**FACULDADE DE TECNOLOGIA DE TAUBATÉ**

**João Marcos de Castilho Otabe**

##### Padrão de projeto MVC no Framework Laravel

**Taubaté**

**2018**

**João Marcos de Castilho Otabe**

##### Padrão de projeto MVC no Framework Laravel

Trabalho desenvolvido e entregue na disciplina de Programação para WEB, ministrada pelo Prof. Me Fernando Salles Claro, disciplina integrante do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Centro Estadual de Educação Tecnológica Paula Souza como parte integrante da avaliação para obtenção da média da disciplina, no 2º bimestre do 2º semestre de 2018.

**Professor  
Prof. Me Fernando Salles Claro**

**Taubaté**

**2018**

**SUMÁRIO**

[1. Conceitos Gerais Sobre MVC 4](#_Toc531180687)

[1.1. Model 4](#_Toc531180688)

[1.2. View 4](#_Toc531180689)

[1.3. Controller 5](#_Toc531180690)

[2. Laravael 5](#_Toc531180691)

[3. Criação de Model no Laravel 5](#_Toc531180692)

[3.1. Criando o Model 5](#_Toc531180693)

[3.2. Criando uma Migration 6](#_Toc531180694)

[1. Criação de Views no Laravel 7](#_Toc531180695)

[2. Criação de Controllers no Laravel 7](#_Toc531180696)

[3. Referências 9](#_Toc531180697)

# Conceitos Gerais Sobre MVC

O MVC é um padrão de projetos muito popular, pois divide o projeto em camadas, que são muito bem definidas, uma maneira de “quebrar” a aplicação em três pedaçossendo elas o Model, View e Controller. O uso do MVC trás muitos benefícios para o desenvolvedor, pois consegue isolar as regras de negócio da lógica de apresentação, a interface com o usuário. As divisões possibilitam mais flexibilidade e reuso das classes, pois abstraindo o desenvolvimento do projeto em três camadas, possibilita que o desenvolvimento do produto fique fácil e fluido, porque divide melhor as partes do projeto em pequenos módulos, cada um com uma parte específica no sistema.

As três camadas do MVC, são a Model, View e Controller e cada uma delas rem sua função dentro do projeto, o Modelo (Model) são as regras de negócio consultas a o banco de dados, a Visão (View) é a interface com o usuário e o Controlador (Controller) que é quem controla o sistema faz as ligações de diferentes partes do sistema.

Imagem 1 - Objetos utilizados no MVC e suas interações.

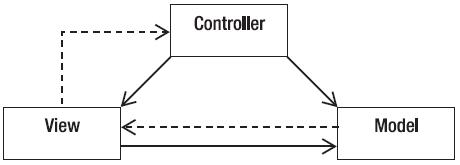


Imagem retirada do site www.devmedia.com.br.

Uma característica do MVC e de outros padrões de projetos em geral é a capacidade de ser utilizado em diferentes projetos. Agora os tópicos abaixo explanarão mais a respeito das camadas do MVC.

## Model

A camada de Modelo (Model) é a regra de negócio, manipulação de dados envolvendo as classes do sistema e acesso a os dados, ele é encarregado de ler e escrever os dados e valida-los.

## View

A camada de Visão (View) é usada para dar o visual a aplicação, o rosto dela, são as telas que serão apresentadas a o usuário, a interface com o usuário, nesta camada somente recursos visuais tem que ser implementados, as telas.

## Controller

A camada de Controle (Controller) é a intermediadora entre a camada Model, as regras de negócio e a camada View, que é ainterface com o usuário. A camada de controle realiza o processamento de dados informados pelo usuário e os transmitindo para as outras camadas

# Laravael

O Laravel é um framework para desenvolvimento em PHP de código aberto, visa o trabalho de maneira estruturada, organizado e rápido é um framework MVC, que é um dos vários padrões de projeto de software como MVW, MVVM. O Laravel tem a premissa de ser simples e fácil de aprender ele foi projetado com esse objetivo e é um dos frameworks mais populares de atualidade, seus desenvolvedores estão sempre ativos e são muito competentes, e tem uma comunidade grande e muito solícita.

# Criação de Model no Laravel

## Criando o Model

Toda tabela no banco de dados possui um Model correspondente que serve para a interação com essa tabela. O laravel tem integrado o Eloquente ORM que é uma técnica de desenvolvimento de software que utiliza tabelas do banco de dados em forma de objetos relacionais tornando assim mais abstrata e elegante a forma de interagir com o banco, facilitando a manutenção de dados. Lançando mão desta técnica o desenvolvedor não precisa se preocupar em escrever códigos em SQL já que a listagem e persistência é toda feita pela interface do ORM. Criando a Model, rodando o seguinte comando no teminal, se passar a flag –m é criada uma migration referente a Model.

php artisan make:model NameModel

php artisan make:model NameModel -m

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Product extends Model

{

protected $fillable = ['nome','descricao','quantidade','preco'];

protected $guarded = ['id', 'created\_at', 'update\_at'];

protected $table = 'products';

}

A model Product possui três variáveis protegidas, fillable, guarded e table. A variável fillable define quais os campos do sistema no banco que o usuário pode inserir, o campo guarded resguarda os campos de inserções, impedindo que alguém insira dados em alguns campos da tabela.

## Criando uma Migration

A migration é uma espécie controle de versão do banco de dados permitindo que sejam adicionados tabelas e colunas no B.D. O objetivo de uma migration é fornecer uma série de recursos como, manter um histórico de alterações que a base de dados sofre ao longo do tempo, uma vantagem de manter os históricos de alteração é poder voltar atrás de uma alteração feita. Rodando o comando artisan para criar a migration.

php artisan make:migration create\_nametable\_table

O código acima utiliza o comando artisan do Laravel para criar uma migration, ela é uma classe que irá tornar possível a manipular a tabela.

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateProductsTable extends Migration

{

public function up()

{

Schema::create('products', function (Blueprint $table) {

$table->increments('id');

$table->string('nome', 30);

$table->text('descricao')->nullable();

$table->string('quantidade');

$table->decimal('preco', 5, 2);

$table->timestamps();

$table->softDeletes();

});

}

public function down()

{

Schema::drop('products');

}

}

A function up está criando uma tabela products com os campos id, nome, descricao, quantidade e preco. O campo timestamps cria duas colunas no banco de dados, create\_at e update\_at, pegando o dia em que o usuário foi cadastrado quando ele faz alguma atualização no seu perfil, e softDeletes que criam uma coluna delete\_at, ao invés de deletar um registro no banco quando um usuário exclui sua conta ela altera o status para deleted e o registro é mantido, porem “desativado”. A função down derruba a tabela criada anteriormente. É interessante reparar que se pode definir coisas simples desde dizer que o campo descrição pode ser nulo até criar chaves primárias de maneira fácil.

# Criação de Views no Laravel

O Laravel possui uma tecnologia de template padrão, chamada blade, para usar a sintaxe das views basta adicionar a extensão blade, o arquivo da view será chamado “nomedoArquivoView.blade.php”, somente mudando o nome do arquivo. O blade possibilita várias funções como por exemplo usar herança entre views. Herdando da view padrão, que é criada automaticamente junto com o projeto, somente usando o @extends(‘app’).

@extends('app')

@section('content')

<h1>Lista de Produtos</h1>

<table class="table">

@foreach ($products as $product)

<tr>

<td>{{$product->nome}}</td>

<td>{{$product->descricao}}</td>

<td>{{$product->valor}}</td>

<td>{{$product->quantidade}}</td>

</tr>

@endforeach

</table>

@stop

# Criação de Controllers no Laravel

O Controller é quem vai ter todo o agrupamento da lógica das requisições em apenas uma única classe. Se todos os códigos ficassem direto nas rotas acabaria se tornando uma bagunça, os controllers ajudam na organização do projeto.

php artisan make:controller NameController

<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProductRequest;

use App\Product;

class ProductController extends Controller

{

public function index()

{

$products = Product::orderBy('created\_at', 'desc')->paginate(10);

return view('products.index',['products' => $products]);

}

public function create()

{

return view('products.create');

}

public function store(ProductRequest $request)

{

$product = new Product;

$product->nome = $request->nome;

$product->descricao = $request->descricao;

$product->quantidade = $request->quantidade;

$product->preco = $request->preco;

$product->save();

return redirect()->route('products.index')->with('message', 'Produto criado com sucesso!');

}

public function show($id)

{

//

}

public function edit($id)

{

$product = Product::findOrFail($id);

return view('products.edit',compact('product'));

}

public function update(ProductRequest $request, $id)

{

$product = Product::findOrFail($id);

$product->nome = $request->nome;

$product->descricao = $request->descricao;

$product->quantidade = $request->quantidade;

$product->preco = $request->preco;

$product->save();

return redirect()->route('products.index')->with('message', 'Produto atualizado com sucesso!');

}

public function destroy($id)

{

$product = Product::findOrFail($id);

$product->delete();

return redirect()->route('products.index')->with('alert-success','Produto deletado!');

}

}

No controller foram criadas as funções que uma CRUD deve fazer. Essas funções são chamadas de rotas que são uma espécie de métodos da controller. Criando as rotas no Laravel na versão 5.4 ou superiores:

<?php

...

Route::resource('products', 'ProductController')->middleware('auth');

...

?>

Se for o caso de uma rota especifica, pode defini-las separadamente, assim.

<?php

...

Route::group(['prefix' => 'admin', 'middleware' => 'auth'], function () {

Route::get('/', 'AdminController@getIndex');

Route::get('projetos', 'ProjetosController@getIndex');

Route::get('projetos/inserir', 'ProjetosController@getInserir');

Route::post('projetos/inserir', 'ProjetosController@postInserir');

Route::get('projetos/editar/{id}', 'ProjetosController@getEditar');

Route::post('projetos/editar/{id}', 'ProjetosController@postEditar');

Route::post('projetos/deletar/{id}', 'ProjetosController@postDeletar');

...

});

?>

# Referências

PROFISSIONAIS TI. **O conceito e as dúvidas sobre o MVC**. Disponível em <<https://www.profissionaisti.com.br/2014/10/o-conceito-e-as-duvidas-sobre-o-mvc/>> Acesso em 18 nov. 2018

DEVMEDIA. **A arquitetura MVC no desenvolvimento em PHP**. Disponível em <<https://www.devmedia.com.br/a-arquitetura-mvc-no-desenvolvimento-em-php/23121>> Acesso em 18 nov. 2018

DEVMEDIA. **Introdução ao Padrão MVC**. Disponível em <<https://www.devmedia.com.br/introducao-ao-padrao-mvc/29308>> Acesso em 18 nov. 2018

ESPECIALISTA TI BLOG. **Listando Registros no Laravel (MVC)**. Disponível em <<https://blog.especializati.com.br/listando-registros-no-laravel-mvc/>> Acesso em 21 nov. 2018

MEDIUM. **O que é Laravel? Porque usá-lo?**. Disponível em <https://medium.com/joaorobertopb/o-que-%C3%A9-laravel-porque-us%C3%A1-lo-955c95d2453d> Acesso em 21 nov. 2018

MEDIUM. **Criando um CRUD básico com Laravel 5.4**. Disponível em <<https://medium.com/trainingcenter/php-criando-um-crud-b%C3%A1sico-com-laravel-5-4-f17afe11c66e>> Acesso em 21 nov. 2018

IMASTERS. **Desenvolvendo migrations utilizando Laravel**. Disponível em <https://imasters.com.br/back-end/desenvolvendo-migrations-utilizando-laravel> Acesso em 21 nov. 2018

CLUBE DOS GEEKS. **Laravel: usando o ORM Eloquent**. Disponível em <<http://clubedosgeeks.com.br/programacao/php/laravel-usando-o-orm-eloquent>> Acesso em 24 nov. 2018

LARAVEL. **Views**. Disponível em <<https://laravel.com/docs/5.7/views>> Acesso em 24 nov. 2018

CAELUM . **Seja um artesão da web com Laravel, o framework MVC do PHP**. Disponível em <http://blog.caelum.com.br/seja-um-artesao-da-web-com-laravel-o-framework-mvc-do-php/ Acesso em 24 nov. 2018