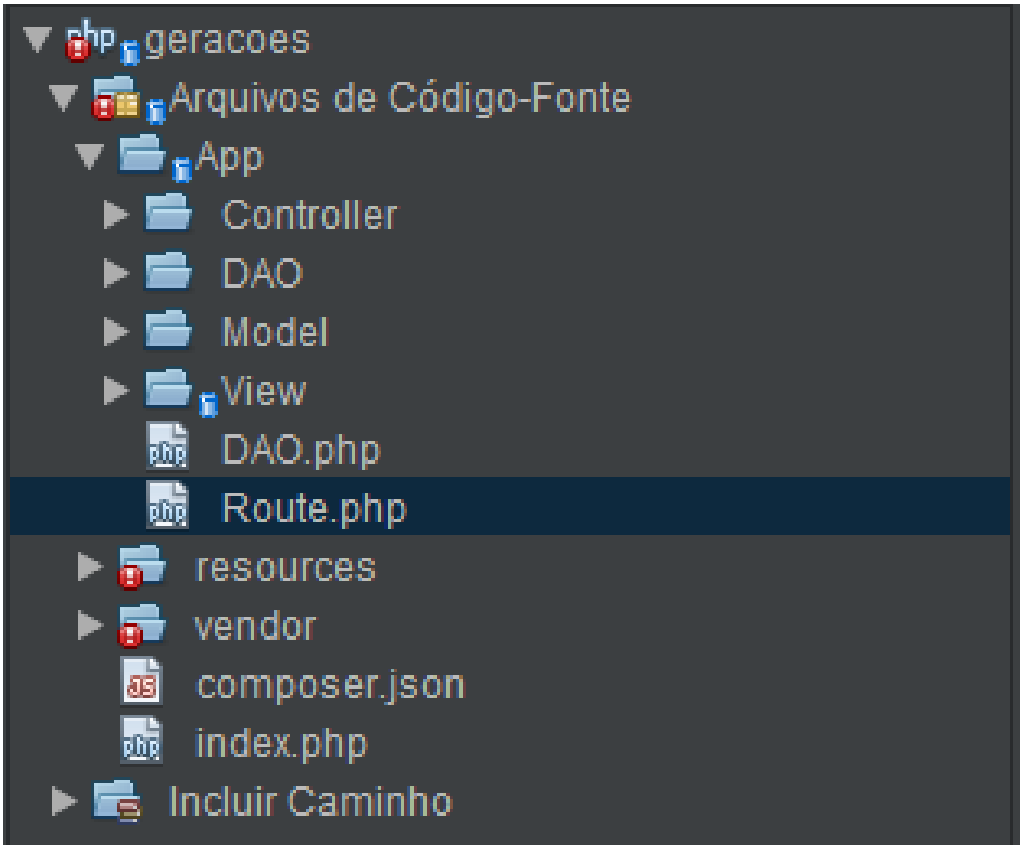
```php
use FW\Controller\Action;
use App\DAO\Cadastro_tarefasDAO;
use App\Model\Cadastro_tarefas;

class Cadastro_tarefasController extends Action{

    public function cadastrar(){

        if(isset($_POST['nometarefa'])&& isset($_POST['setor'])&& isset($_POST['funcionario'])&& isset($_POST['desc

            $cadastrotarefas = new Cadastro_tarefas();

            $cadastrotarefas->__set('nome', $_POST['nometarefa']);
            $cadastrotarefas->__set('setor', $_POST['setor']);
            $cadastrotarefas->__set('funcionario', $_POST['funcionario']);
            $cadastrotarefas->__set('descricao', $_POST['descricao']);
            $cadastrotarefas->__set('observacao', $_POST['observacao']);

            $cadastrotarefaDAO = new Cadastro_tarefasDAO();
            $cadastrotarefaDAO->inserir($cadastrotarefas);
        }

        header('Location: /md8/lista_tarefas');
        die();
    }
```

Vemos que pelo fato de o Controller interagir com o Model ele claramente tem que herdar seus atributos e também da nossa classe tem todos os modelos de funções. Neste exemplo vemos apenas uma das funções que está presente nesse Controller que é o Cadastro do Cadastro das tarefas.

2.2.3.4 Protótipo Finalizado

Como vimos alguns exemplos do projeto Gerações nesse trabalho, como alguns exemplos de Model, Controller e View e mais alguns exemplos de códigos em PHP etc. A utilização do MVC foi feita da seguinte maneira.

Figura 42 Divisão das camadas do MVC no projeto Gerações.



Vemos que temos as camadas do MVC, realizando suas determinadas funções, porém estamos também utilizando mais dois recursos. Um e o arquivo DAO.php e também uma camada chamada DAO, o mesmo e como se fosse um "Model" das principais funções que teremos ao longo do projeto como por exemplo, o cadastro de funcionários ou mesmo a sua exclusão ou sua edição.

Figura 43 DAO.php projeto Gerações.

```php
<?php

    namespace App;

    use FW\DB\Connection;


    abstract class DAO extends Connection{

        public abstract function inserir($obj);
        public abstract function excluir($obj);
        public abstract function alterar($obj);
        public abstract function buscarPorId($obj);
        public abstract function listar();

    }
```

Basicamente o DAO além de ser usado para chamar as funções como mostra no exemplo da figura 36, Ele também fará a comparação e a junção entre o Modelo que foi inserido através do formulário e o código do banco, para que todas as inserções no banco sejam feitas de forma correta.

Figura 44 Cadastro_TarefasDAO projeto Gerações.

```php
use App\DAO;
use App\Model\Cadastro_tarefas;

class Cadastro_tarefasDAO extends DAO {

    public function alterar($cadastrotarefa) {

        $uDAO = new UsuarioDAO();
        $uDAO->verify();

        $sql = "UPDATE tarefas_limpeza SET TAR_NOME=:NOME,FK_SETORES_INSTITUICAO_SET_ID=:SETOR,FK_SETOR_LIMPEZA_USU_ID=

        $stmt = $this->getConn()->prepare($sql);

        $id = $cadastrotarefa->__get('id');
        $nome = $cadastrotarefa->__get('nome');
        $setor = $cadastrotarefa->__get('setor');
        $funcionario = $cadastrotarefa->__get('funcionario');
        $descricao = $cadastrotarefa->__get('descricao');
        $observacao = $cadastrotarefa->__get('observacao');

        $stmt->bindParam(":ID",$id);
        $stmt->bindParam(":NOME", $nome);
        $stmt->bindParam(":SETOR", $setor);
        $stmt->bindParam(":FUNCIONARIO", $funcionario);
        $stmt->bindParam(":DESCRICAO", $descricao);
        $stmt->bindParam(":OBSERVACAO", $observacao);
        $stmt->execute();
    }
```

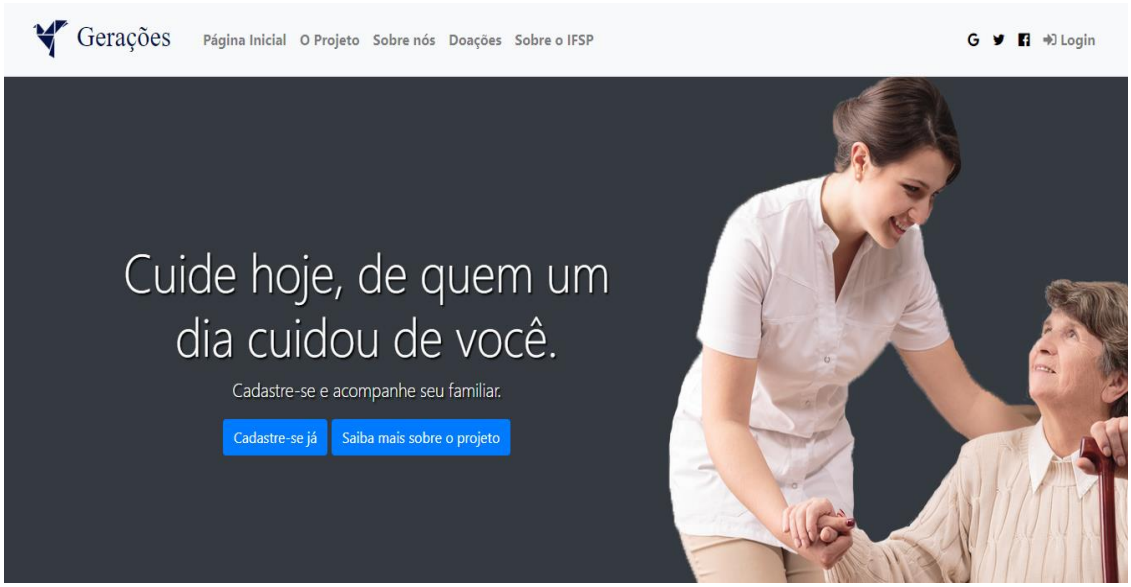
Já o Route, como o próprio nome sugere, são rotas, são as rotas de ligação entre todas as páginas e interfaces para o programa nunca se perder e ir para páginas erradas ou páginas que ele não deveria ir. Então como vemos na figura a seguir, cada módulo do projeto Gerações tem seu próprio campo, onde será inserido as Rotas de suas páginas e também as rotas dos códigos e funções que serão chamados por outras páginas.

Figura 45 Routes.php módulo 8 projeto Gerações

```
//MOD08
new RouteUnique("md8-index",                 "/md8/",                          "Modulo08Controller", "index"),
new RouteUnique("md8-cadastro_tarefas",      "/md8/cadastro_tarefas",          "Modulo08Controller", "cadastro_tarefas"),
new RouteUnique("md8-controle_Doacoes",      "/md8/controle_Doacoes",          "Modulo08Controller", "controle_Doacoes"),
new RouteUnique("md8-ficha_Funcionario",     "/md8/ficha_Funcionario",         "Modulo08Controller", "ficha_Funcionario"),
new RouteUnique("md8-fluxo_Caixa",           "/md8/fluxo_Caixa",               "Modulo08Controller", "fluxo_Caixa"),
new RouteUnique("md8-lista_Despesas",        "/md8/lista_Despesas",            "Modulo08Controller", "lista_Despesas"),
new RouteUnique("md8-cadastro_Despesas",     "/md8/cadastro_Despesas",         "Modulo08Controller", "cadastro_Despesas"),
new RouteUnique("md8-AlterarCadastro_despesas", "/md8/AlterarCadastro_despesas", "Modulo08Controller", "AlterarCadastro_Despesas"),
new RouteUnique("md8-lista_Funcionarios",    "/md8/lista_Funcionarios",        "Modulo08Controller", "lista_Funcionarios"),
new RouteUnique("md8-lista_FuncionariosPorID", "/md8/lista_FuncionariosPorID", "Modulo08Controller", "lista_FuncionariosPorID"),
new RouteUnique("md8-lista_tarefas",         "/md8/lista_tarefas",             "Modulo08Controller", "lista_tarefas"),
new RouteUnique("md8-editar_tarefas",        "/md8/editar_tarefas",            "Modulo08Controller", "editar_tarefas"),
new RouteUnique("md8-orcamento_Instituicao", "/md8/orcamento_Instituicao",     "Modulo08Controller", "orcamento_Instituicao"),
new RouteUnique("md8-pagamentos_Paciente",   "/md8/pagamentos_Paciente",       "Modulo08Controller", "pagamentos_Paciente"),
new RouteUnique("md8-editar_DadosFunc",      "/md8/editar_DadosFunc",          "Modulo08Controller", "editar_DadosFunc"),
new RouteUnique("md8-cadastro_concluido",    "/md8/cadastro_tarefas/cadastrar", "Cadastro_tarefasController", "cadastrar"),
new RouteUnique("md8-exclusao_concluida",    "/md8/lista-tarefas/excluir",      "Cadastro_tarefasController", "excluir"),
new RouteUnique("md8-alteracao_concluida",   "/md8/alterar-tarefas/tarefa",     "Cadastro_tarefasController", "alterar"),
```

E por fim a página inicial do projeto Gerações que e onde tudo começa.

Figura 46 Página inicial projeto Gerações



Conclusões e Recomendações

No mais vimos que apesar de precisarmos saber linguagens de programação diversas como PHP e HTML, e também alguns conceitos como Herança,Programação orientada a objetos, Estruturas de controle e etc, vemos que o MVC por sua estrutura “simples” entrega uma padronização onde se executada bem fará com que o projeto flua bem.

O modelo de padronização escolhido para o projeto Gerações foi o MVC que está como citado em vários exemplos durante este documento, facilitando a junção de todas as partes do projeto com base nas suas camadas.

Por fim, deixo minhas fortes recomendações sobre esse padrão de padronização, apesar de ter suas falhas também, as vantagens que são mais numerosas compensam o fato. Devo acrescentar que a experiência citada com a ferramenta e no projeto do Instituto Federal do campus de São João da Boa Vista, onde será aplicado em uma casa de idoso.

Referências Bibliográficas

[1] Portal da Rede Federal de Educação Profissional, Científica e Tecnológica. Histórico da Rede Federal de Educação Profissional, Científica e Tecnológica. Disponível em: http://redefederal.mec.gov.br/historico. Acesso em: 9 de Setembro de 2019.

[2] Portal da Rede Federal de Educação Profissional, Científica e Tecnológica. Expansão da Rede Federal de Educação Profissional, Científica e Tecnológica. Disponível em: http://redefederal.mec.gov.br/historico. Acesso em: 9 de Setembro de 2019.

[3] IBGE População cidade São João da Boa Vista Disponivel em: http://redefederal.mec.gov.br/historico. Acesso em: 9 de Setembro de 2019.

[4] Portal Acadêmico sobre disciplina Prática de Desenvolvimento de Sistemas. Disponivel em: https://sites.google.com/site/blromano/. Acesso em: 9 de Setembro de 2019.

[5] Portal Acadêmico sobre disciplina Prática de Desenvolvimento de Sistemas. Disponivel em: https://www.cos.ufrj.br/~sergio/ApostilaPascal.pdf. Acesso em: 2 de Outubro de 2019.

[6] Apostila E-tec Programação Orientada a Objetos, Victorio Albani de Carvalho Giovany Frossard Teixeira. Disponível em: http://redeetec.mec.gov.br/images/stories/pdf/eixo_infor_comun/tec_inf/081112_progr_obj.pdf. Acessado em: 3 de Outubro de 2019

[7] Conceitos de Programação J.Barbosa. Disponível em: https://web.fe.up.pt/~jbarbosa/ensino/meb/TGEI_2.pdf. Acessado em 7 de Outubro de 2019

[8] Introdução a programação Disponível em: h tt p:// bi bli otec a.virt ual.ufpb.br/files/introduaao_a_prog r amaaa o_146 31 50 04 7 .pd f Acessado em 7 de Outubro de 2019

[9] Site Oficial do PHP. Disponivel em: https://www.php.net/. Acesso em: 10 de Outubro de 2019.

[10] Apostila PHP pelo campus do Instituto Federal de São Paulo, campus Guarulhos. Disponivel em: http://www.dca.fee.unicamp.br/~glaucya/ifsp/ApostilaPHP_2014.pdf Acesso em 15 de Outubro de 2019

[11] Curso PHP por Maurício Vivas de Souza Barretos. Disponivel em: http://www.etelg.com.br/paginaete/downloads/informatica/php.pdf Acessado em: 15 de Outubro de 2019

[12] Model-View-Controller(MVC) por Rodrigo Rebouças de Almeida. Disponível em: http://www.dsc.ufcg.edu.br/~jacques/cursos/map/pdf/3.5-Model-View-Controller%20(MVC).pdf. Acessado em: 28 de Outubro de 2019

[13] Abordagem Teorico-Pratica MVC por Prof. Giuliano Prado M. Giglio. Disponível em: http://www.facom.ufu.br/~ronaldooliveira/PDS-2019-2/Aula12-MVC.pdf Acessado em: 28 de Outubro de 2019

[14] Padrões de Projeto por Vítor E.Silva Souza. Disponivel em: http://www.inf.ufes.br/~vitorsouza/wp-content/uploads/java-br-curso-padroesdeprojeto-slides05.pdf

Acessado em: 28 de Outubro de 2019