**Curso Laravel 5.4 (Aulas 1 a 50)**

**SYS4SOFT (João Ribeiro)**

<https://www.youtube.com/watch?v=0T5gM1WRNsY&list=PLXik_5Br-zO893qVjjP7a4qg4NYrl33w1&index=1>

Resumo do curso feito por Roberto Pinheiro

No Google Chrome instale a extensão: json formatter.

# Aula 008 – Laravel Routing

**routes/web.php**

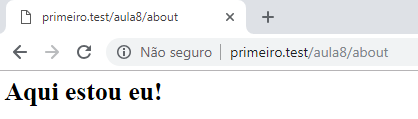
Route::get('aula8/about', function () {

return view('aula8/acerca\_de');

});

**views/aula8/acerca\_de.php**

<h1>Aqui estou eu!</h1>



# Aula 009 – Pequenos testes com Routes

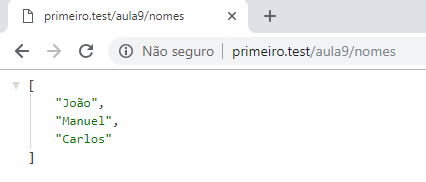
**routes/web.php**

Route::get('aula9/about', function () {

$nomes = ['João','Manuel','Carlos'];

return $nomes;

});





**routes/web.php**

Route::get('aula9/primeira', function () {

return view('aula9/primeira');

});

Route::get('aula9/segunda', function () {

return view('aula9/segunda');

});

Route::get('aula9/terceira', function () {

return view('aula9/terceira');

});

**views/aula9/primeira.php**

<h1>Primeira</h1>

<a href="segunda">Segunda</a>

**views/aula9/segunda.php**

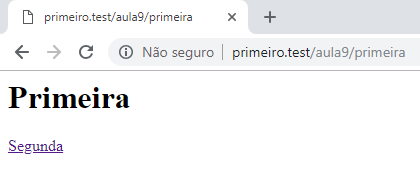
<h1>Segunda</h1>

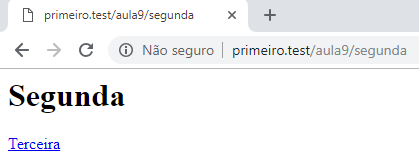
<a href="terceira">Terceira</a>

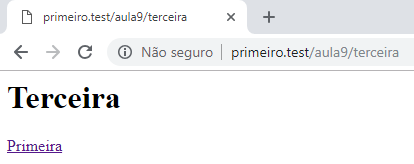
**views/aula9/terceira.php**

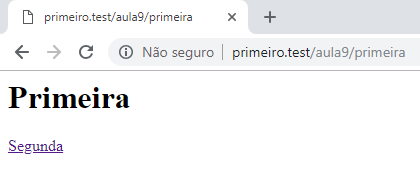
<h1>Terceira</h1>

<a href="primeira">Primeira</a>









# Aula 010 – Continuando a falar de routes simples

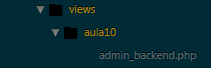
**routes/web.php**

/\* Routes de administração \*/

Route::get('aula10/admin', function () {

return view('aula10/admin\_backend');

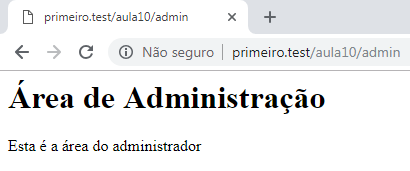
});



**views/aula10/admin.backend.php**

<h1>Área de Administração</h1>

<p>Esta é a área do administrador</p>



# Aula 011 – Passar valores através das Views

**routes/web.php**

Route::get('aula11/admin', function () {

return view('aula11/admin/admin');

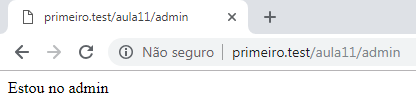
});

**views/aula11/admin/admin.php**

<?php

echo 'Estou no admin';

?>



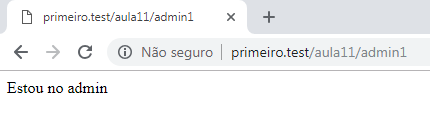


**routes/web.php**

Route::get('aula11/admin1', function () {

return view('aula11.admin.admin');

});





**routes/web.php**

Route::get('aula11/admin\_login', function () {

return view('aula11.admin.login');

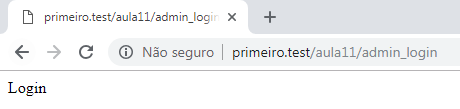
});

**views/aula11/admin/admin\_login.php**

<?php

echo 'Login';

?>





**routes/web.php**

Route::get('aula11/novo', function () {

$i = ['nome' => 'João Ribeiro', 'idade' => 43];

return view('aula11/novo',$i);

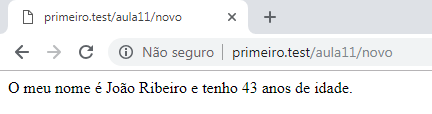
});

**views/aula11/novo.php**

<?php

echo "O meu nome é " . $nome . " e tenho " . $idade . " anos de idade.";

?>

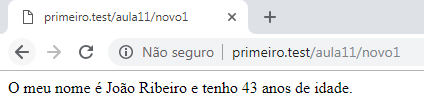




Route::get('aula11/novo1', function () {

return view('aula11/novo',['nome'=>'João Ribeiro'],['idade'=>43]);

});





**routes/web.php**

Route::get('aula11/novo2', function () {

return view('aula11/novo2')->with('nome','João Ribeiro');

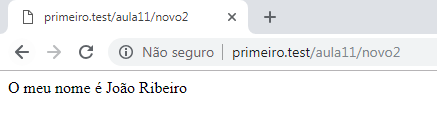
});

**views/aula11/novo2.php**

<?php

echo "O meu nome é " . $nome;

?>



# Aula 012 – Passar array multidimensional através das Views

**routes/web.php**

Route::get('aula12/medico', function () {

$medicos = [

'nome\_medico'=>'Antonio',

'especialidade'=>'Ortopedista'

];

return view('aula12/medico',$medicos);

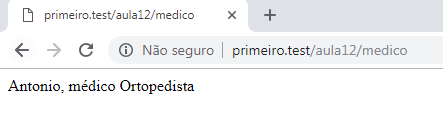
});

**views/aula12/medico.php**

<?php

echo "<p>$nome\_medico, médico $especialidade</p>";

?>





**routes/web.php**

Route::get('aula12/medicos', function () {

$medicos = [

0 => [

'nome\_medico'=>'Antonio',

'especialidade'=>'Ortopedista'

],

1 => [

'nome\_medico'=>'Maria',

'especialidade'=>'Geral'

],

2 => [

'nome\_medico'=>'Joaquim',

'especialidade'=>'Cirurgião'

]

];

return view('aula12/medicos') -> with('dados',$medicos);

});

**views/aula12/medicos.php**

<?php

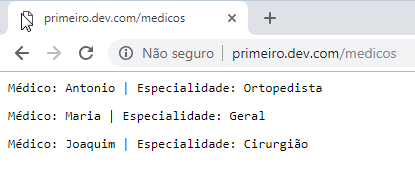
// var\_dump($dados);

foreach ($dados as $medico) {

echo "<p>Médico: " . $medico['nome\_medico'] . " | Especialidade: " . $medico['especialidade'] . "</p>";

}

?>



# Aula 013 – Routes com parâmetros

**routes/web.php**

Route::get('aula13/teste', function () {

$nomes = [

'João',

'Carlos',

'Ana'

];

$apelidos = [

'Rodrigues',

'Oliveira',

'Silva'

];

return view('aula13/teste',compact('nomes','apelidos'));

});

**views/aula13/teste.php**

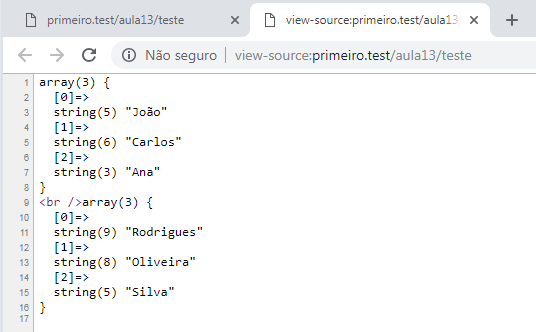
<?php

var\_dump($nomes);

echo "<br />";

var\_dump($apelidos);

?>





**routes/web.php**

Route::get('aula13/medicos\_da\_clinica/{id\_medico}', function ($id\_medico) {

return view('aula13/medicos') -> with('id', $id\_medico);

});

**views/aula13/medicos.php**

<?php

$medicos = [

0 => [

'nome\_medico'=>'Antonio',

'especialidade'=>'Ortopedista'

],

1 => [

'nome\_medico'=>'Maria',

'especialidade'=>'Geral'

],

2 => [

'nome\_medico'=>'Joaquim',

'especialidade'=>'Cirurgião'

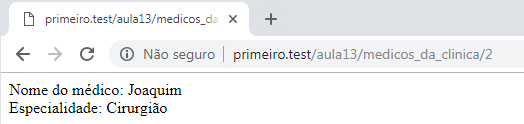
]

];

// echo $id;

echo "Nome do médico: " . $medicos[$id]['nome\_medico'] . "<br />Especialidade: " . $medicos[$id]['especialidade'];

?>



# Aula 014 – Routes com múltiplos parâmetros e parâmetro padrão

**routes/web.php**

Route::get('aula14/dados/{id\_medico?}/{id\_consulta?}', function ($id\_medico = 1, $id\_consulta = null) {

return view('aula14/dados') -> with('id\_medico', $id\_medico) -> with('id\_consulta', $id\_consulta);

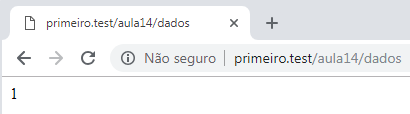
});

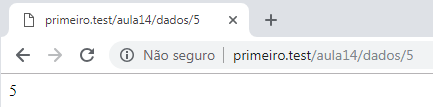
**views/aula14/dados.php**

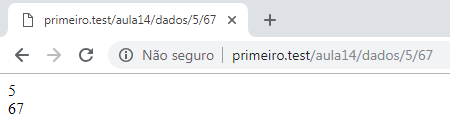
<?php

echo $id\_medico . '<br />' . $id\_consulta;

?>







# Aula 015 – Controllers

**routes/web.php**

Route::get('aula15/medicos', function () {

$medicos = [

'Joao',

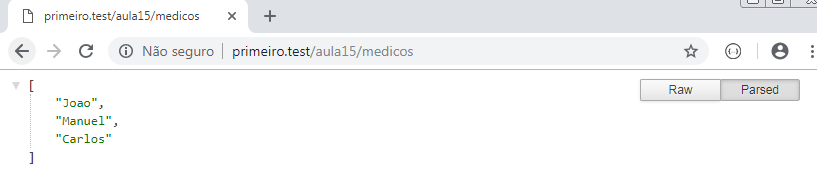
'Manuel',

'Carlos'

];

return $medicos;

});



## Criando um controller

No terminal do Laragon, na pasta do projeto (C:\laragon\www\primeiro), crie o controller chamado medicosController, na pasta aula15, entrando com o comando:

**php artisan make:controller aula15\medicosController**

**app/Http/Controllers/aula15/medicosController.php**

<?php

namespace App\Http\Controllers\aula15;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class medicosController extends Controller

{

//

}

**routes/web.php**

Route::get('aula15/medicosListar', ['uses' => 'aula15\medicosController@GetMedicos']);

Route::get('aula15/medicosApagar', ['uses' => 'aula15\medicosController@EliminarMedicos']);

**app/Http/Controllers/aula15/medicosController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class medicosController extends Controller

{

public function GetMedicos(){

$medicos = [

'Joao',

'Manuel',

'Carlos',

'Ana',

'Manuela'

];

return $medicos;

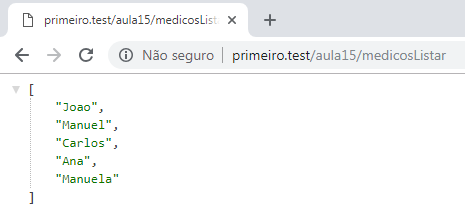
}

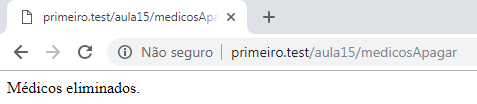
public function EliminarMedicos(){

return "Médicos eliminados.";

}

}





# Aula 016 – Nomeando Routes

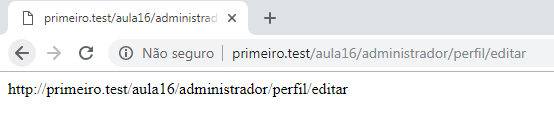
**routes/web.php**

Route::get('aula16/administrador/perfil/editar', array('as' => 'adm\_per\_edit', function(){

// return "Esta route foi renomeada";

return route('adm\_per\_edit');

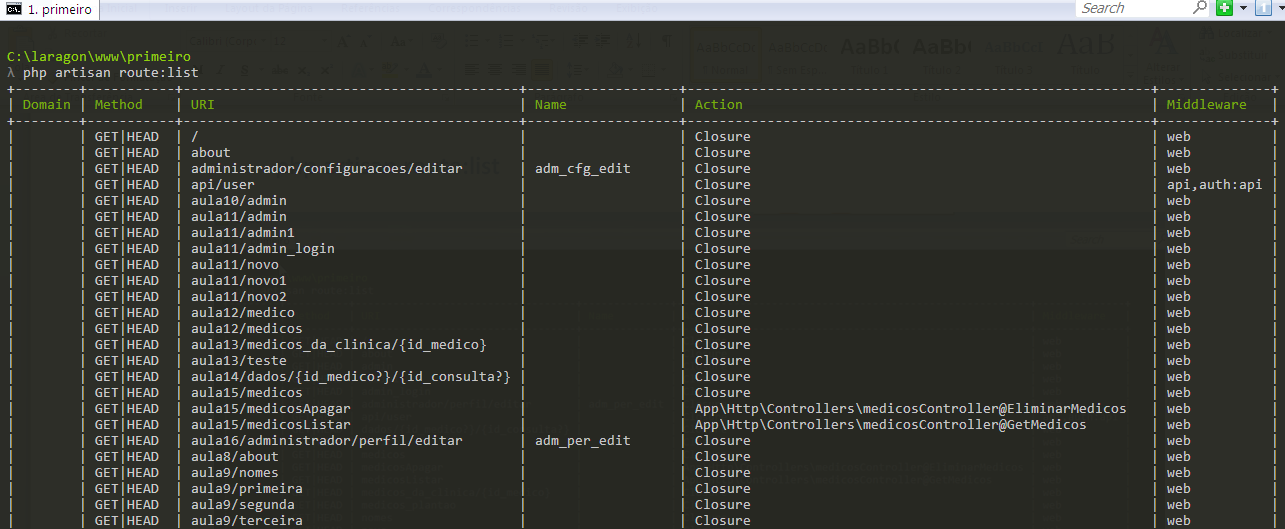
}));



## Listando as routes criadas

No terminal do Laragon, na pasta do projeto (C:\laragon\www\primeiro), entre com o comando:

**php artisan route:list**



# Aula 017 – Utilização de Routes nomeadas

**routes/web.php**

Route::get('aula17/administrador/configuracoes/editar', array('as' => 'adm\_cfg\_edit', function(){

return view('aula17/admin/config/editar');

}));



**views/aula17/admin/config/editar.php**

<?php

echo '<h1>Estou aqui!</h1>';

?>

No arquivo welcome.blade.php,inserir a linha em negrito:

**welcome.blade.php**

<div class="links">

<a href="https://laravel.com/docs">Documentation</a>

<a href="https://laracasts.com">Laracasts</a>

<a href="https://laravel-news.com">News</a>

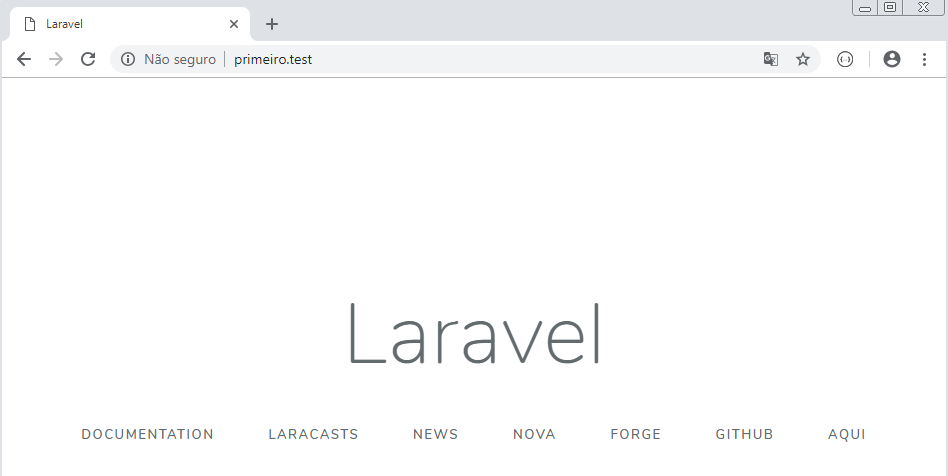
<a href="https://nova.laravel.com">Nova</a>

<a href="https://forge.laravel.com">Forge</a>

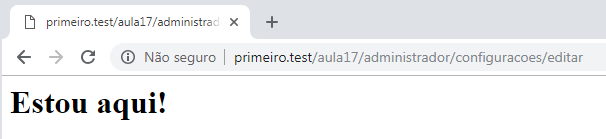
<a href="https://github.com/laravel/laravel">GitHub</a>

**<a href="{{ route('adm\_cfg\_edit') }}">AQUI</a>**

</div>



Clicando no link AQUI:

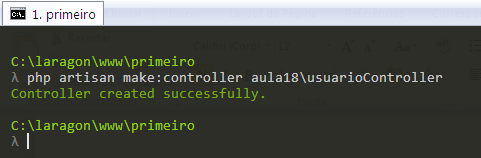


# Aula 018 – Passando dados entre Routes e Controllers

## Criando o controller

No terminal do Laragon, na pasta do projeto (C:\laragon\www\primeiro), crie o controller chamado usuarioController, na pasta aula18, entrando com o comando:

**php artisan make:controller aula18\usuarioController**



**routes/web.php**

Route::get('aula18/ver\_usuario/{nome}/{senha}', 'aula18\usuarioController@ApresentarUsuario');

**app/Http/Controllers/aula18/usuarioController.php**

<?php

namespace App\Http\Controllers\aula18;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class usuarioController extends Controller

{

function ApresentarUsuario($nome, $senha){

return "<p>Usuario: " . $nome . "</p><p>Senha: " . $senha . "</p>";

}

function Tratar($usuario, $senha){

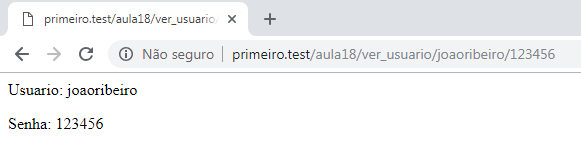
$usuario\_tratado = "Bem vindo " . $usuario;

$senha\_tratada = md5($senha);

return view('mostrar\_usuario', compact('usuario\_tratado', 'senha\_tratada'));

}

}



# Aula 019 – De Route para Controller e de Controller para View

**routes/web.php**

Route::get('aula18/tratar\_usuario/{usuario}/{senha}', 'aula18\usuarioController@Tratar');

**app/Http/Controllers/aula18/usuarioController.php**

<?php

namespace App\Http\Controllers\aula18;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class usuarioController extends Controller

{

function ApresentarUsuario($nome, $senha){

return "<p>Usuario: " . $nome . "</p><p>Senha: " . $senha . "</p>";

}

function Tratar($usuario, $senha){

$usuario\_tratado = "Bem vindo " . $usuario;

$senha\_tratada = md5($senha);

return view('aula18/mostrar\_usuario', compact('usuario\_tratado', 'senha\_tratada'));

}

}

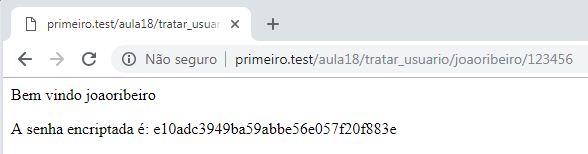
**mostrar\_usuario.php**

<?php

echo "<p>" . $usuario\_tratado . "</p>";

echo "<p>A senha encriptada é: " . $senha\_tratada;

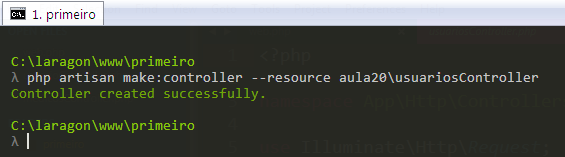
?>



# Aula 020 – Resource Controller

No terminal do Laragon, na pasta do projeto (C:\laragon\www\primeiro), crie o controller chamado usuariosController, na pasta aula20, entrando com o comando:

**php artisan make:controller --resource aula20\usuariosController**



Será criado um controller com métodos CRUD:

**routes/web.php**

Route::get('indice', function(){

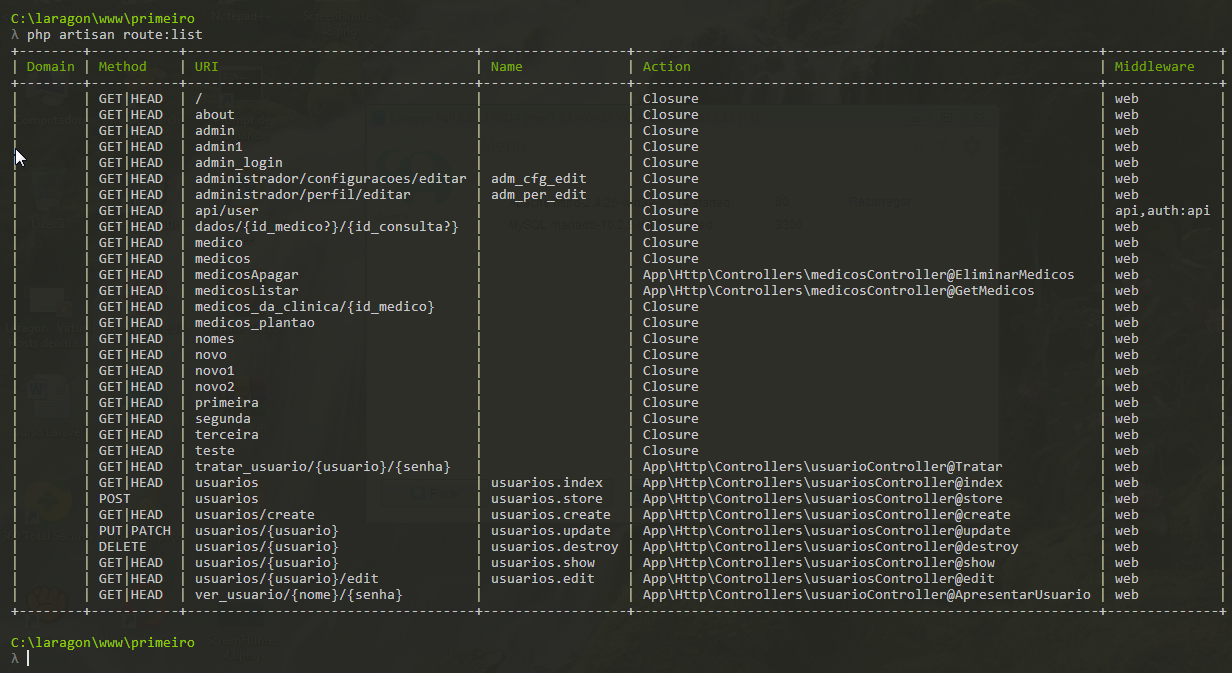
return view('indice');

});

Route::resource('usuarios', 'usuariosController');

No terminal do Laragon, na pasta do projeto (C:\laragon\www\primeiro), entre com o comando:

**php artisan route:list**



**views/aula20/índice.php**

<?php

$id\_usuario = 10;

echo "<a href=" . route('usuarios.index') . ">Index</a><br />";

echo "<a href=" . route('usuarios.create'). ">Create</a><br />";

echo "<a href=" . route('usuarios.show', $id\_usuario) . ">Show</a>";

?>

**app/Http/Controllers/aula20/usuariosController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class usuariosController extends Controller

{

public function index()

{

return "Estou no index.";

}

public function create()

{

return "Estou no create.";

}

public function store(Request $request)

{

//

}

public function show($id)

{

return "O usuário a apresentar tem o ID: " . $id;

}

public function edit($id)

{

//

}

public function update(Request $request, $id)

{

//

}

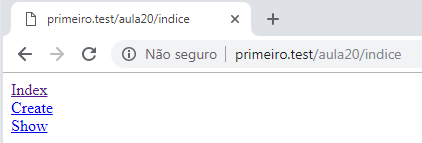
public function destroy($id)

{

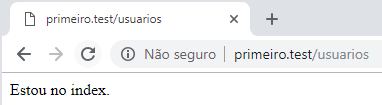
//

}

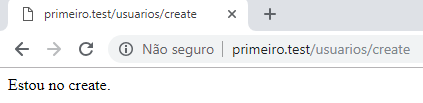
}



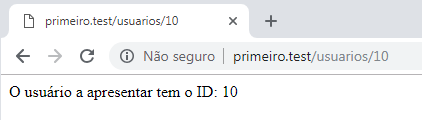
Clicando em “Index”:



Clicando em “Create”:



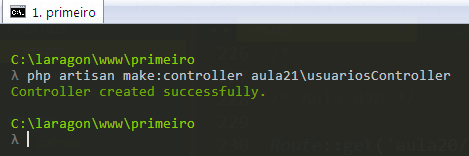
Clicando em “Show”:



# Aula 021 – Blade Template Engine

No terminal do Laragon, na pasta do projeto (C:\laragon\www\primeiro), crie o controller chamado usuariosController, na pasta aula21, entrando com o comando:

**php artisan make:controller aula21\usuariosController**



**routes/web.php**

Route::get('aula21/usuarios', 'aula21\usuariosController@MostrarUsuarios');

**app/Http/Controllers/aula21/usuariosController.php**

<?php

namespace App\Http\Controllers\aula21;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class usuariosController extends Controller

{

public function MostrarUsuarios(){

$usuarios = [

'Joao',

'Maria',

'Antonio'

];

// $usuarios = [];

// $usuarios = ['João'];

// apresentar a view com os usuários

return view('aula21/show\_usuarios', compact('usuarios'));

}

}

**views/aula21/show\_usuarios.blade.php**

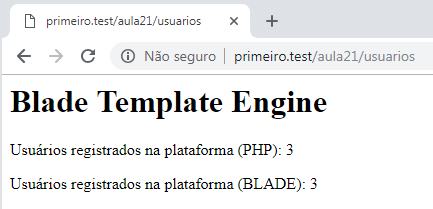
<div>

<h1>Blade Template Engine</h1>

<p>Usuários registrados na plataforma (PHP): <?php echo count($usuarios); ?></p>

<p>Usuários registrados na plataforma (BLADE): {{count($usuarios)}}</p>

</div>



# Aula 022 – Blade – Sintaxe básica

No terminal do Laragon, na pasta do projeto (C:\laragon\www\primeiro), crie o controller chamado usuariosController, na pasta aula21, entrando com o comando:

**php artisan make:controller aula22\usuariosController**

**app/Http/Controllers/aula22/usuariosController.php**

<?php

namespace App\Http\Controllers\aula22;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class usuariosController extends Controller

{

public function MostrarUsuarios(){

$usuarios = [

'Joao',

'Maria',

'Antonio'

];

// $usuarios = [];

// $usuarios = ['João'];

// apresentar a view com os usuários

return view('aula22/show\_usuarios', compact('usuarios'));

}

}

**routes/web.php**

Route::get('aula22/usuarios', 'aula21\usuariosController@MostrarUsuarios');

**views/aula22/show\_usuarios.blade.php**

<div>

<h1>Blade Template Engine</h1>

{{-- Bloco de código em PHP puro --}}

@php

// Esse é um comentário de PHP puro

echo '<p>Usuários inscritos na plataforma: ' . count($usuarios) . '</p>';

@endphp

{{-- Bloco de código condicional - if statement --}}

@if(count($usuarios) === 0)

<p>Não existem usuários inscritos na plataforma</p>

@elseif(count($usuarios) === 1)

<p>Há apenas um usuário inscrito na plataforma</p>

@else

<p>Há {{count($usuarios)}} usuários inscritos</p>

@endif

@unless(count($usuarios) === 0)

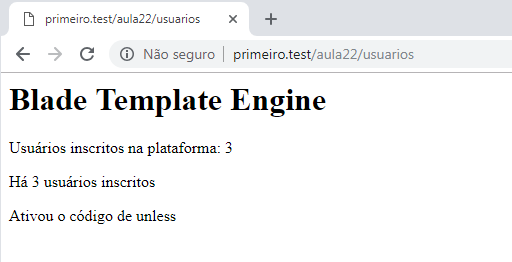
<p>Ativou o código de unless</p>

@endunless

</div>

unless significa “a não ser que”

unless 🡪 se não for



# Aula 023 – Blade Snippets

No Sublime Text Instale o plugin “**blade snippets**”

**routes/web.php**

Route::get('aula23/usuarios', 'aula23\usuariosController@MostrarUsuarios');

**views/aula23/show\_usuarios.blade.php**

<div>

<h1>Blade Template Engine</h1>

{{-- Bloco de código em PHP puro --}}

@php

// Esse é um comentário de PHP puro

echo '<p>Usuários inscritos na plataforma: ' . count($usuarios) . '</p>';

@endphp

{{-- Bloco de código condicional - if statement --}}

@if(count($usuarios) === 0)

<p>Não existem usuários inscritos na plataforma</p>

@elseif(count($usuarios) === 1)

<p>Há apenas um usuário inscrito na plataforma</p>

@else

<p>Há {{count($usuarios)}} usuários inscritos</p>

@endif

@unless(count($usuarios) === 0)

<p>Ativou o código de unless</p>

@endunless

@php

$nome = "João";

@endphp

<!-- <p>{{ $nome }}</p> -->

@isset($nome)

<p>A variavel $nome está definida e tem o valor {{$nome}}</p>

@endisset

@empty($nome)

<p>A variavel $nome está definida e tem o valor {{$nome}}</p>

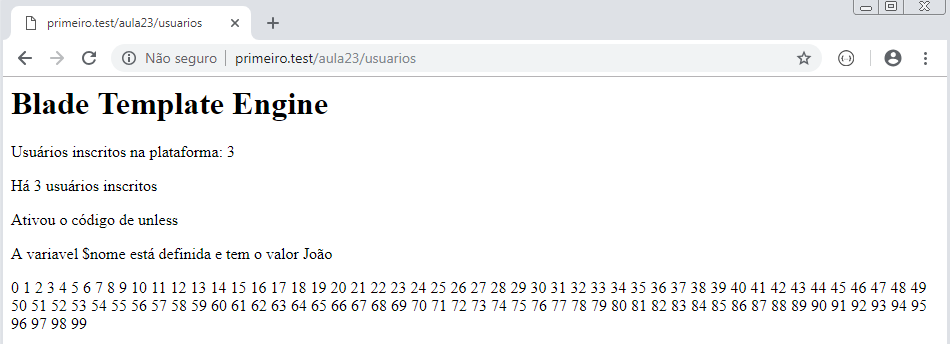
@endempty

@for($i = 0; $i < 100; $i++)

{{$i}}

@endfor

</div>



# Aula 024 – Blade Layouts

Dentro da pasta views 🡪 aula24, criar a pasta layouts. E dentro da pasta layouts criar o arquivo base\_layout.blade.php

**views/aula24/layouts/base\_layout.blade.php**

<!DOCTYPE html>

<html>

<head>

<title>Título da página</title>

</head>

<body>

{{-- header --}}

<div>

@yield('cabecalho')

</div>

{{-- content --}}

<div>

@yield('conteudo')

</div>

{{-- footer --}}

<div>

@yield('rodape')

</div>

</body>

</html>

**routes/web.php**

Route::get('aula24/contato', function(){

return view('aula24/contato');

});

Route::get('aula24/about', function(){

return view('aula24/about');

});

Dentro da pasta views🡪aula24 criar dois arquivos:

contato.blade.php e about.blade.php.

**views/aula24/contato.blade.php**

@extends('layouts.base\_layout')

@section('cabecalho')

<p>Este é o cabeçalho da página dos contatos</p>

@stop

@section('conteudo')

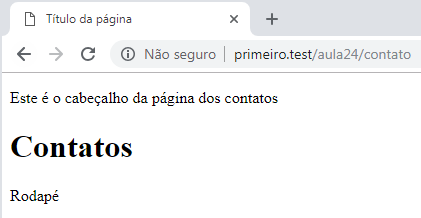
<h1>Contatos</h1>

@stop

@section('rodape')

<p>Rodapé</p>

@stop



**views/aula24/about.blade.php**

@extends('layouts.base\_layout')

@section('cabecalho')

<p>Este é o cabeçalho da página do about</p>

@stop

@section('conteudo')

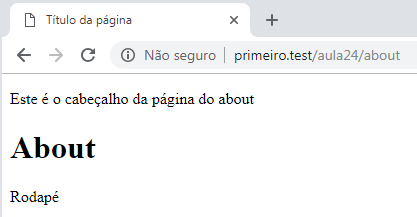
<h1>About</h1>

@stop

@section('rodape')

<p>Rodapé</p>

@stop



# Aula 025 – Blade Layouts – Parte 2

- Dentro da pasta views 🡪 aula25 🡪layouts criar o arquivo app.blade.php;

**views/aula25/layouts/app.blade.php**

<!DOCTYPE html>

<html>

<head>

<title>@yield('titulo', 'Título default')</title>

</head>

<body>

{{-- navegacao --}}

@section('navegacao')

<p>Conteúdo transversal</p>

@show

</body>

</html>

**views/aula25/usuarios.blade.php**

@extends('aula25.layouts.app')

{{-- titulo --}}

@section('titulo')

{{'Este é o título verdadeiro'}}

@stop

{{-- navegação --}}

@section('navegacao')

@parent

<p>Navegação dos usuários</p>

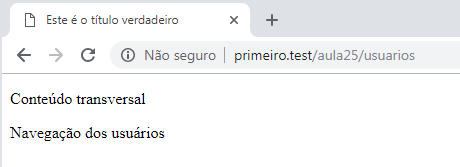
@stop

**routes/web.php**

Route::get('aula25/usuarios', function(){

return view('aula25/usuarios');

});



# Aula 026 – Blade Layouts – Parte 3

## Usando o blade para exibir mensagens de erro

titulo

texto da mensagem

titulo – fundo cinza

mensagem – texto cinza com fundo branco

**routes/web.php**

Route::get('aula26/teste', function(){

return view('aula26/teste');

});

- Na pasta view 🡪 aula26 criar dois arquivos:

teste.blade.php e comp\_mensagem.blade.php:

**views/aula26/teste.blade.php**

<div>

<p>Primeira seção da página</p>

</div>

@component('aula26.comp\_mensagem')

{{-- titulo --}}

Este é o título

@endcomponent

<div>

<p>Fundo da página</p>

</div>

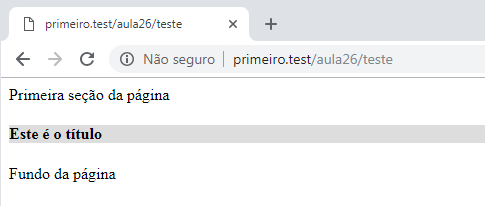
**views/aula26/comp\_mensagem.blade.php**

{{-- titulo --}}

<div style="background-color: #ddd">

<h4>{{$slot}}</h4>

</div>





**views/aula26/comp2\_mensagem.blade.php**

<div style="background-color: #ddd">

<h4>{{$titulo}}</h4>

</div>

<div style="color: #ddd">

<p>{{$mensagem}}</p>

</div>

**views/aula26/teste2.blade.php**

<div>

<p>Primeira seção da página</p>

</div>

@component('comp\_mensagem')

@slot('titulo')

Este é o título da mensagem

@endslot

@slot('mensagem')

Este é o texto que vai surgir na mensagem

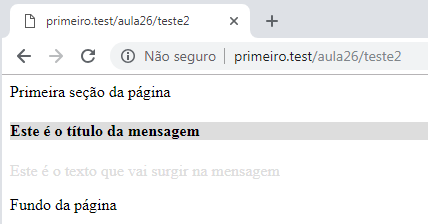
@endslot

@endcomponent

<div>

<p>Fundo da página</p>

</div>



# Aula 027 – Blade include sub-views

**routes/web.php**

Route::get('show', function(){

return view('apresentacao');

});

- Dentro da pasta views 🡪 aula27 criar dois arquivos:

apresentacao.blade.php e inc/dados.php

**views/aula27/apresentacao.blade.php**

<h1>Include</h1>

<div>Este é o texto da primeira seção</div>

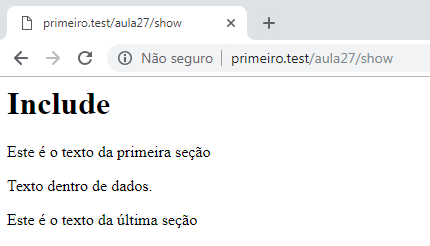
{{-- include --}}

@include('inc.dados')

<div>Este é o texto da última seção</div>

**views/aula27/dados.php**

<p>Texto dentro de dados.</p>



**routes/web.php**

Route::get('aula27/show2', function(){

return view('aula27/apresentacao2');

});

**views/aula27/apresentacao2.blade.php**

<h1>Include</h1>

<div>Este é o texto da primeira seção</div>

@php

$nomes = [

'Joao',

'Manuel',

'Maria'

];

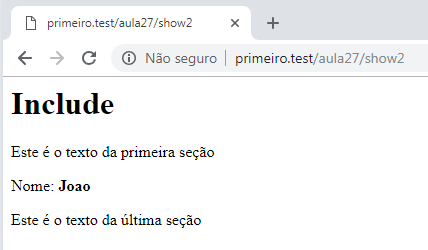
@endphp

@include('aula27.inc.linha\_dados2')

<div>Este é o texto da última seção</div>

**views/aula27/inc/linha\_dados2.blade.php**

<p>Nome: <strong> {{ $nomes[0] }} </strong></p>



**routes/web.php**

Route::get('aula27/show3', function(){

return view('aula27/apresentacao3');

});

**views/aula27/apresentacao3.blade.php**

<h1>Include</h1>

<div>Este é o texto da primeira seção</div>

@php

$nomes = [

'Joao',

'Manuel',

'Maria'

];

@endphp

{{-- ciclo de apresentação dos dados --}}

@foreach ($nomes as $nome)

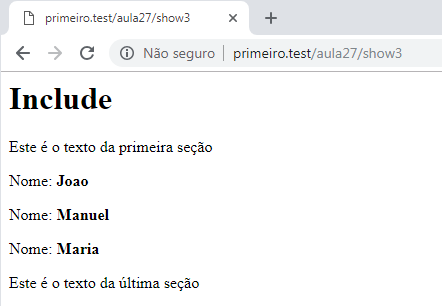
@includeIf('aula27.inc.linha\_dados3')

@endforeach

<div>Este é o texto da última seção</div>

**views/aula27/inc/linha\_dados3.blade.php**

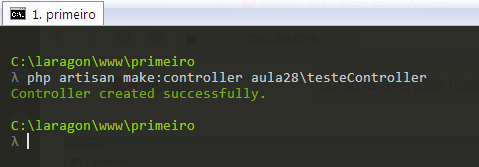
<p>Nome: <strong> {{ $nome }} </strong></p>



# Aula 028 – Blade Loops

- No terminal, entre com o seguinte comando:

**php artisan make:controller aula28\testeController**



**routes/web.php**

Route::get('aula28/teste', 'aula28\testeController@Executar');

## Loop For

**app/http/Controllers/aula28/testeController.php**

<?php

namespace App\Http\Controllers\aula28;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class testeController extends Controller

{

public function executar(){

// variáveis

// view

return view('aula28/teste');

}

}

**views/aula28/teste.blade.php**

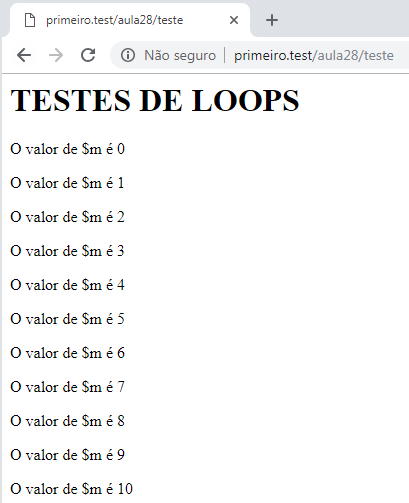
<h1>TESTES DE LOOPS</h1>

{{-- for loop --}}

@for ($m = 0; $m <= 10; $m++)

<p>O valor de $m é {{$m}}</p>

@endfor



## Foreach

**routes/web.php**

Route::get('aula28/teste2', 'aula28\testeController@Executar2');

**app/http/Controllers/aula28/testeController.php**

<?php

namespace App\Http\Controllers\aula28;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class testeController extends Controller

{

public function executar(){

// variáveis

// view

return view('aula28/teste');

}

public function executar2(){

// variáveis

$max = 10;

// nomes

$nomes = [

'Joao',

'Ana',

'Manuel',

'Francisco',

'Maria'

];

// view

return view('aula28/teste2', compact('max', 'nomes'));

}

**views/aula28/teste2.blade.php**

<h1>TESTES DE LOOPS</h1>

{{-- for loop --}}

@for ($m = 0; $m <= $max; $m++)

<p>O valor de $m é {{$m}}</p>

@endfor

{{-- foreach --}}

@foreach ($nomes as $nome)

@if ($loop->first)

<p style="color: red;">O nome é {{$nome}}</p>

@elseif($loop->last)

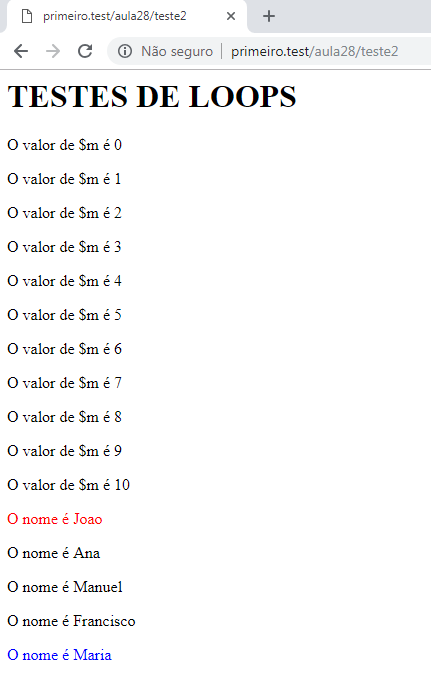
<p style="color: blue;">O nome é {{$nome}}</p>

@else

<p>O nome é {{$nome}}</p>

@endif

@endforeach



# Aula 029 – Blade Views para coleções

**routes/web.php**

Route::get('aula29/teste', 'aula29\testeController@Executar');

**app/Http/Controllers/aula29/testeController.php**

<?php

namespace App\Http\Controllers\aula29;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class testeController extends Controller

{

public function executar(){

// nomes

$nomes = [

'Joao',

'Ana',

'Manuel',

'Francisco',

'Maria'

];

// view

return view('aula29/teste', compact('nomes'));

}

}

**views/aula29/teste.blade.php**

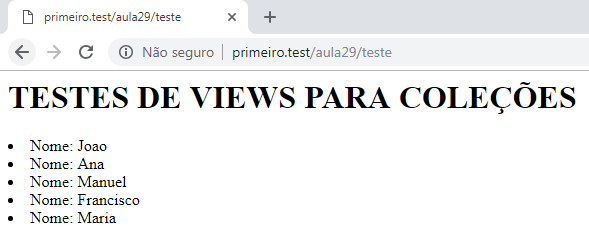
<h1>TESTES DE VIEWS PARA COLEÇÕES</h1>

@each ('aula29.linha', $nomes, 'nome', 'aula29.vazio')

- Na pasta Views criar dois arquivos: linha.blade.php e vazio.blade.php

**views/aula29/linha.blade.php**

<li>Nome: {{$nome}}</li>





**app/Http/Controllers/aula29/testeController.php**

<?php

namespace App\Http\Controllers\aula29;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class testeController extends Controller

{

public function executar(){

// nomes

// $nomes = [

// 'Joao',

// 'Ana',

// 'Manuel',

// 'Francisco',

// 'Maria'

// ];

**$nomes = [];**

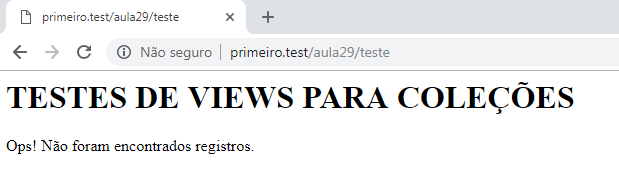
return view('teste3', compact( 'nomes'));

}

}

**views/aula29/vazio.blade.php**

<p>Ops! Não foram encontrados registros.</p>



# Aula 030 – Passando dados de formulário para Controller

**routes/web.php**

Route::get('aula30/formulario', function(){

return view('aula30/formulario');

});

- Na pasta Views🡪 aula30, crie um arquivo chamado formulario.blade.php

**views/aula30/formulario.blade.php**

<form method="POST" action=" ">

{{csrf\_field()}}

Usuário:

<input type="text" name="text\_usuario">

<br />

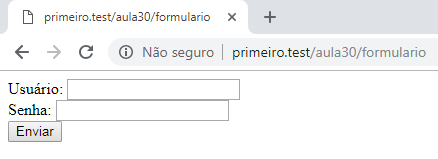
Senha:

<input type="password" name="text\_senha">

<br />

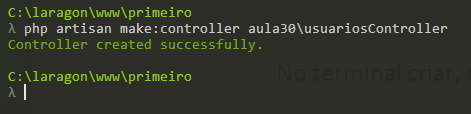
<input type="submit" value="Enviar">

</form>





No terminal criar, na pasta aula30, o controller “usuariosController”



**app/http/Controllers/aula30/usuariosController.php**

<?php

namespace App\Http\Controllers\aula30;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class usuariosController extends Controller

{

// Apresentar o formulário

public function show(){

return view('aula30/formulario');

}

// Tratar os dados do formulário

public function tratar(Request $dados){

return 'Cheguei!';

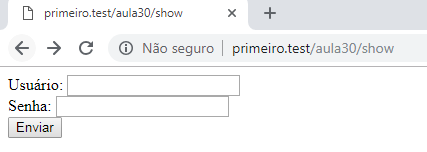
}

}

**routes/web.php**

Route::get('aula30/show', 'aula30\usuariosController@show');

Route::post('aula30/tratar', 'aula30\usuariosController@tratar');



**views/aula30/formulario.blade.php**

<form method="POST" action="tratar">

{{csrf\_field()}}

Usuário:

<input type="text" name="text\_usuario">

<br />

Senha:

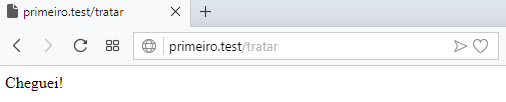
<input type="password" name="text\_senha">

<br />

<input type="submit" value="Enviar">

</form>

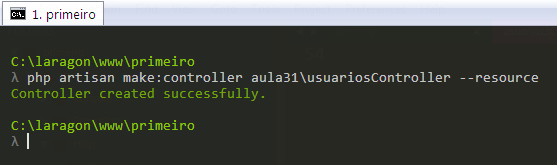
- Clique no botão “Enviar”



# Aula 031 – Resource Controller e dados de formulário

No terminal criar, na pasta aula31, o controller “usuariosController” do tipo resource:

**php artisan make:controller aula31\usuariosController --resource**



**app/Http/Controllers/aula31/usuariosController.php**

<?php

namespace App\Http\Controllers\aula31;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

class usuariosController extends Controller

{

public function index()

{

// Apresenta o formulário

return view('formulario');

}

public function create()

{

//

}

public function store(Request $request)

{

$usuario = $request->text\_usuario;

$senha = $request->text\_senha;

$token = $request->\_token;

$senha\_enc = md5($senha);

echo "Usuário: $usuario <br />Senha: $senha <br />Senha encriptada: $senha\_enc";

// return 'Cheguei';

}

public function show($id)

{

//

}

public function edit($id)

{

//

}

public function update(Request $request, $id)

{

//

}

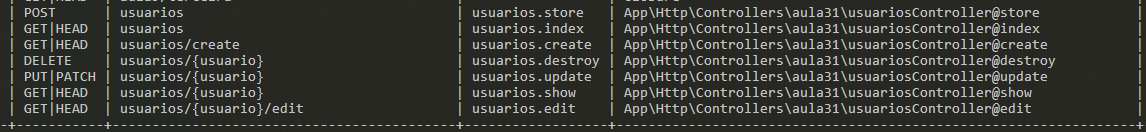
public function destroy($id)

{

//

}

}



**views/aula31/formulario.blade.php**

<form method="POST" action="usuarios">

{{csrf\_field()}}

Usuário:

<input type="text" name="text\_usuario">

<br />

Senha:

<input type="password" name="text\_senha">

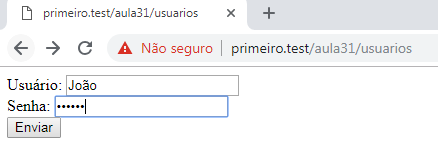
<br />

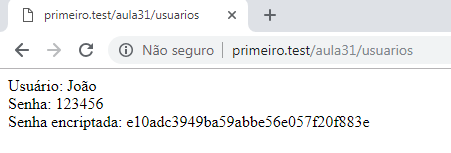
<input type="submit" value="Enviar">

</form>

**routes/web.php**

Route::resource('aula31/usuarios', 'aula31\usuariosController');





# Aula 032 – Bases de dados - Configurações

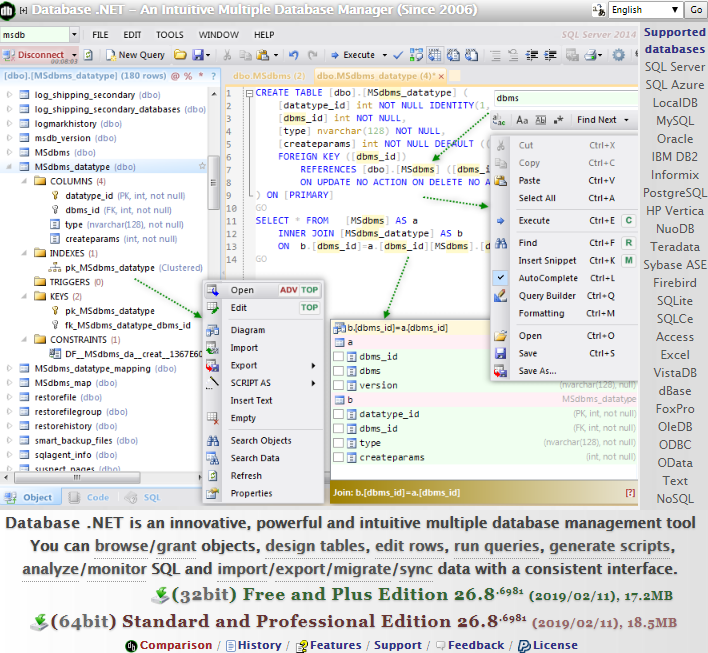
Acessar:

https://fishcodelib.com/database.htm

E baixar database .NET

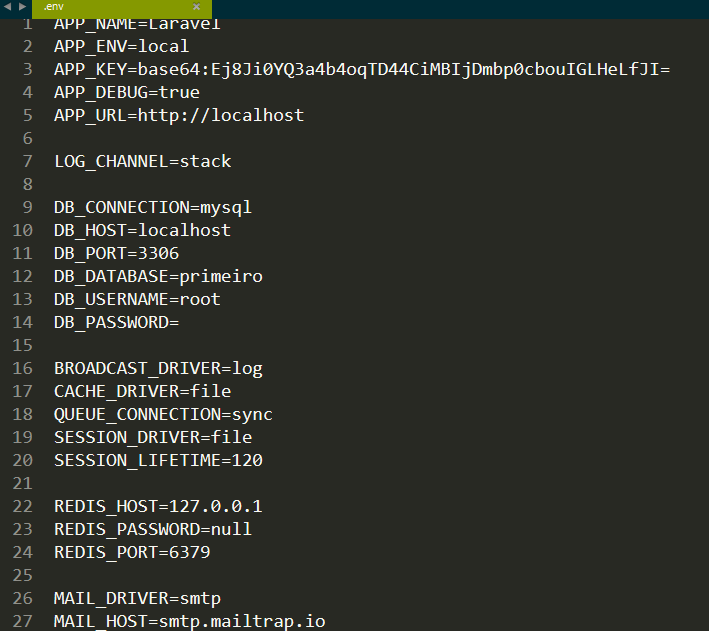
É um arquivo único. Inseri-lo em uma pasta chamada: C:\database.net e criar um atalho na área de trabalho do Windows.

Para iniciar o aplicativo basta clicar sobre o arquivo.



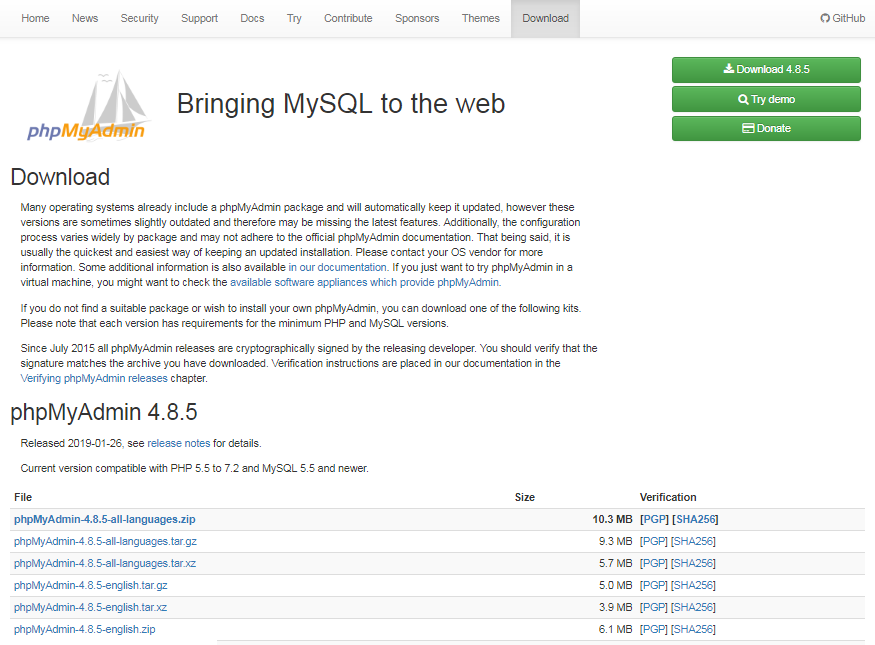
- Criar um banco de dados chamado “**primeiro**”.

- Abrir o arquivo .env e configurar a conexão com banco de dados.

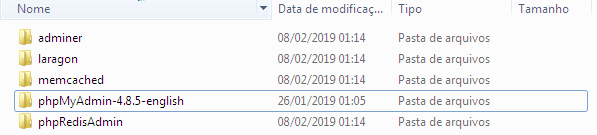


## Instalando o phpmyadmin dentro do Laragon

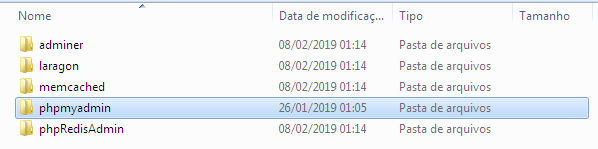
Baixar o aplicativo phpmyadmin, na versão zipada (phpMyAdmin-4.8.5-english.zip), em: https://www.phpmyadmin.net/downloads/



- Descompactar dentro da pasta C:\laragon\etc\apps



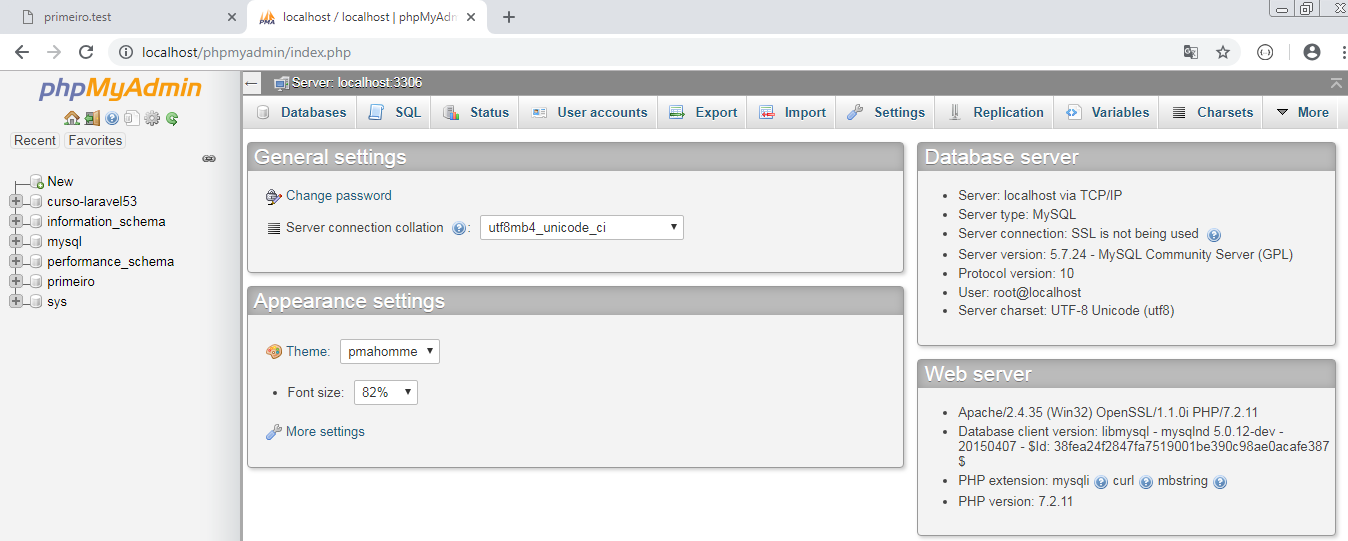
- Alterar o nome da pasta para: phpmyadmin:



No Laragon, no botão “Banco de dados” clicar em <http://localhost/phpmyadmin>:



Entrar com usuário: root e senha vazia.



- Clicar na aba “user accounts” e criar um novo usuário com todos os privilégios.

# Aula 033 – Introdução a Migrations

- Na pasta Database 🡪 Migrations o Laravel cria automaticamente dois arquivos para registro de usuários:

* 2014\_10\_12\_000000\_create\_users\_table.php
* 2014\_10\_12\_100000\_create\_password\_resets\_table.php

**Database/Migrations/2014\_10\_12\_000000\_create\_users\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateUsersTable extends Migration

{

public function up()

{

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

$table->increments('id');

$table->string('name');

$table->string('email')->unique();

$table->timestamp('email\_verified\_at')->nullable();

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('users');

}

}

A função up é usada para criar a estrutura da tabela e função down é usada para apagá-la (faz um rollback).

A função timestamps() cria os campos created\_at e updated\_at.

## Atualizando uma tabela

Para que atualizações feitas em uma tabela sejam refletidas no banco de dados, entre, no terminal, com o seguinte comando:

**php artisan migrate:refresh**

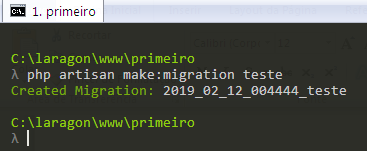
# Aula 034 – Criação de Migration

Para criar uma migration, no terminal entre com o seguinte comando:

**php artisan make:migration <nome\_migration>**

Exemplo:

**php artisan make:migration teste**



**database/migrations/2019\_02\_12\_004444\_teste.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class Teste extends Migration

{

public function up()

{

//

}

public function down()

{

//

}

}

## Criando uma migration e dando o nome para a tabela

Apague a migration criada anteriormente e crie uma nova migration entrando com o comando:

**php artisan make:migration usuarios --create=usuarios**

**database/migrations/** **2019\_02\_12\_005501\_usuarios.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class Usuarios extends Migration

{

public function up()

{

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

$table->increments('id');

$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('usuarios');

}

}

- Insira os seguintes campos na tabela:

**database/migrations/2019\_02\_12\_005501\_usuarios.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class Usuarios extends Migration

{

public function up()

{

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

$table->increments('id');

$table->string('usuario');

$table->string('senha');

$table->string('email');

$table->timestamps();

});

}

public function down()

{

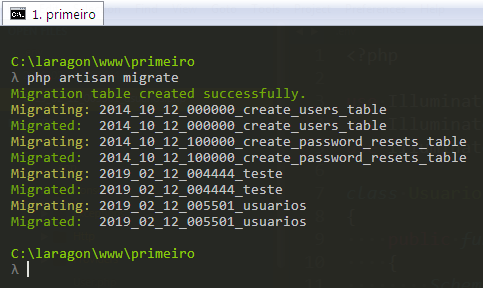
Schema::dropIfExists('usuarios');

}

}

Para criar a tabela, entre no terminal com o seguinte comando:

**php artisan migrate**



No Laravel, por convenção, os nomes das migrations devem ser criados como:

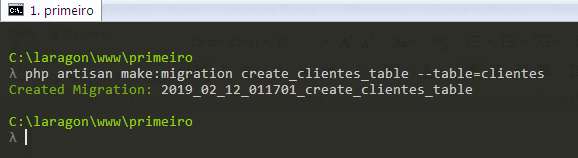
**create\_<nome\_migration>\_table**

Exemplo:

**create\_clientes\_table**

- Entre no terminal e crie a migration com o seguinte comando:

**php artisan make:migration create\_clientes\_table --table=clientes**



**database/migrations/2019\_02\_12\_011701\_create\_clientes\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateClientesTable extends Migration

{

public function up()

{

Schema::table('clientes', function (Blueprint $table) {

//

});

}

public function down()

{

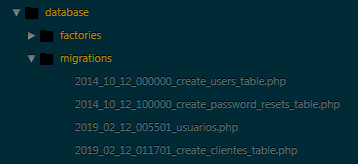
Schema::table('clientes', function (Blueprint $table) {

//

});

}

}

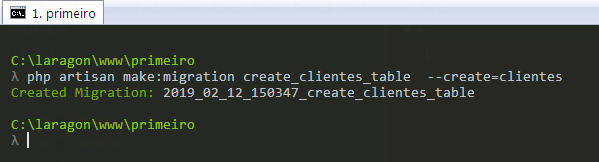


- Apague as duas tabelas criadas: de usuários e de clientes

# Aula 036 – Migrations – Criações de tabelas com Blueprint

No terminal crie uma migration com o comando:

**php artisan make:migration create\_clientes\_table --create=clientes**



**database/migrations/2019\_02\_12\_144606\_create\_testes\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateClientesTable extends Migration

{

public function up()

{

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

$table->increments('id\_cliente');

$table->string('nome\_cliente');

$table->string('endereco',50);

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

$table->boolean('ativo');

$table->dateTime('data\_nascimento');

$table->text('observacoes');

$table->timestamps();

});

}

public function down()

{

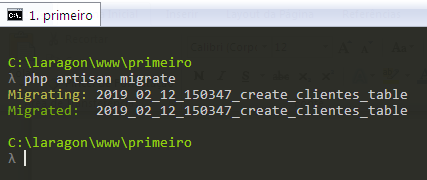
Schema::dropIfExists('clientes');

}

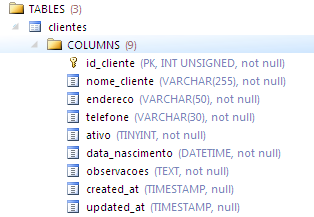
}

No terminal, crie a tabela com o seguinte comando:

**php artisan migrate**



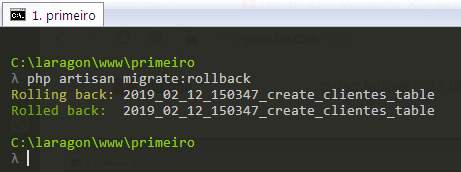




# Aula 037 – Migrations – Criação de relação entre tabelas

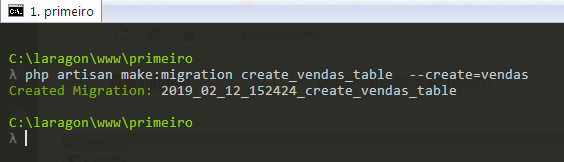
No terminal, faça um rollback da tabela criada anteriormente (tabela clientes), com o seguinte comando:

**php artisan migrate:rollback**



No terminal, crie uma migration chamada “create\_vendas\_table” com o seguinte comando:

**php artisan make:migration create\_vendas\_table --create=vendas**



**database/migrations/2019\_02\_12\_152424\_create\_vendas\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateVendasTable extends Migration

{

public function up()

{

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

$table->bigIncrements('id\_venda');

// relação

$table->integer('id\_cliente')->unsigned();

$table->string('produto');

$table->integer('quantia');

$table->timestamps();

// criar a relação

$table->foreign('id\_cliente')->references('id\_cliente')->on('clientes');

});

}

public function down()

{

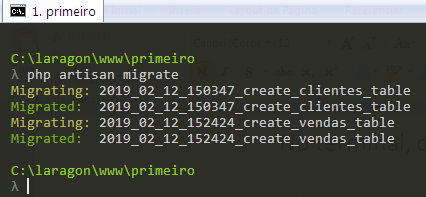
Schema::dropIfExists('vendas');

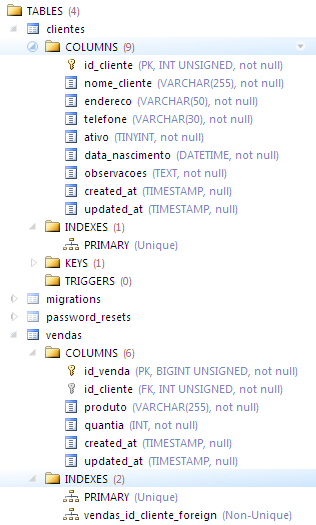
}

}

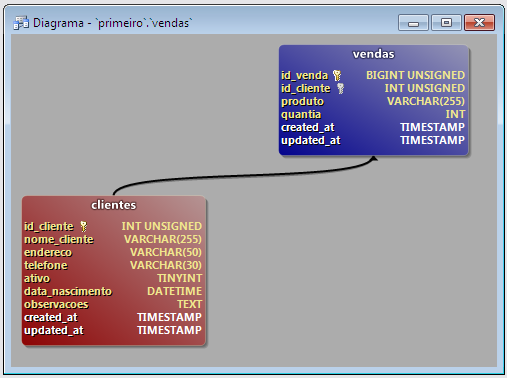
No terminal, crie as tabelas clientes e vendas com o seguinte comando:

**php artisan migrate**





Clique com o botão direito sobre a tabela “vendas” e no menu de contexto que irá se abrir, selecione a opção “diagrama”:



É uma relação de “um para muitos” (um cliente pode fazer várias vendas).

Na migration da tabela de vendas faça a seguinte alteração:

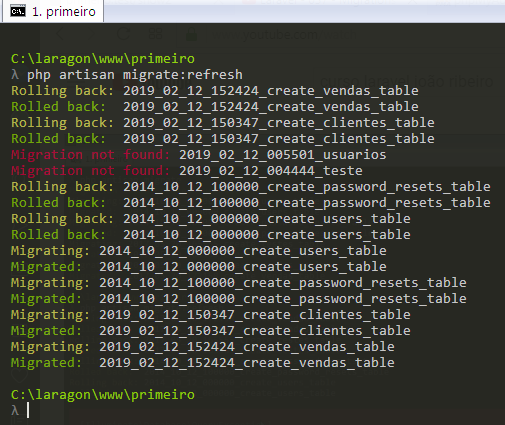
// criar a relação

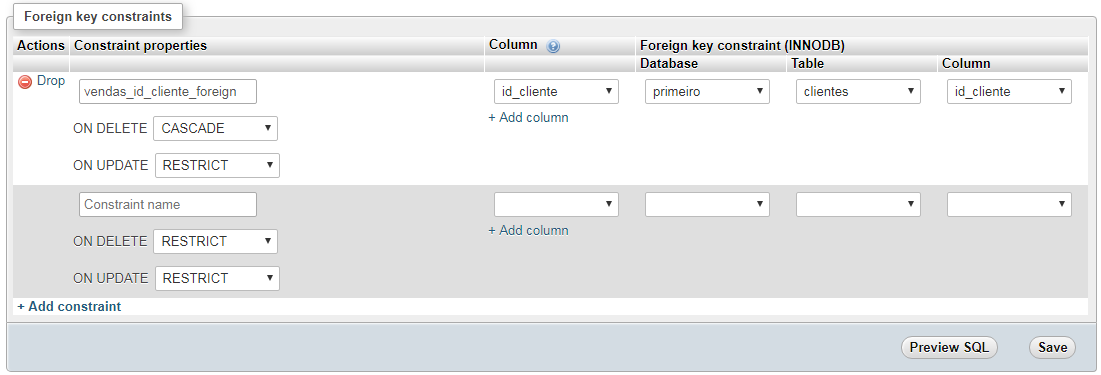
$table->foreign('id\_cliente')->references('id\_cliente')->on('clientes')->onDelete('cascade');

Dessa forma, se um cliente for eliminado, todas as suas vendas também serão.

E para atualizar a tabela, entre no terminal com o seguinte comando:

**php artisan migrate:refresh**





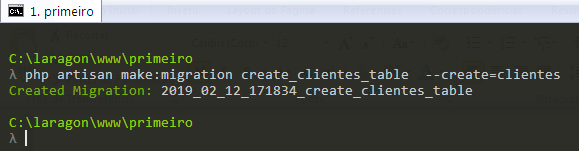
# Aula 038 – Query Builder

Apague as duas migrations criadas anteriormente (clientes e vendas).

Apague o banco de dados primeiro e o recrie vazio.

No terminal recrie a migration “create\_clientes\_table” com o comando:

**php artisan make:migration create\_clientes\_table --create=clientes**



**database/migrations/2019\_02\_12\_171834\_create\_clientes\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateClientesTable extends Migration

{

public function up()

{

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

$table->increments('id\_cliente');

$table->string('nome\_cliente');

$table->string('cidade');

});

}

public function down()

{

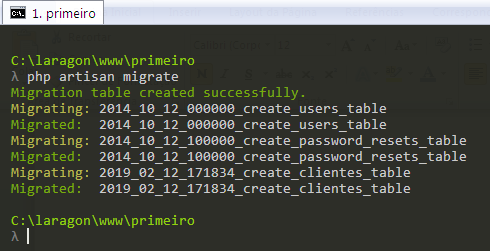
Schema::dropIfExists('clientes');

}

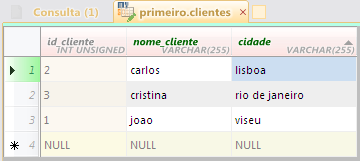
}

No terminal crie a tabela com o comando:

**php artisan migrate**



Na tabela, no banco de dados, insira manualmente os seguintes registros:



Crie a route a seguir, para testar a comunicação com a base de dados:

**routes/web.php**

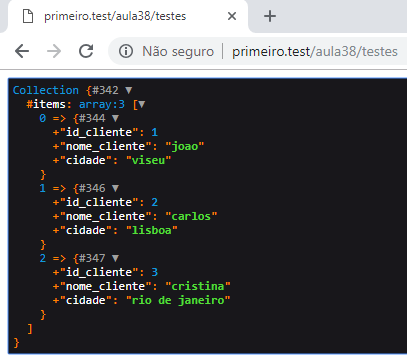
Route::get('aula38/testes', function(){

// listar todos os registros da tabela clientes

$dados = DB::table('clientes')->get();

DD($dados);

});



**routes/web.php**

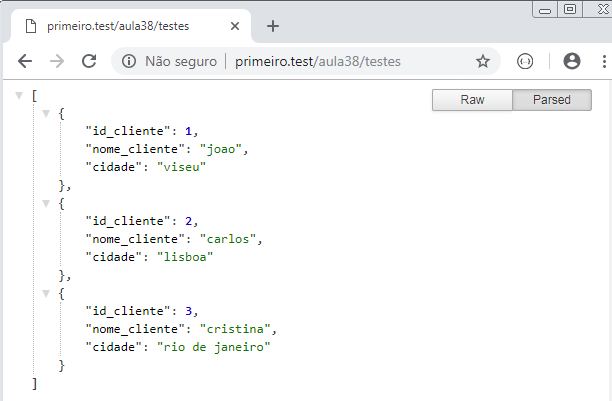
Route::get('aula38/testes', function(){

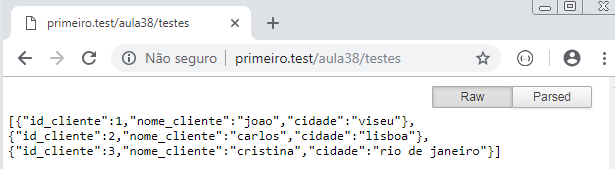
// listar todos os registros da tabela clientes

$dados = DB::table('clientes')->get();

return $dados;

});





## Listar os dados de um cliente específico

**routes/web.php**

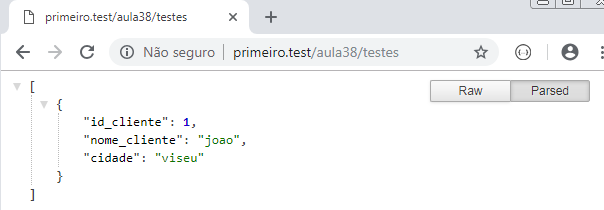
Route::get('aula38/testes', function(){

// listar os dados do cliente joao

$dados = DB::table('clientes')->where('nome\_cliente','joao')->get();

return $dados;

});



**routes/web.php**

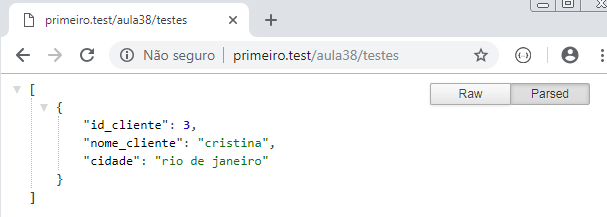
Route::get('aula38/testes', function(){

// listar os dados dos clientes do rio de janeiro

$dados = DB::table('clientes')->where('cidade','rio de janeiro')->get();

return $dados;

});

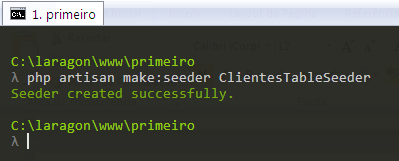


# Aula 039 – Seeding

É um mecanismo que permite popular uma base de dados.

No terminal, crie o seeder chamado ClientesTableSeeder entrando com o comando:

**php artisan make:seeder ClientesTableSeeder**



**database/seeds/ClientesTableSeeder.php**

<?php

use Illuminate\Database\Seeder;

class ClientesTableSeeder extends Seeder

{

public function run()

{

$dados = [

'nome\_cliente'=>'novo cliente',

'cidade'=>'paris'

];

DB::table('clientes')->insert($dados);

}

}

**database/seeds/ClientesTableSeeder.php**

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

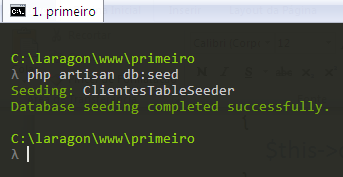
$this->call(ClientesTableSeeder::class);

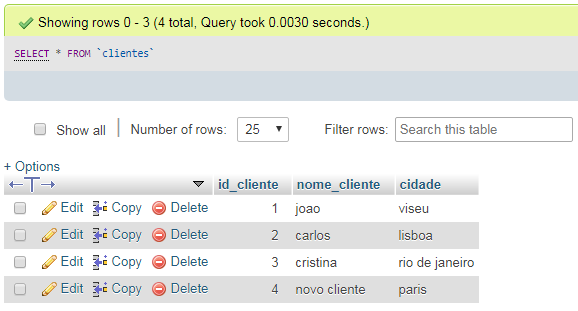
}

}

Para inserir o registro, entre no terminal com o comando:

**php artisan db:seed**





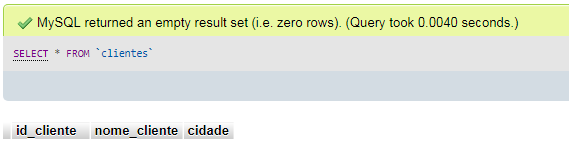
Crie a route a seguir para apagar todos os dados da tabela clientes:

Route::get('aula39/delete\_all', function(){

DB::table('clientes')->delete();

return 'Ok';

});



**database/seeds/ClientesTableSeeder.php**

<?php

use Illuminate\Database\Seeder;

class ClientesTableSeeder extends Seeder

{

public function run()

{

$nomes = [

'joao','antonio','carlos','miguel','ana','maria','suzana','cristina'

];

$cidades = [

'viseu','lisboa','lisboa','rio de janeiro','coimbra','são paulo','são paulo','rio de janeiro'

];

for($indice = 0; $indice < count($nomes) ; $indice++){

$dados = [

'nome\_cliente' => $nomes[$indice],

'cidade' => $cidades[$indice]

];

DB::table('clientes')->insert($dados);

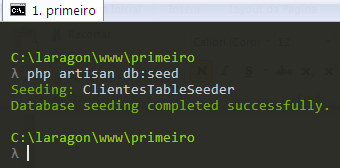
}

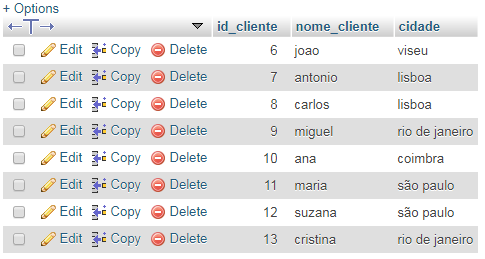
}

}

Para inserir os registros, entre no terminal com o comando:

**php artisan db:seed**





# Aula 040 – Tinker

No terminal, entre com o seguinte comando:

**php artisan tinker**

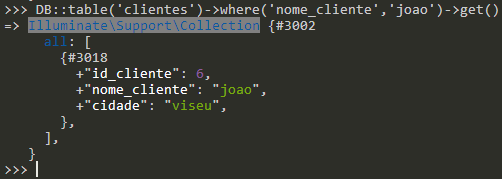
Em seguida, no prompt entre com:

**DB::table('clientes')->get()**



Entre com o seguinte comando:

**DB::table('clientes')->where('nome\_cliente', 'joao')->get()**



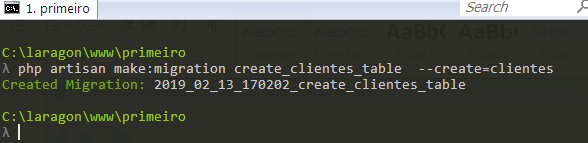
# Aula 041 – Exercício de Seeding

Apague a migration create\_clientes\_table.

Apague o banco de dados primeiro e o recrie vazio.

No terminal recrie a migration “create\_clientes\_table” com o comando:

**php artisan make:migration create\_clientes\_table --create=clientes**



**database/migrations/2019\_02\_13\_170202\_create\_clientes\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateClientesTable extends Migration

{

public function up()

{

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

$table->increments('id\_cliente');

$table->string('nome');

$table->string('sobrenome');

$table->string('cidade');

$table->string('email');

$table->date('data\_nascimento');

});

}

public function down()

{

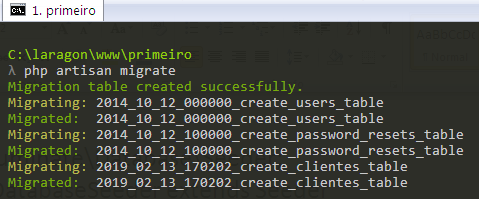
Schema::dropIfExists('clientes');

}

}

Para criar a tabela:

**php artisan migrate**



**database/seeds/ClientesTableSeeder.php**

<?php

use Illuminate\Database\Seeder;

class ClientesTableSeeder extends Seeder

{

public function run()

{

// Número total de clientes que pretende inserir na base de dados

$numero\_total\_clientes = 500;

// Define fonte de informações para seeds

$nomes = ['Joao','Ana','Antonio','Marilia','Gabriela','Carlos','Francisco','Marta','Suzana','Alice','Pedro','Rui','Manuel','Gilberto','Mario','Guilherme','Rubens','Mateus','Luis','Miguel','Elias','Mariana','Diego','Alberto','Alexandre','Rita','Alfredo','Ademir','Augusto','Bernardo','Bruno','Cesar','Rodrigo','Reinaldo','Afonso','Marcia','Simone','Diogenes','Fernando','Isabel','Roberto','Ricardo','Tulio','Marcelo','Nair','Nadia','Katia','Renata','Geraldo','Irene','Adalberto','Denise','Jose','Jair','Eduardo','Heitor','Alex','Sandra','Erica','Karina','Marlene','Josias','Luana','Tereza','Monica','Marina','Moacir','Paulo','Ramon','Sergio'];

$sobrenomes=['Rodrigues','Santos','Oliveira','Martins','Costa','Teixeira','Almeida','Ribeiro','Pereira','Simoes','Damiao','Souza','Alves','Valentino','Guimaraes','Pacheco','Cardoso','Moreira','Silva','Alencar', 'Figueiredo', 'Carvalho','Gutierrez','Batista','Pinheiro','Bueno','Andrade','Cruz','Aguiar','Malta','Albuquerque','Vilela','Domingues','Camargo','Lima','Ferraz','Vaz','Rufino','Dias','Barroso','Seixas','Junqueira','Alcantara','Garcia','Matos','Costa','Vasconcelos','Pinto','Moreira','Ferreira'];

$cidades=['Lisboa','Porto','Coimbra','Viseu','Faro','Braga','Aveiro','Fortaleza','Sao Paulo','Santos','Belem','Brasilia','Paris','Manaus','Roma','Curitiba','Salvador','Vitoria','Cuiaba','Guarulhos','Osasco','Goiania','Londrina','Limeira','Campinas','Taubate','Niteroi','Jundiai','Barretos','Franca','Colatina','Linhares','Teresina','Aracaju','Natal','Palmas','Arizona','Texas','Setubal','Madrid'];

$dominios\_de\_email=['gmail.com','hotmail.com','uol.com.br','oesp.com.br','redecard.com.br','vesper.com.br','kepler.com.br','yahoo.com.br','ig.com.br','bol.com.br','terra.com.br','venturini.com.br','globo.com','aol.com','artplan.com.br','click21.com.br','uninet.com.br','barralink.com.br','avipal.com.br','edunet.com.br','brturbo.com','ibest.com.br','msn.com','digiplus.com.br','sercomtel.com.br','twister.com.br','engeplus.com.br','zipmail.com.br','fasterair.com.br','maxiplast.com'];

// ciclo para inserção de clientes no BD

for($i = 0; $i < $numero\_total\_clientes; $i++){

$nome = $nomes[array\_rand($nomes)];

$sobrenome = $sobrenomes[array\_rand($sobrenomes)];

$cidade = $cidades[array\_rand($cidades)];

// data de nascimento

$ano = rand(1960, 2000);

$mes = rand(1,12);

$dia = rand(1,28);

$data\_nascimento = $ano . '-' . $mes . '-' . $dia;

// cria o endereço de email do cliente

$email = strtolower($nome) . '.' . strtolower($sobrenome) . '@' . $dominios\_de\_email[array\_rand($dominios\_de\_email)];

$dados = [

'nome' => $nome,

'sobrenome' => $sobrenome,

'cidade' => $cidade,

'email' => $email,

'data\_nascimento' => $data\_nascimento

];

DB::table('clientes')->insert($dados);

}

}

}

**database/seeds/DatabaseSeeder.php**

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

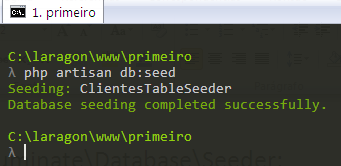
$this->call(ClientesTableSeeder::class);

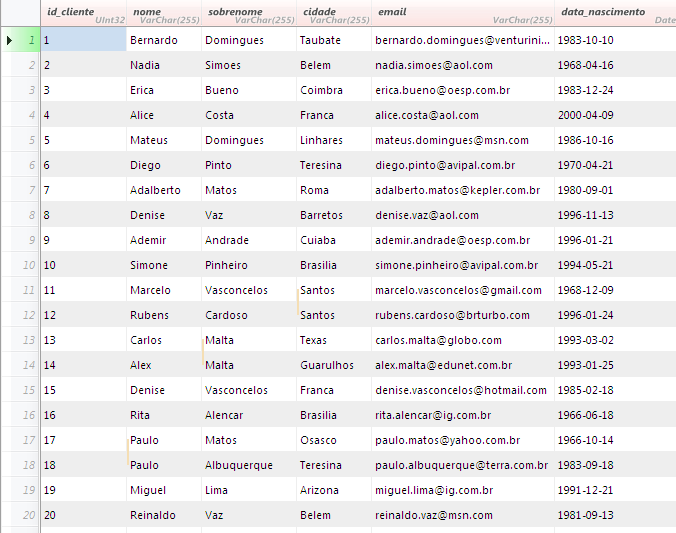
}

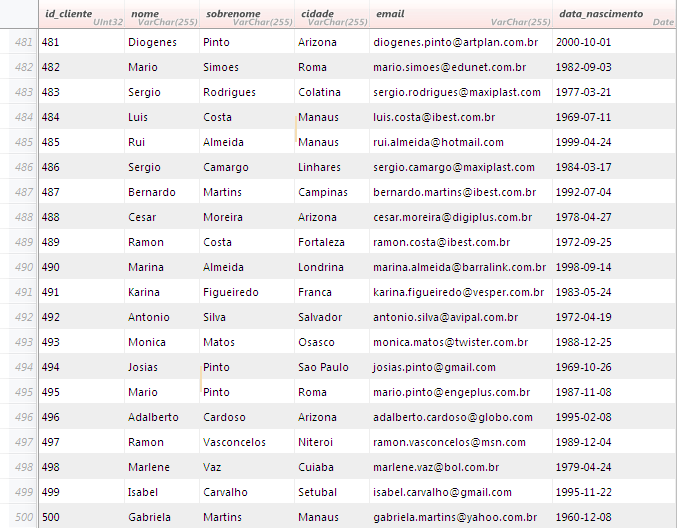
}

Para inserir os dados na tabela, no terminal, entre com o comando:

**php artisan db:seed**



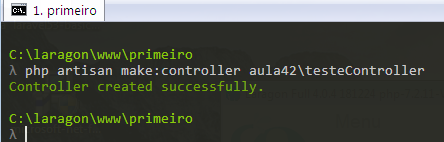




# Aula 042 – Query Builder - Exercícios

No terminal, crie um controlador chamado "testeController" em uma pasta chamada "aula42", entrando com o seguinte comando:

php artisan make:controller aula42\testeController



O Query Builder usa a classe DB.

**app/Http/Controllers/aula42/testeController.php**

<?php

namespace App\Http\Controllers\aula42;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

use Illuminate\Support\Facades\DB;

class testeController extends Controller

{

public function teste(){

$dados = DB::table('clientes')->get();

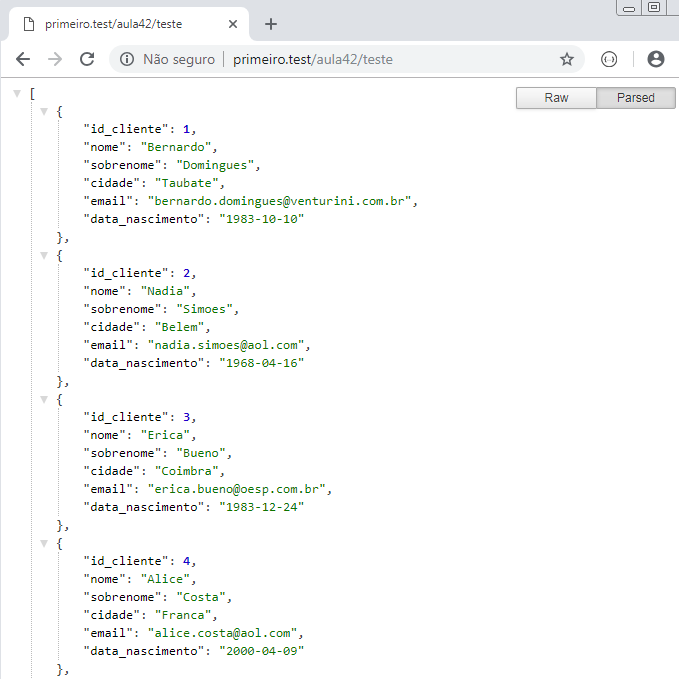
return $dados;

}

}

**routes/web**

Route::get('aula42/teste', 'aula42\testeController@teste');





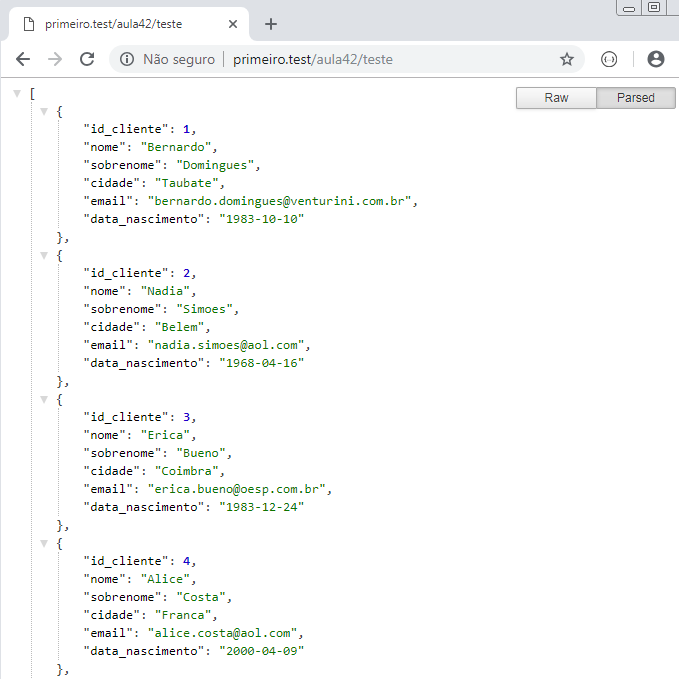
**app/Http/Controllers/aula42/testeController.php**

public function teste(){

$dados = DB::select('select \* from clientes');

return $dados;

}



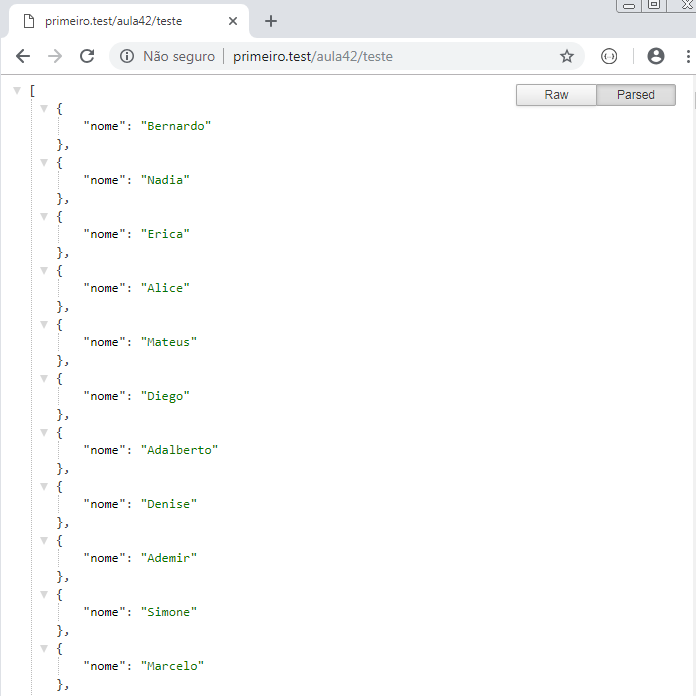
**app/Http/Controllers/aula42/testeController.php**

public function teste(){

$dados = DB::table('clientes')->select('nome')->get();

return $dados;

}



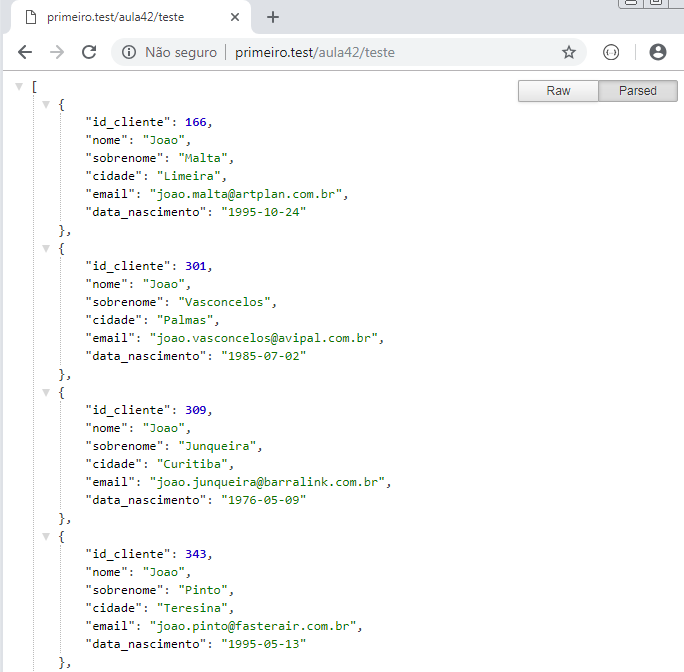
**app/Http/Controllers/aula42/testeController.php**

public function teste(){

$dados = DB::table('clientes')->where('nome','Joao')->get();

return $dados;

}





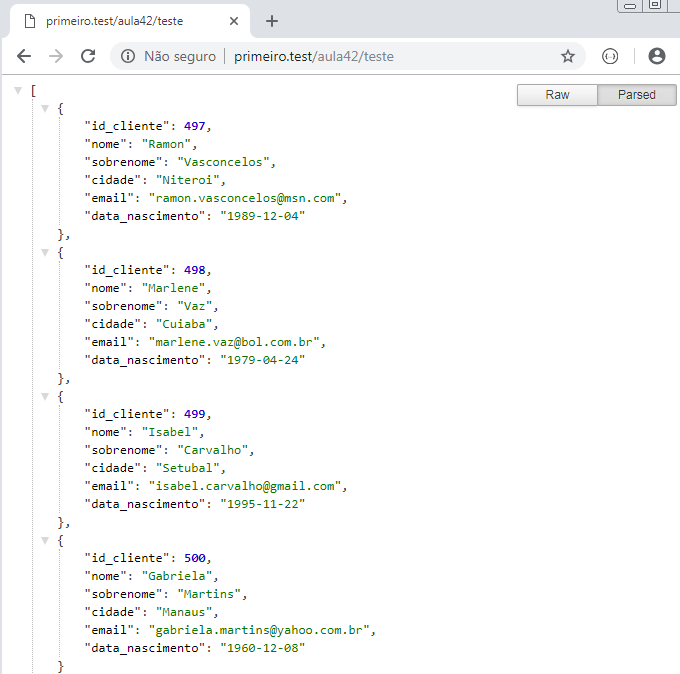
**app/Http/Controllers/aula42/testeController.php**

public function teste(){

$dados = DB::table('clientes')->where('id\_cliente','>',496)->get();

return $dados;

}



**app/Http/Controllers/aula42/testeController.php**

public function teste(){

$id = 123;

$dados = DB::table('clientes')

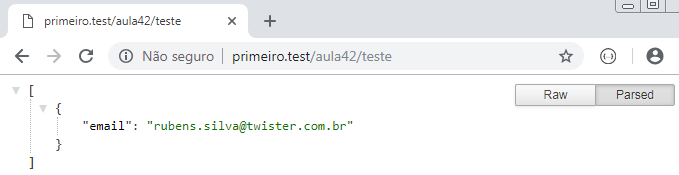
->where('id\_cliente',$id)

->select('email')

->get();

return $dados;

}



**app/Http/Controllers/aula42/testeController.php**

public function teste(){

$dados = DB::table('clientes')

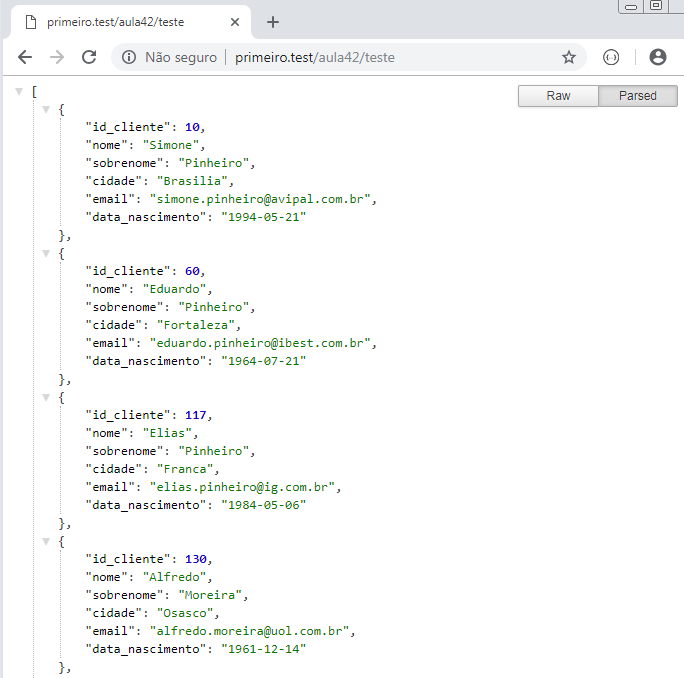
->whereYear('data\_nascimento','=',1961)

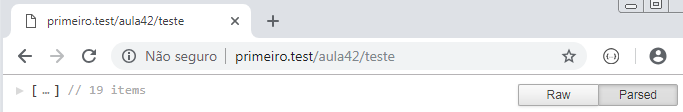
->orWhere('sobrenome','Pinheiro')

->get();

return $dados;

}



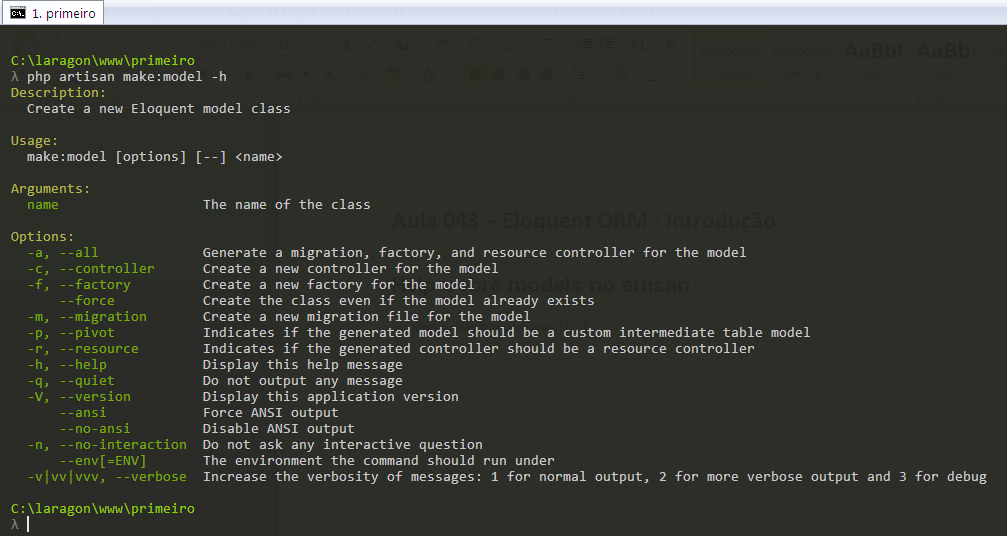


# Aula 043 – Eloquent ORM - Introdução

## Help sobre models no artisan

No terminal, entre com o comando:

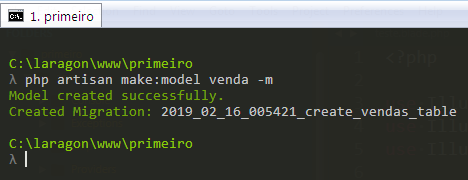
**php artisan make:model -h**



## Criando uma model e uma migration simultaneamente

No terminal, entre com o comando:

**php artisan make:model venda -m**



## Model

**app/venda.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class venda extends Model

{

//

}

## Migration

**Http/models/migrations/2019\_02\_16\_005421\_create\_vendas\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateVendasTable extends Migration

{

public function up()

{

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

$table->increments('id');

$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('vendas');

}

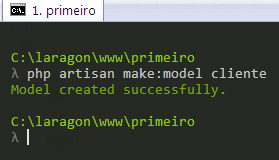
}

O Model é criado no singular, porém a tabela será criada no plural: vendas (snake case).

# Aulas 044 – Eloquent - Relação entre Model e tabela

No terminal, entre com o comando:

**php artisan make:model cliente**



**routes/web.php**

Route::get('aula44/clientes', function(){

// Query Builder

$dados = DB::table('clientes')->get();

return $dados;

});



**routes/web.php**

use App\cliente;

Route::get('aula44/clientes', function(){

// Eloquent ORM

$dados = cliente::get();

return $dados;

});



**routes/web.php**

Route::get('aula44/clientes', function(){

// Eloquent ORM

$dados = cliente::where('nome','Joao')->get();

return $dados;

});



**app/teste.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class teste extends Model

{

protected $table = 'clientes';

}

**routes/web.php**

use App\teste;

Route::get('aula44/clientes', function(){

$dados = teste::where('nome','Joao')->get();

return $dados;

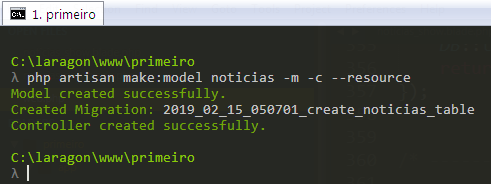
});



# Aulas 045, 046 e 047 – Colocando em prática o MVC

No terminal, crie simultaneamente um model chamado "noticias", uma migration (chamada create\_noticias\_table) e um controller chamado "noticias", entrando com o seguinte comando:

**php artisan make:model noticias -m -c --resource**



**database/migrations/2019\_02\_15\_050701\_create\_noticias\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateNoticiasTable extends Migration

{

public function up()

{

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

$table->increments('id\_noticia');

$table->string('titulo');

$table->text('texto');

$table->string('autor');

$table->timestamps();

});

}

public function down()

{

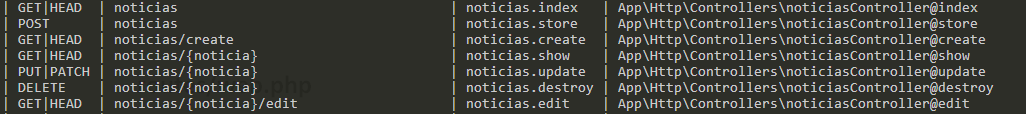
Schema::dropIfExists('noticias');

}

}

## Rotas criadas

**php route:list**



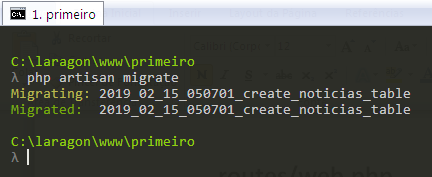
**routes/web.php**

Route::resource('noticias', 'noticiasController');

## Criando a tabela

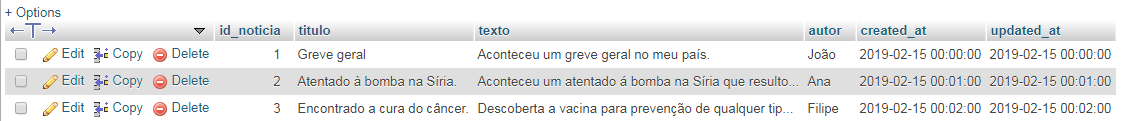
Para criar, entre no terminal com o seguinte comando:

**php artisan migrate**



## Inserindo dados na tabela

Insira manualmente na tabela três registros:



**app/Http/NoticiasController.php**

<?php

namespace App\Http\Controllers;

use App\noticias;

use Illuminate\Http\Request;

class NoticiasController extends Controller

{

public function index()

{

$dados = noticias::all();

return $dados;

}

public function create()

{

//

}

public function store(Request $request)

{

//

}

public function show($id)

{

$dados = noticias::where('id\_noticia', $id)->get();

return $dados;

}

public function edit(noticias $noticias)

{

//

}

public function update(Request $request, noticias $noticias)

{

//

}

public function destroy(noticias $noticias)

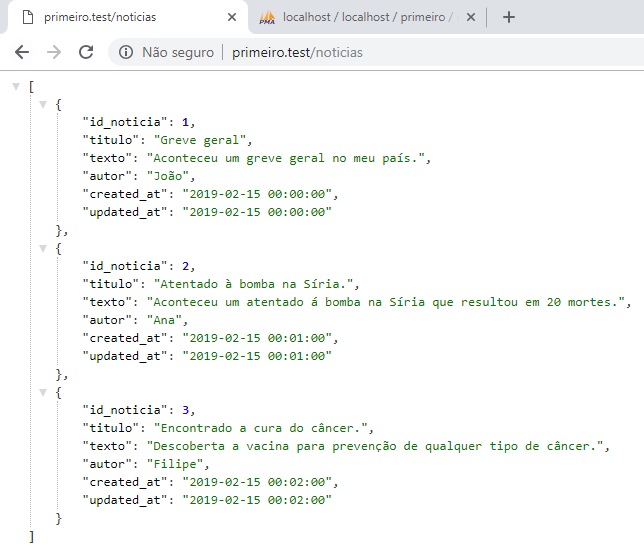
{

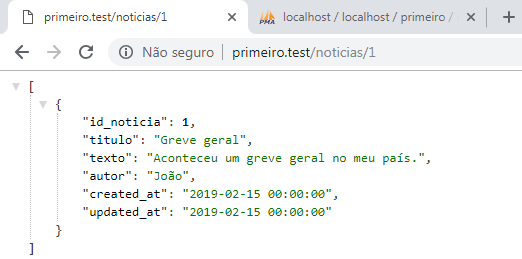
//

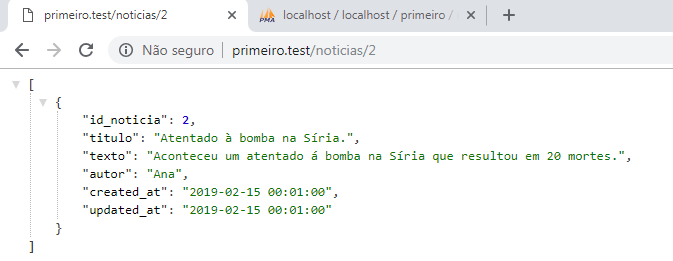
}

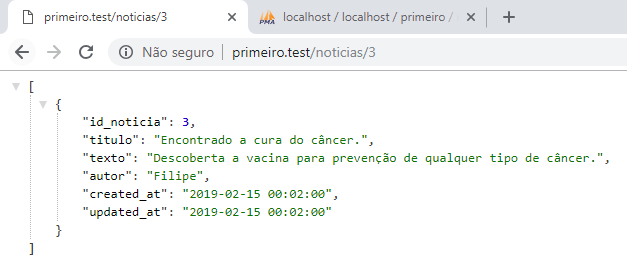
}

## Visualizando as notícias









## Criar novo registro a partir de formulário

**app/Http/NoticiasController.php**

<?php

namespace App\Http\Controllers\aula45;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

use App\noticias;

class noticiasController extends Controller

{

public function index()

{

$dados = noticias::all();

return view('aula47/noticias\_show', compact('dados'));

}

public function create()

{

return view('aula47/noticias\_new');

}

public function store(Request $request)

{

$dados = [

$request->text\_titulo,

$request->text\_texto,

$request->text\_autor

];

return $dados;

}

public function show($id)

{

$dados = noticias::where('id\_noticia', $id)->get();

return $dados;

}

public function edit($id)

{

//

}

public function update(Request $request, $id)

{

//

}

public function destroy($id)

{

//

}

}

**resources/views/aula47/noticias\_show.blade.php**

<ul>

@foreach ($dados as $noticia)

<li>{{$noticia->titulo}} | {{$noticia->texto}} | {{$noticia->autor}}</li>

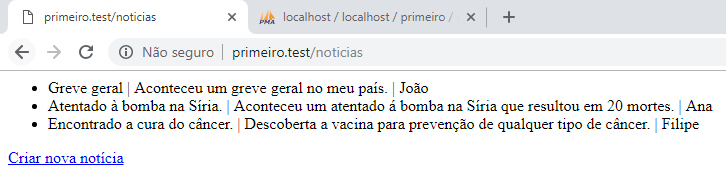
@endforeach

</ul>

<div>

<a href="noticias/create">Criar nova notícia</a>

</div>



**resources/views/aula47/noticias\_new.blade.php**

<form method="POST" action="/noticias">

{{csrf\_field()}}

<label>Título: </label>

<input type="text" name="text\_titulo">

<br>

<label>Texto: </label>

<input type="text" name="text\_texto">

<br>

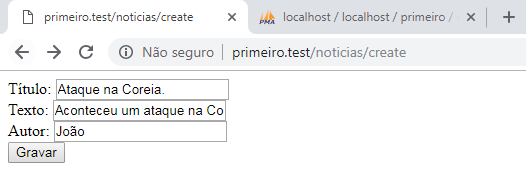
<label>Autor: </label>

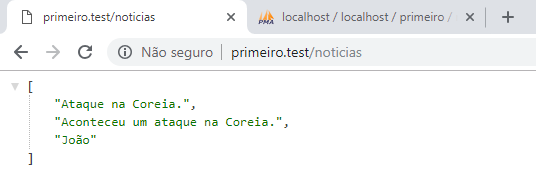
<input type="text" name="text\_autor">

<br>

<input type="submit" value="Gravar">

</form>





## Gravando o registro no banco de dados

**app/Http/NoticiasController.php**

public function store(Request $request)

{

// $dados = [

// $request->text\_titulo,

// $request->text\_texto,

// $request->text\_autor

// ];

// return $dados;

$noticia = new noticias;

$noticia->titulo = $request->text\_titulo;

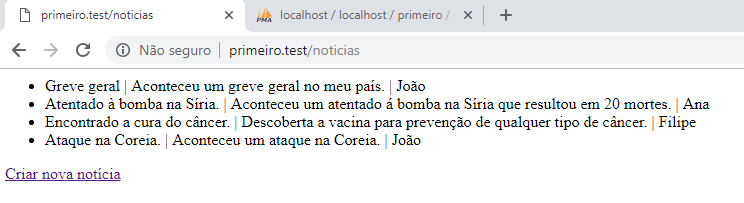
$noticia->texto = $request->text\_texto;

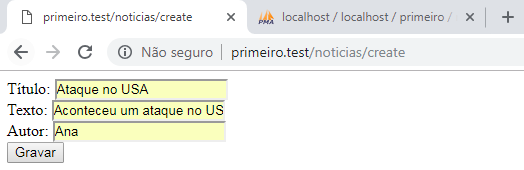
$noticia->autor = $request->text\_autor;

$noticia->save();

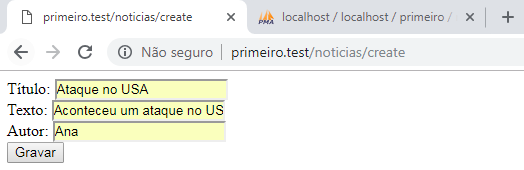
return redirect('/noticias');

}

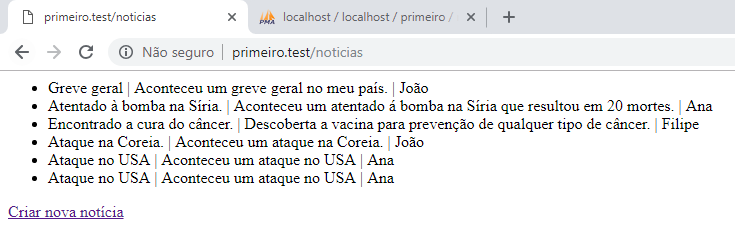




Clique no botão "Gravar"



Clique no botão "Gravar"



# Aula 048 – Adaptar o Model para apresentar registros

**app/Http/NoticiasController.php**

public function show($id)

{

// $dados = noticias::where('id\_noticia', $id)->get();

$dados = noticias::find($id);

return $dados;

}

**app/noticias.php (model)**

<?php

namespace App;

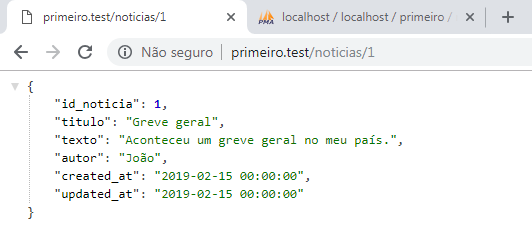
use Illuminate\Database\Eloquent\Model;

class noticias extends Model

{

protected $primaryKey = 'id\_noticia';

}



# Aula 049 – Eloquent - Editar e eliminar registros

**routes/web**

Route::get('noticias/edit/{id}', 'NoticiasController@edit');

Route::get('noticias/update/{id}', 'NoticiasController@update');

Route::get('noticias/delete/{id}', 'NoticiasController@destroy');

**app/Http/NoticiasController.php**

public function edit($id)

{

return noticias::find($id);

// return "Editar {$id}";

}

public function update(Request $request, $id)

{

$noticia = noticias::find($id);

$noticia->titulo = "Não aconteceu um ataque";

$noticia->save();

// return "Update {$id}";

}

public function destroy($id)

{

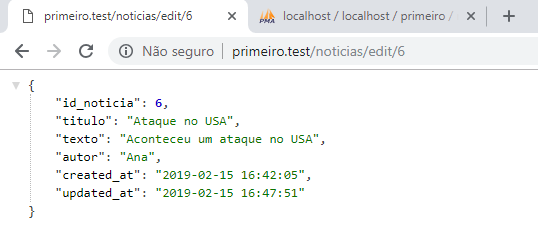
$noticia = noticias::find($id);

$noticia->delete();

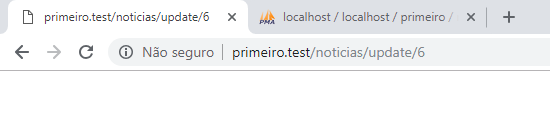
// return "Apagar {$id}";

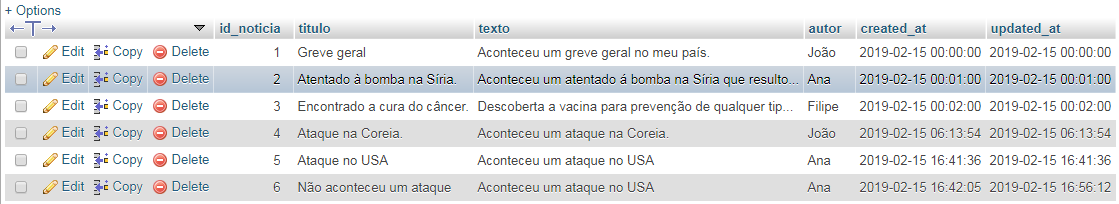
}

## Editando registro

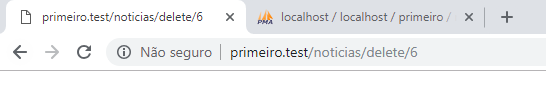


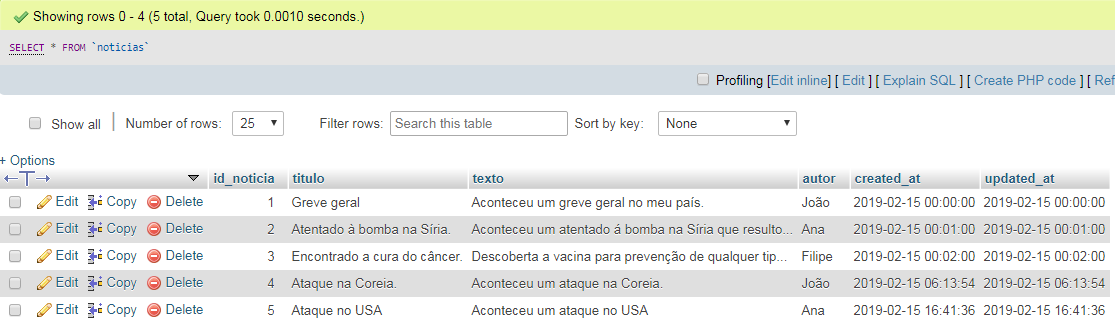
## Atualizando registro





## Deletando registro





# Aula 050 – Utilização do bootstrap no Laravel

**routes/web.php**

Route::get('aula50/teste', function(){

return view('aula50/teste');

});

**resources/views/aula50/teste.blade.php**

@extends('layouts.app');

@section('conteudo')

<h1>Título</h1>

<p>Este é o texto de um parágrafo</p>

<button class="btn btn-primary" type="submit">Button</button>

@endsection

@section('conteudo')

@endsection

@section('conteudo')

@endsection

Baixar bootstrap e:

* copiar para pasta public/css o arquivo bootstrap.min.css
* copiar para pasta public/js o arquivo bootstrap.min.js
* copiar para pasta public a pasta fonts

**resources/views/layouts/app.blade.php**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="utf-8">

<meta http-equiv="X-UA-Compatible" content="IE=edge">

<meta name="viewport" content="width=device-width, initial-scale=1">

<!-- As 3 meta tags acima \*devem\* vir em primeiro lugar dentro do `head`; qualquer outro conteúdo deve vir \*após\* essas tags -->

<title>Teste</title>

<!-- Bootstrap -->

<link href="/css/bootstrap.min.css" rel="stylesheet">

<!-- <link href="/css/app.css" rel="stylesheet"> -->

<!-- HTML5 shim e Respond.js para suporte no IE8 de elementos HTML5 e media queries -->

<!-- ALERTA: Respond.js não funciona se você visualizar uma página file:// -->

<!--[if lt IE 9]>

<script src="https://oss.maxcdn.com/html5shiv/3.7.3/html5shiv.min.js"></script>

<script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>

<![endif]-->

</head>

<body>

<div class="container">

@yield('navegacao')

@yield('conteudo')

@yield('rodape')

</div>

<!-- jQuery (obrigatório para plugins JavaScript do Bootstrap) -->

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>

<!-- Inclui todos os plugins compilados (abaixo), ou inclua arquivos separadados se necessário -->

<script src="js/bootstrap.min.js"></script>

<!-- <script src="/js/app.js"></script> -->

</body>

</html>

