**Curso Laravel 6 – especializati**

**Ministrado por Carlos Ferreira**

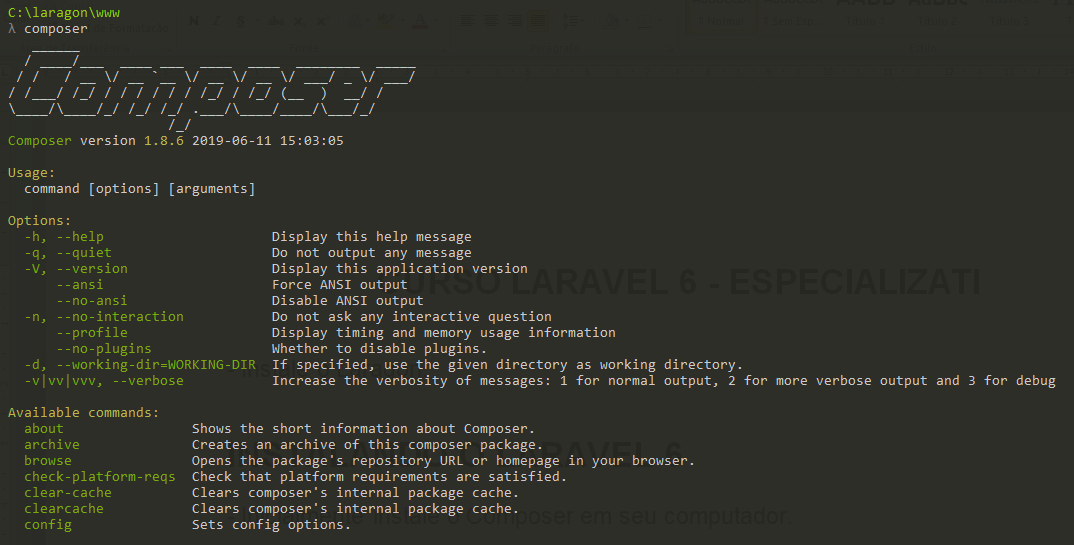
<https://www.youtube.com/watch?v=fCR6ogiPopU&list=PLVSNL1PHDWvQBtcH_4VR82Dg-aFiVOZBY>

Resumo do curso feito por Roberto Pinheiro

- Instale o Laragon e use o seu terminal para entrada de comandos.

# Aula 6 – Instalando o laravel

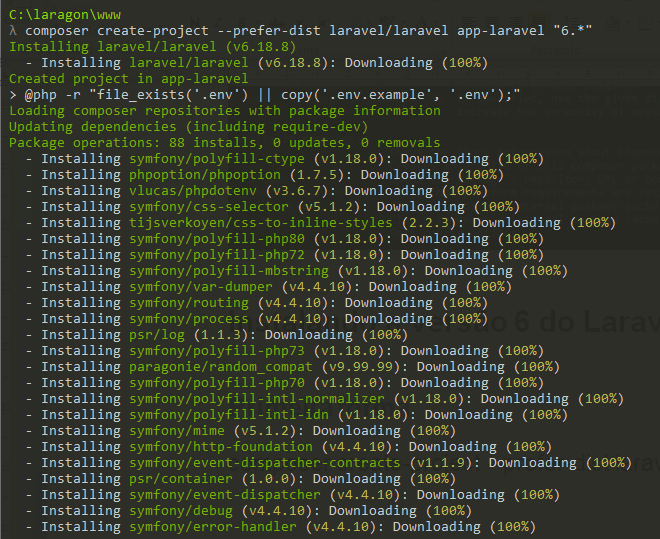
- Inicialmente instale o Composer em seu computador.



## Instalando a versão 6 do Laravel

### Primeira forma

composer create-project --prefer-dist laravel/laravel app-laravel "6.\*"



## Segunda forma

composer global require laravel/installer

laravel new app-laravel

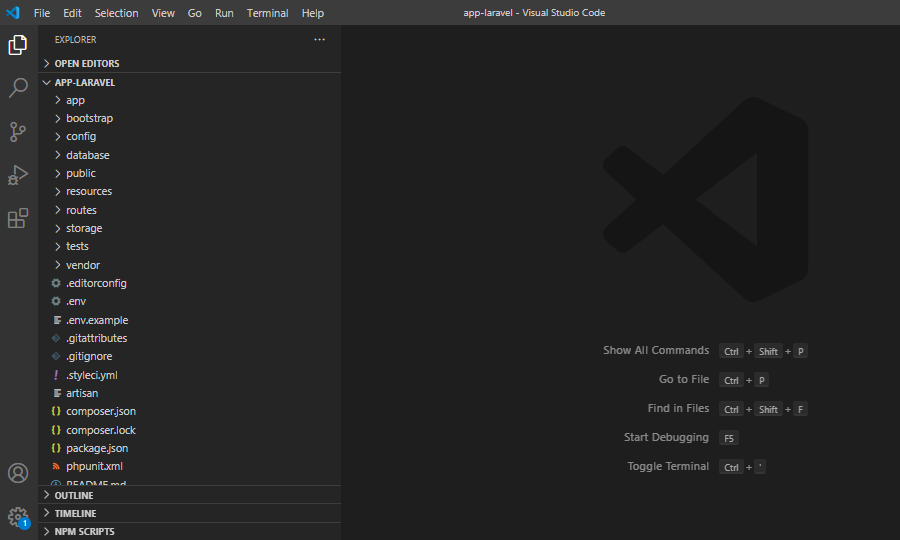
- Em seguida, vá para a página do projeto:

cd app-laravel

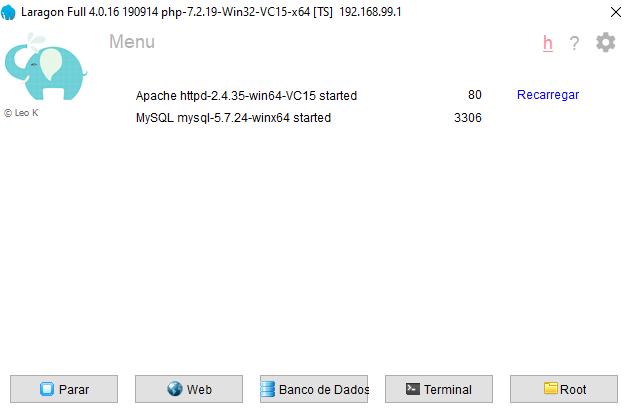
- Abra o Visual Studio Code na página do projeto:

code .



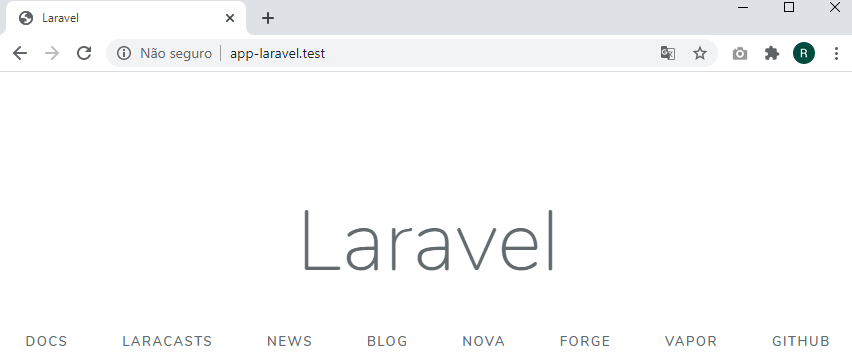


- Acesse o servidor Apache no Laragon.

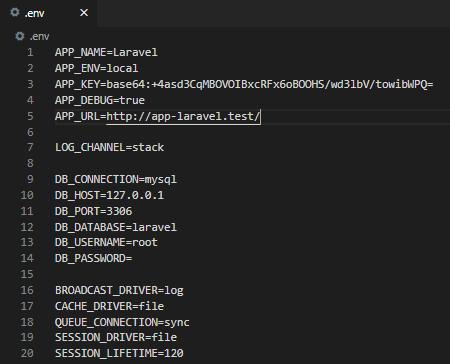


- No navegador entre com a URL:

http://app-laravel.test/

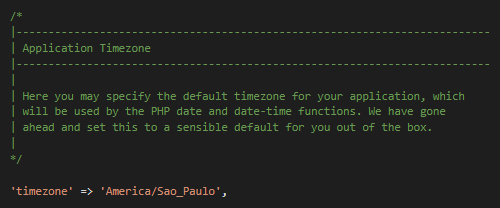


- Alterar o arquivo .env



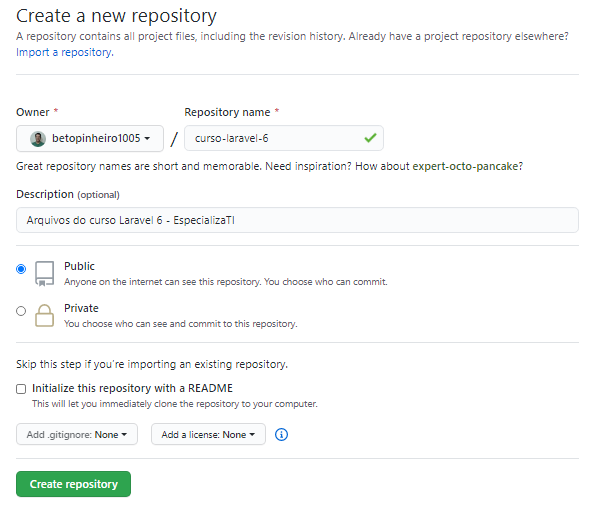
- Abrir o arquivo config/app.php

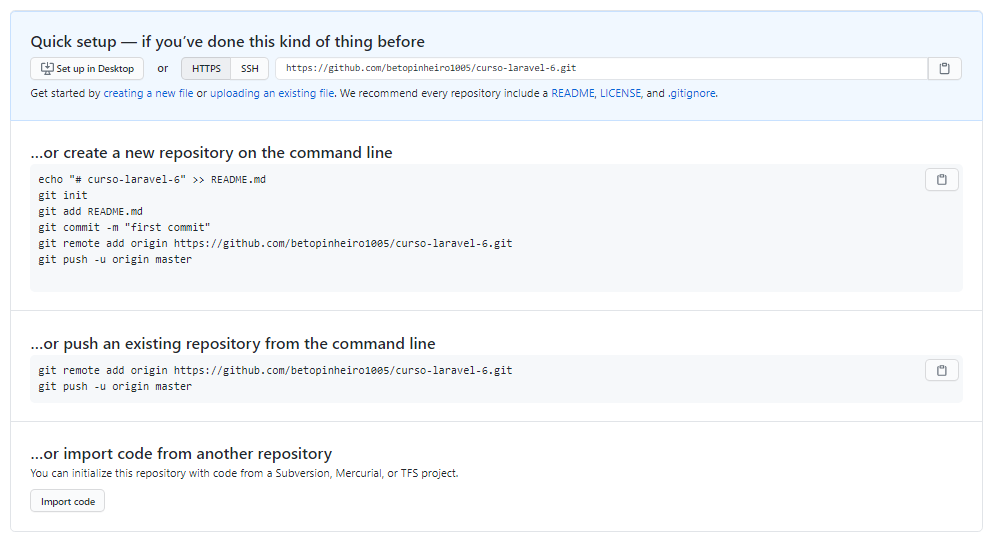
- Alterar o timezone de UTC para “America/Sao\_Paulo”:



# Aula 08 – Versionando Laravel com GIT e Armazenar no GitHub

- Crie um novo repositório no GitHub chamado curso-laravel-6:



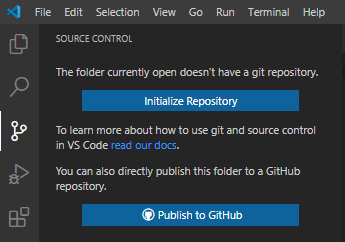


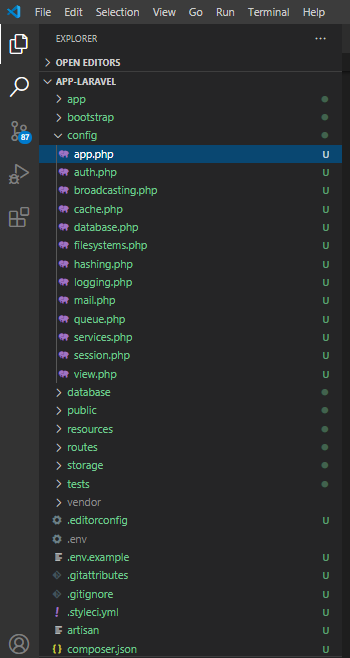
No terminal, entre com o comando:

git init



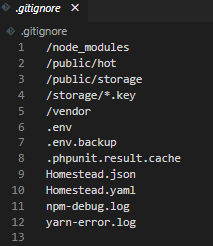
Ou inicialize o repositório no VS Code:



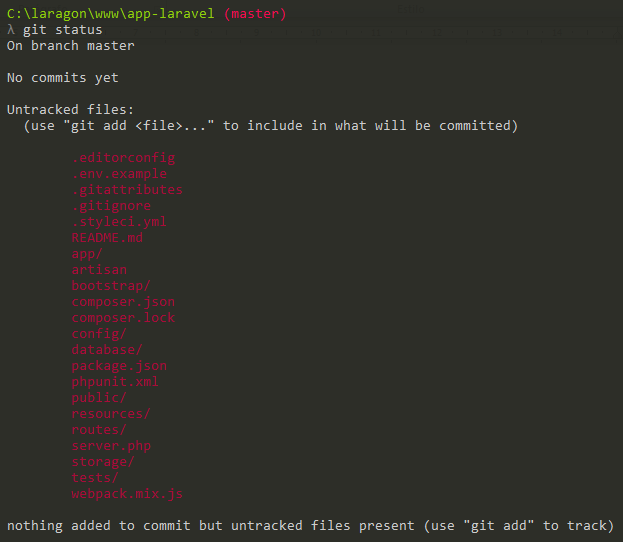


u = untrack

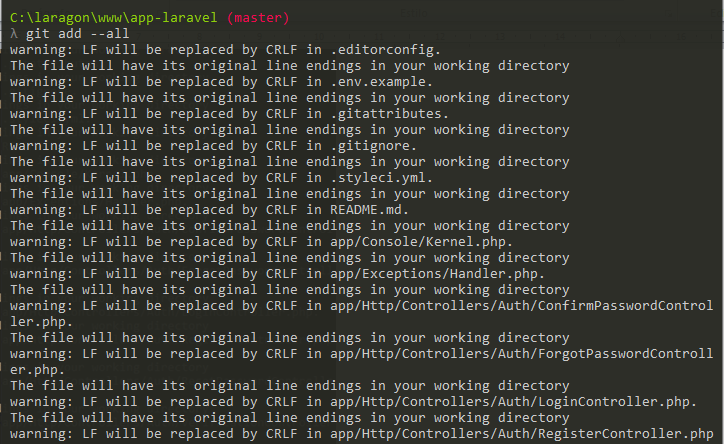
OBS.: Arquivos e diretórios exibidos na cor cinza, estão incluídos em .gitignore e serão ignorados ao se criar o repositório.



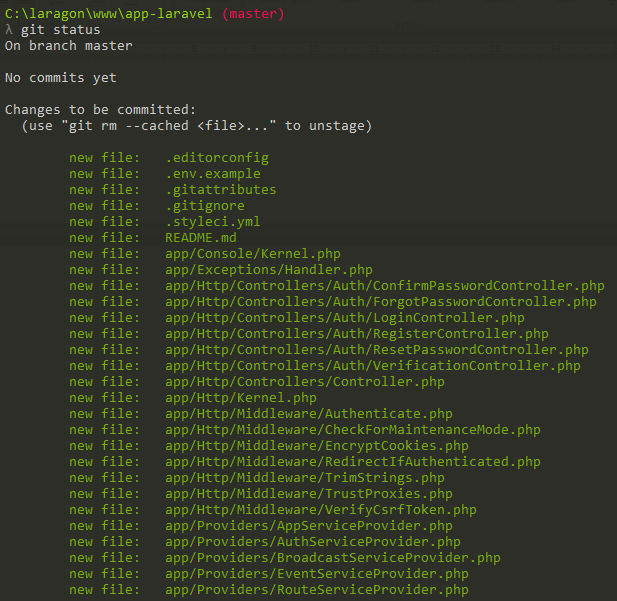
git status

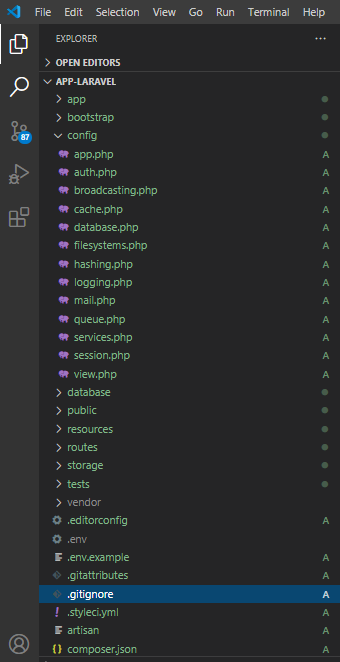


git add --all

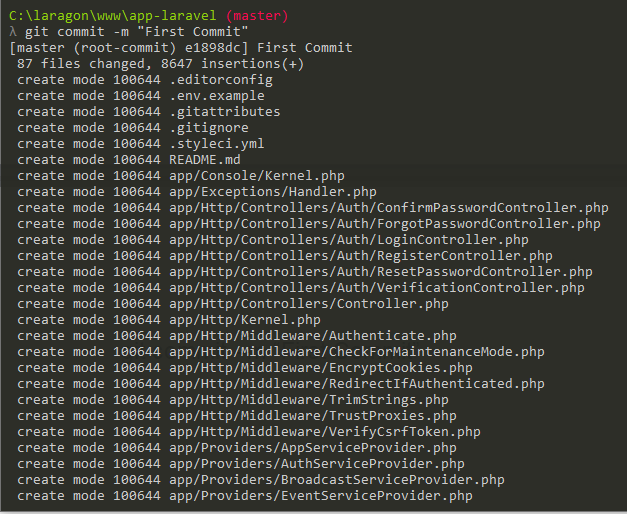


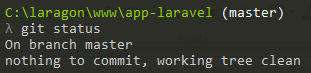
git status





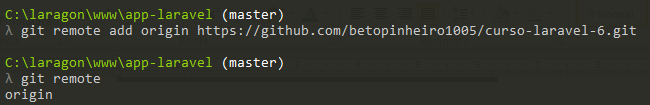
git commit –m “First Commit”



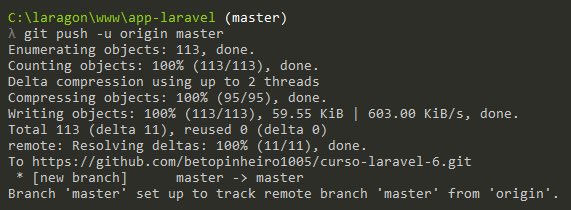


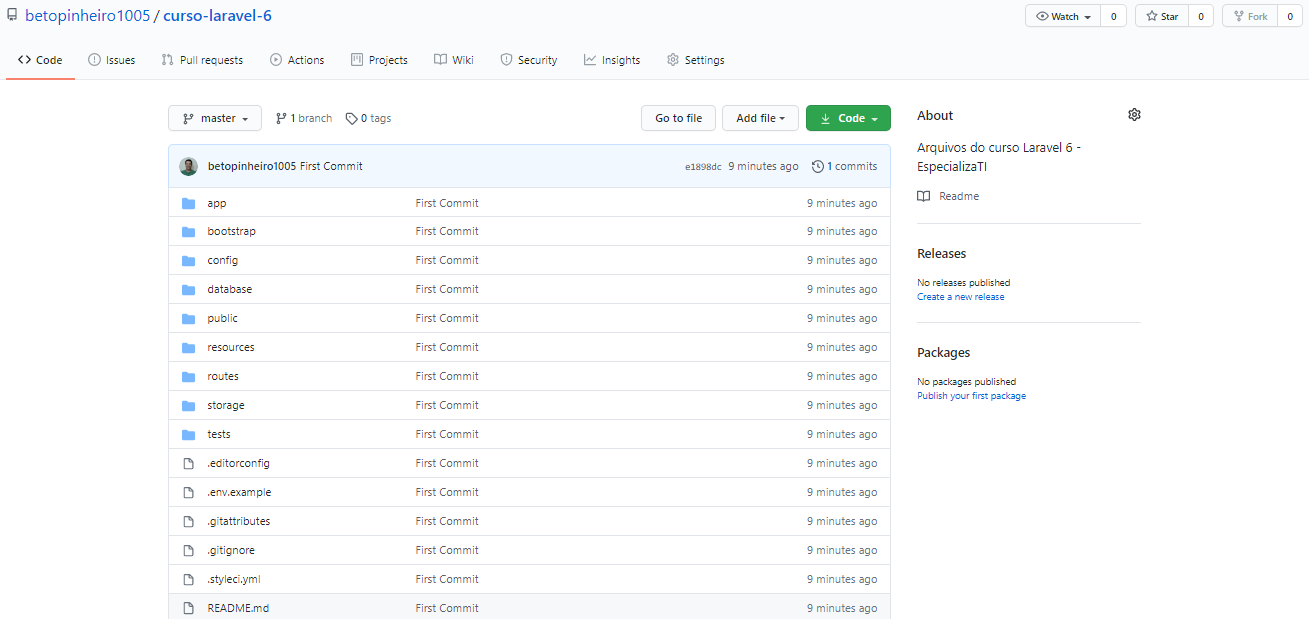


git remote add origin https://github.com/betopinheiro1005/curso-laravel-6.git



git push -u origin master

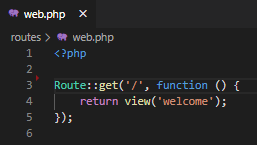


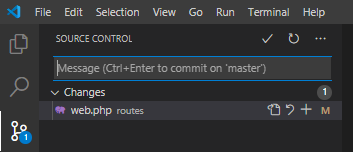


Se precisar clonar o projeto, basta entrar com o comando:

git clone https://github.com/betopinheiro1005/curso-laravel-6.git

- Elimine os comentários do arquivo routes/web.php e salve-o:





- Clique no botão + (Stage changes)



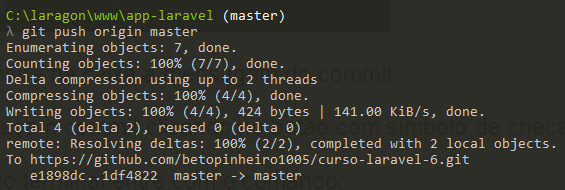


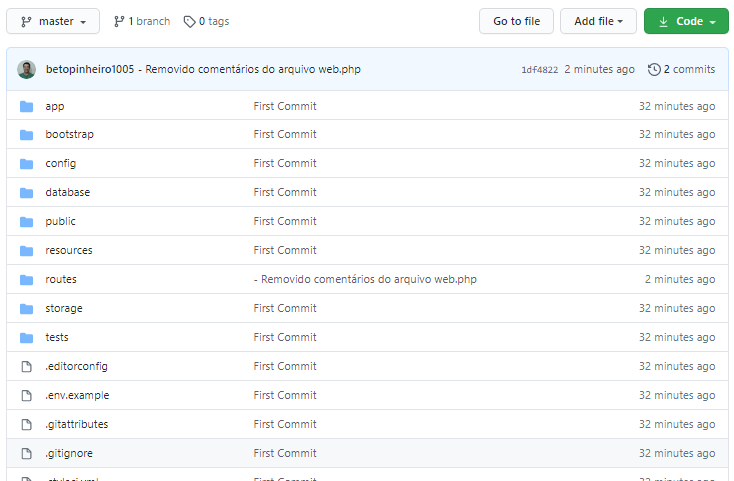
- Insira na caixa a mensagem do commit.

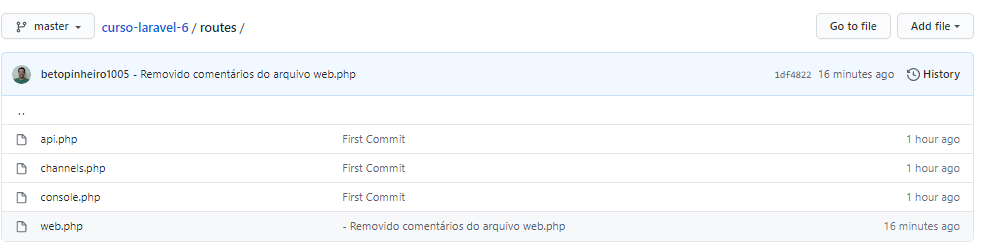
- Faça o commit clicando no botão com símbolo de checado.

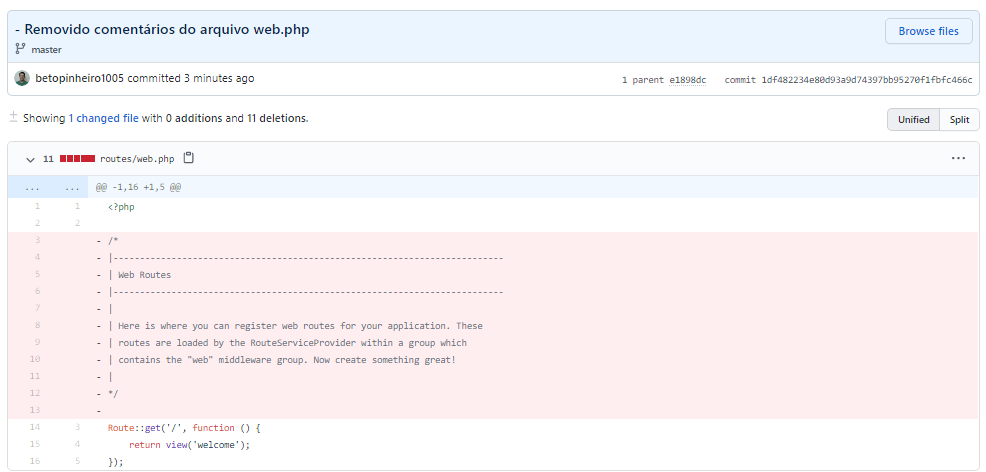
- No terminal entre com o comando:

git push origin master









# Aula 09 – Introdução a Rotas no Laravel 6x

## Retornando um texto

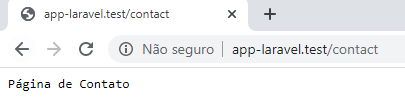
routes/web.php

<?php

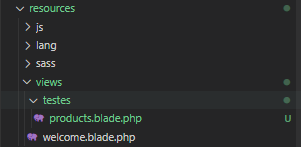
Route::get('/contact', function () {

return 'Página de Contato';

});



## Retornando uma view



routes/web.php

<?php

Route::get('/customers', function () {

return view('testes.customers');

});

resources\views\testes\customers.blade.php

<h1>Listagem de clientes</h1>



# Aula 10 – Rotas any e match no Laravel

## Rota any

A rota any permite todos tipos de acesso de verbo HTTP.

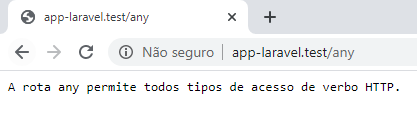
routes/web.php

<?php

Route::any('/any', function () {

return 'A rota any permite todos tipos de acesso de verbo HTTP.';

});



## Rota match

É semelhante a rota any, porém deve ser especificado um array com os verbos HTTP que ela irá aceitar.

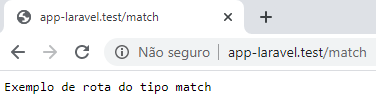
routes/web.php

<?php

Route::match(['get', 'post'],'/match', function () {

return 'Exemplo de rota do tipo match';

});

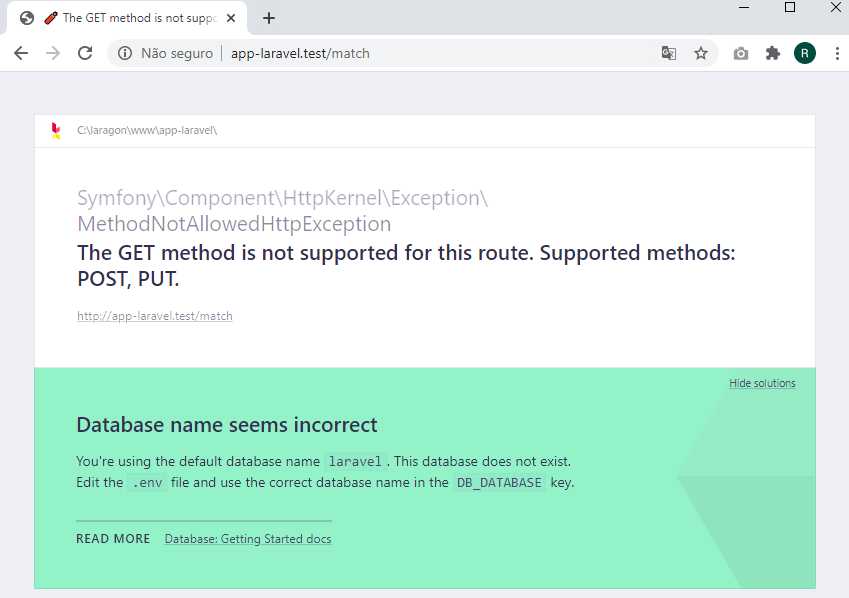


<?php

Route::match(['put', 'post'],'/match', function () {

return 'Exemplo de rota do tipo match';

});



# Aula 11 – Rotas com parâmetros no Laravel

routes/web.php

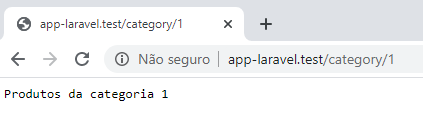
<?php

// Rotas com parâmetros

Route::get('/category/{id}', function ($id) {

return "Produtos da categoria {$id}";

});



routes/web.php

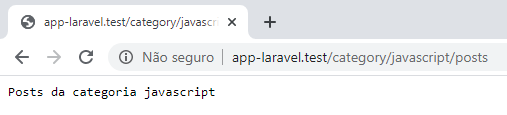
<?php

// Rotas com parâmetros

Route::get('/category/{id}/posts', function ($id) {

return "Posts da categoria {$id}";

});



## Rotas com parâmetros opcionais

routes/web.php

<?php

// Rotas com parâmetros opcionais

Route::get('/products/{id?}', function ($id = '') {

if ($id == '') {

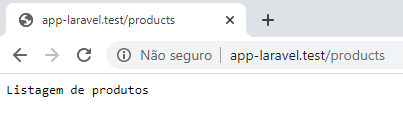
return "Listagem de produtos";

} else {

return "Produtos da categoria {$id}";

}

});





# Aula 12 – Rotas no Laravel com redirect e view

## Rotas com redirect

routes/web.php

<?php

// Redirecionando para uma outra rota

Route::get('/old\_page', function () {

return redirect('/new\_page');

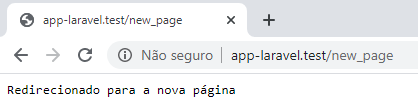
});

Route::get('/new\_page', function () {

return 'Redirecionado para a nova página';

});





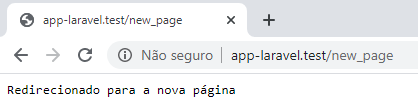
routes/web.php

<?php

// Rotas com redirect

Route::redirect('/old\_page', '/new\_page');





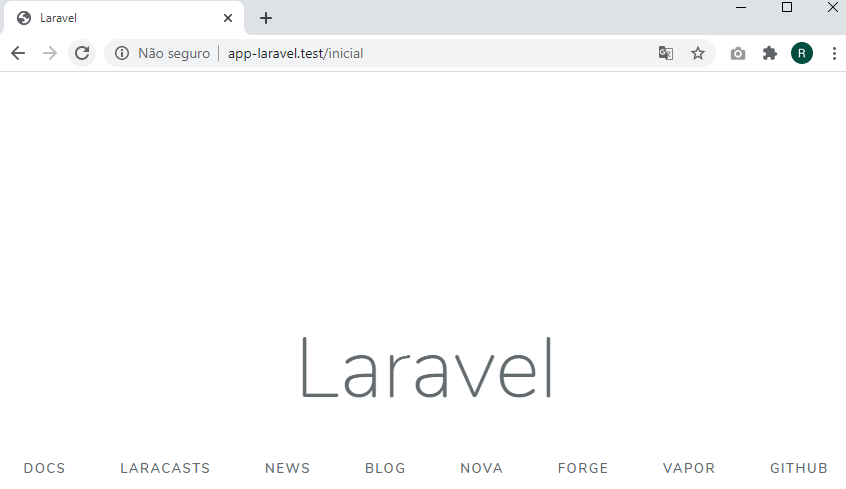
## Redirecionando para uma view

routes/web.php

<?php

// Redirecionando para uma view

Route::view('/clientes', 'testes.customers');



# Aula 13 – Rotas nomeadas no Laravel

routes/web.php

<?php

// Rotas nomeadas

Route::get('/exemplo1', function () {

return redirect()->route('url.name');

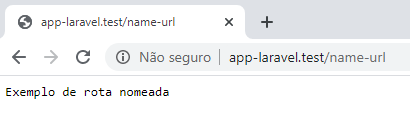
});

Route::get('/name-url', function () {

return 'Exemplo de rota nomeada';

})->name('url.name');





# Aula 14 – Grupo de rotas no Laravel

routes/web.php

<?php

// Grupo de rotas

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

return 'Página de Login';

})->name('login');

Route::get('/panel/dashboard', function () {

return 'Home Panel';

})->middleware('auth');

Route::get('/panel/financeiro', function () {

return 'Financeiro Panel';

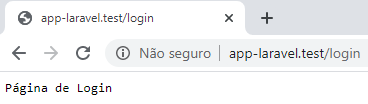
})->middleware('auth');

Route::get('/panel/produtos', function () {

return 'Produtos Panel';

})->middleware('auth');





routes/web.php

<?php

// Grupo de rotas

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

return 'Página de Login';

})->name('login');

Route::middleware([])->group(function () {

Route::prefix('/panel')->group(function() {

Route::get('/dashboard', function () {

return 'Home Panel';

});

Route::get('/financeiro', function () {

return 'Financeiro Panel';

});

Route::get('/produtos', function () {

return 'Produtos Panel';

});

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

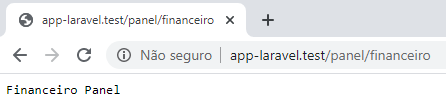
return 'Panel';

});

});

});





routes/web.php

<?php

// Grupo de rotas

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

return 'Página de Login';

})->name('login');

Route::middleware(['auth'])->group(function () {

Route::prefix('/panel')->group(function() {

Route::get('/dashboard', function () {

return 'Home Panel';

});

Route::get('/financeiro', function () {

return 'Financeiro Panel';

});

Route::get('/produtos', function () {

return 'Produtos Panel';

});

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

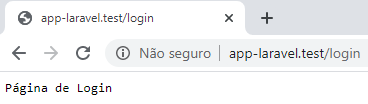
return 'Panel';

});

});

});



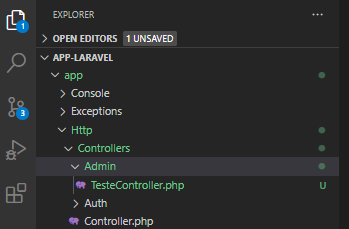


## Trabalhando com controllers

No terminal:

php artisan make:controller Admin\TesteController





app\Http\Controllers\Admin\TesteController.php

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

class TesteController extends Controller

{

public function home() {

return 'Teste Controller - Home';

}

public function dashboard() {

return 'Teste Controller - Dashboard';

}

public function financeiro() {

return 'Teste Controller - Financeiro';

}

public function produtos() {

return 'Teste Controller - Produtos';

}

}

routes/web.php

<?php

// Grupo de rotas

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

return 'Página de Login';

})->name('login');

Route::middleware([])->group(function () {

Route::prefix('/admin')->group(function() {

Route::namespace('Admin')->group(function() {

Route::get('/dashboard', 'TesteController@dashboard')->name('admin.dashboard');

Route::get('/financeiro', 'TesteController@financeiro')->name('admin.financeiro');

Route::get('/produtos', 'TesteController@produtos')->name('admin.produtos');

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

return redirect()->route('admin.dashboard');

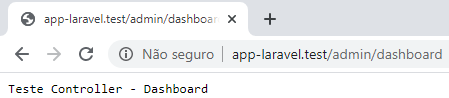
});

});

});

});





routes/web.php

<?php

// Grupo de rotas

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

return 'Página de Login';

})->name('login');

Route::group([

'middleware' => [],

'prefix' => 'admin',

'namespace' => 'Admin'

], function(){

Route::get('/dashboard', 'TesteController@dashboard')->name('admin.dashboard');

Route::get('/financeiro', 'TesteController@financeiro')->name('admin.financeiro');

Route::get('/produtos', 'TesteController@produtos')->name('admin.produtos');

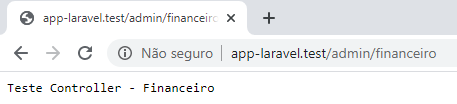
Route::get('/', function () {

return redirect()->route('admin.dashboard');

});

});

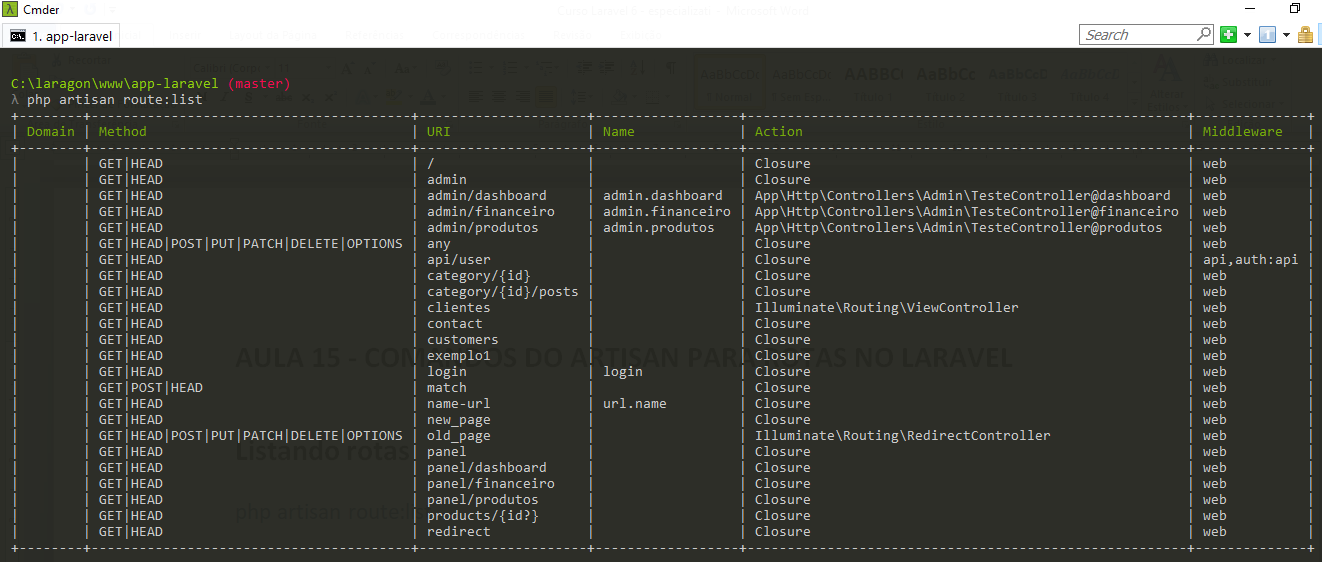




# Aula 15 – Comandos do Artisan para rotas no Laravel

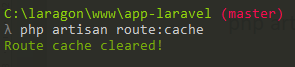
## Listando rotas

php artisan route:list



## Limpando as rotas em cache

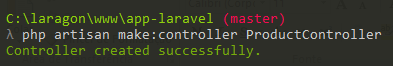
php artisan route:cache

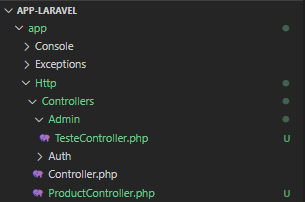


# Aula 16 – Introdução aos controllers no Laravel 6.x

## Criando um controller

php artisan make:controller ProductController





app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function index(){

$products = ['liquidificador', 'TV de leds', 'rádio relógio'];

return $products;

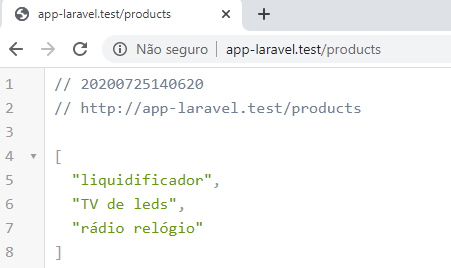
}

}

routes/web.php

<?php

Route::get('/products', 'ProductController@index')->name('products.index');



# Aula 17 – Controllers com Parâmetros de Rotas no Laravel 6.x

routes/web.php

<?php

Route::get('/products/{id}', 'ProductController@show')->name('products.show');

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function index(){

$products = ['liquidificador', 'TV de leds', 'rádio relógio'];

return $products;

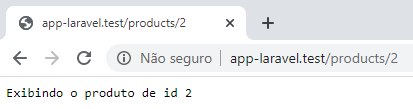
}

public function show($id){

return "Exibindo o produto de id {$id}";

}

}



# Aula 18 – Controllers de CRUD no Laravel 6.x

routes/web.php

<?php

// CRUD

Route::delete('/products/{id}', 'ProductController@destroy')->name('products.destroy');

Route::put('/products/{id}', 'ProductController@update')->name('products.update');

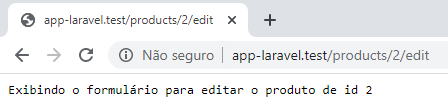
Route::get('/products/{id}/edit', 'ProductController@edit')->name('products.create');

Route::get('/products/create', 'ProductController@create')->name('products.create');

Route::post('/products', 'ProductController@store')->name('products.atore');

Route::get('/products/{id}', 'ProductController@show')->name('products.show');

Route::get('/products', 'ProductController@index')->name('products.index');



app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function index(){

$products = ['liquidificador', 'TV de leds', 'rádio relógio'];

return $products;

}

public function show($id){

return "Exibindo o produto de id {$id}";

}

public function create(){

return "Exibindo o formulário de cadastro de um novo produto";

}

public function store(){

return "Cadastrando um novo produto";

}

public function edit($id){

return "Exibindo o formulário para editar o produto de id {$id}";

}

public function update($id){

return "Editando o produto de id {$id}";

}

public function destroy($id){

return "Deletando o produto de id {$id}";

}

}

## Aula 19 – Controllers Resources no Laravel 6.x

- Apague o controller ProductController criado na aula anterior.

- No arquivo web.php comente as rotas do CRUD criadas manualmente.

## Criando um controller resource

php artisan make:controller ProductController --resource



routes/web.php

<?php

Route::resource('/products', 'ProductController');

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

return "Listando os produtos";

}

/\*\*

\* Show the form for creating a new resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function create()

{

//

}

/\*\*

\* Store a newly created resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @return \Illuminate\Http\Response

\*/

public function store(Request $request)

{

//

}

/\*\*

\* Display the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function show($id)

{

return "Detalhes do produto de id {$id}";

}

/\*\*

\* Show the form for editing the specified resource.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function edit($id)

{

//

}

/\*\*

\* Update the specified resource in storage.

\*

\* @param \Illuminate\Http\Request $request

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

public function update(Request $request, $id)

{

//

}

/\*\*

\* Remove the specified resource from storage.

\*

\* @param int $id

\* @return \Illuminate\Http\Response

\*/

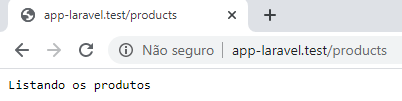
public function destroy($id)

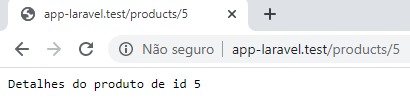
{

//

}

}





# Aula 20 – Injeção de dependências no Laravel 6.x

app\Http\Controllers\ProductController.php

use Illuminate\Http\Request;

public function store(Request $request)

{

//

}

Este trecho de código está criando um objeto da classe Request e armazenando uma instância desse objeto na variável $request.

É o mesmo que fazer:

use Illuminate\Http\Request;

public function store()

{

$request = new Request;

}

Porém no primeiro caso economiza-se 1 linha.

## Criando um construtor para injetar a dependência

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

public function \_\_construct(Request $request){

$this->request = $request;

}

/\*\*

\* Display a listing of the resource.

\*

\* @return \Illuminate\Http\Response

\*/

public function index()

{

return "Listando os produtos";

}

## Verificando o conteúdo do objeto $request da classe de Request

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

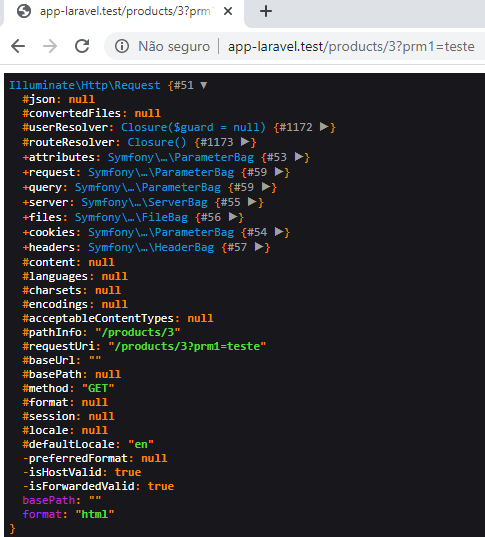
protected $request;

public function \_\_construct(Request $request){

dd($request);

$this->request = $request;

}



## Obtendo o valor do parâmetro passado

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

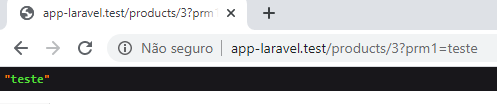
protected $request;

public function \_\_construct(Request $request){

dd($request->prm1);

$this->request = $request;

}



OBS: Se nenhum parâmetro fosse passado retornaria NULL.

# Aula 21 – Middlewares em Controllers no Laravel 6x

Middlewares são filtros. E o Laravel já traz alguns filtros pré-definidos como, por exemplo, os filtros de autenticação que garante somente usuários autenticados que podem acessar os recursos do seu sistema.

## Aplicando um middleware nas rotas do CRUD

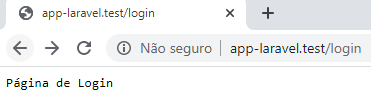
routes/web

<?php

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

Agora para acessar qualquer uma das rotas do CRUD é necessário estar autenticado, caso contrário o usuário será redirecionado para a página de login.





## Trabalhando com os middlewares nos controllers

Outra forma de se trabalhar com middlewares é nos controllers, ao invés das rotas.

routes/web

<?php

Route::resource('/products', 'ProductController');

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

public function \_\_construct(Request $request){

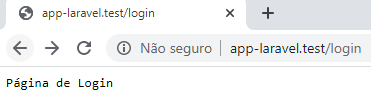
// dd($request->prm1);

$this->middleware('auth');

$this->request = $request;

}





## Especificando os métodos nos quais o middleware será aplicado

### Usando o método only()

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

public function \_\_construct(Request $request){

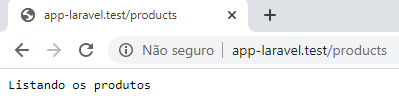
$this->middleware('auth')->only([

'create', 'store'

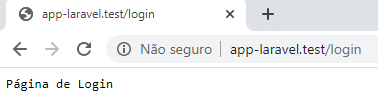
]);

$this->request = $request;

}







### Usando o método except()

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

public function \_\_construct(Request $request){

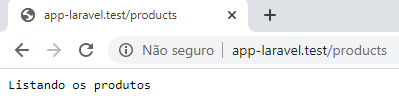
$this->middleware('auth')->except([

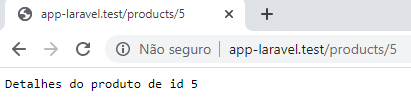
'index', 'show'

]);

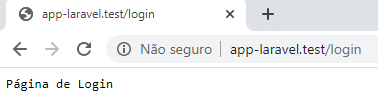
$this->request = $request;

}









# Aula 22 – Introdução a Views no Laravel 6x

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

public function \_\_construct(Request $request){

$this->middleware('auth')->except([

'index', 'show'

]);

$this->request = $request;

}

public function index()

{

return view('teste');

}

}

resources\views\teste.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Minha view</title>

</head>

<body>

<h2>Teste usando views</h2>

</body>

</html>



Uma diretiva começa com @. O blade possui várias diretivas.

O projeto não deve conter lógica dentro de views.

## Passando parâmetros para a view

No controller, é possível passar variáveis para uma view.

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function index()

{

$teste = 123;

return view('teste', [

'teste' => $teste

]);

}

}

resources\views\teste.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Minha view</title>

</head>

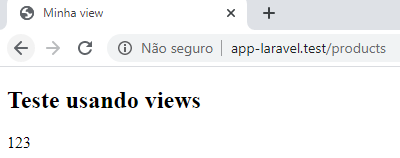
<body>

<h2>Teste usando views</h2>

{{ $teste }}

</body>

</html>



### Outra forma de se passar parâmetros

app\Http\Controllers\ProductController.php

public function index()

{

$teste = 123;

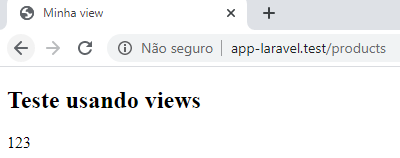
// return view('teste', [

// 'teste' => $teste

// ]);

return view('teste', compact('teste'));

}



# Aula 23 – Impressões no Blade Laravel {{ vs {!!

## Imprimindo com {{

Imprime como texto puro (é uma proteção contra ataques do tipo XSS).

app\Http\Controllers\ProductController.php

public function index()

{

$teste = "<script>alert('Olá mundo!')</script>";

return view('teste', compact('teste'));

}

resources\views\teste.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Minha view</title>

</head>

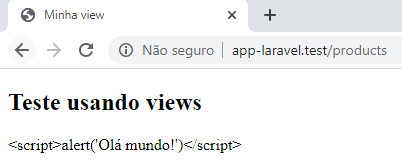
<body>

<h2>Teste usando views</h2>

{{ $teste }}

</body>

</html>



## Imprimindo com {!!

Faz a impressão como realmente é (leva em conta tags HTML e Javascript). Não tem proteção contra ataques do tipo XSS e portanto deve ser evitado ao máximo, ou usado com cuidado.

app\Http\Controllers\ProductController.php

public function index()

{

$teste = "<h1>Olá mundo!</h1>";

return view('teste', compact('teste'));

}

resources\views\teste.blade.php

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Minha view</title>

</head>

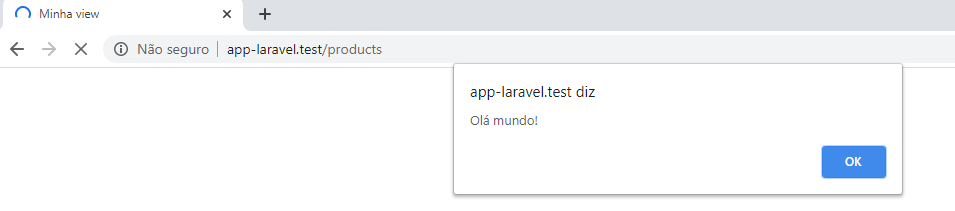
<body>

<h2>Teste usando views</h2>

{!! $teste !!}

</body>

</html>





# Aula 24 – Templates Blade Laravel

app\Http\Controllers\ProductController.php

public function index()

{

$teste = 123;

return view('admin.pages.products.index', compact('teste'));

}

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

<h2>{{ $teste }}</h2>

@endsection

## Página de template

Tudo que é comum a todas as páginas fica aqui.

resources\views\admin\layouts\app.blade.php

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@yield('title') - EspecializaTi</title>

</head>

<body>

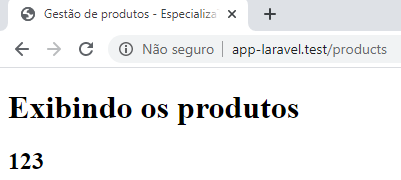
<div class="container">

@yield('content')

</div>

</body>

</html>



# Aula 25 – Estruturas de controle Blade Laravel

app\Http\Controllers\ProductController.php

public function index()

{

$teste = 123;

$teste2 = 456;

$teste3 = 789;

$teste4 = [];

return view('admin.pages.products.index', compact('teste', 'teste2', 'teste3', 'teste4'));

}

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

@if ($teste === '123')

'123'

@elseif($teste === 123)

123

@else

É diferente de '123' ou de 123

@endif

@unless ($teste2 === '456')

456

@else

'456'

@endunless

@isset($teste3)

{{ $teste3 }}

@endisset

@empty($teste4)

<p>Vazio...</p>

@endempty

@auth

<p>Autenticado</p>

@else

<p>Não autenticado</p>

@endauth

@guest

<p>Não autenticado</p>

@endguest

@switch($teste)

@case(1)

Igual a 1

@break

@case(2)

Igual a 2

@break

@case(123)

Igual a 123

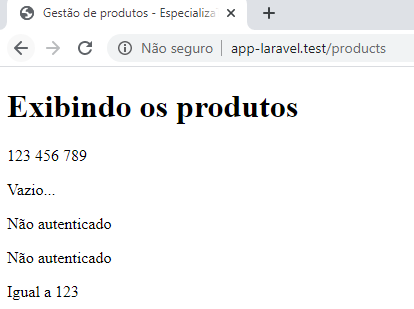
@break

@default

Default

@endswitch

@endsection



# Aula 26 – Estruturas de repetição Blade Laravel

app\Http\Controllers\ProductController.php

public function index()

{

// $teste = "<script>alert('Olá mundo!')</script>";

$teste = 123;

$teste2 = 456;

$teste3 = 789;

$teste4 = [];

$products = ['TV', 'Geladeira', 'Forno', 'Sofá'];

$products2 = [];

return view('admin.pages.products.index', compact('teste', 'teste2', 'teste3', 'teste4', 'products', 'products2'));

}

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

@if(isset($products))

@foreach ($products as $product)

<p class="@if($loop->last) last @elseif($loop->first) first @endif)">{{ $product }}</p>

@endforeach

@endif

<hr>

@forelse ($products2 as $product)

<p >{{ $product }}</p>

@empty

<p>Não existem produtos cadastrados!</p>

@endforelse

<hr>

@if ($teste === '123')

'123'

@elseif($teste === 123)

123

@else

É diferente de '123' ou de 123

@endif

@unless ($teste2 === '456')

456

@else

'456'

@endunless

@isset($teste3)

{{ $teste3 }}

@endisset

@empty($teste4)

<p>Vazio...</p>

@endempty

@auth

<p>Autenticado</p>

@else

<p>Não autenticado</p>

@endauth

@guest

<p>Não autenticado</p>

@endguest

@switch($teste)

@case(1)

Igual a 1

@break

@case(2)

Igual a 2

@break

@case(123)

Igual a 123

@break

@default

Default

@endswitch

@endsection

<style>

.last{

background: #CCC;

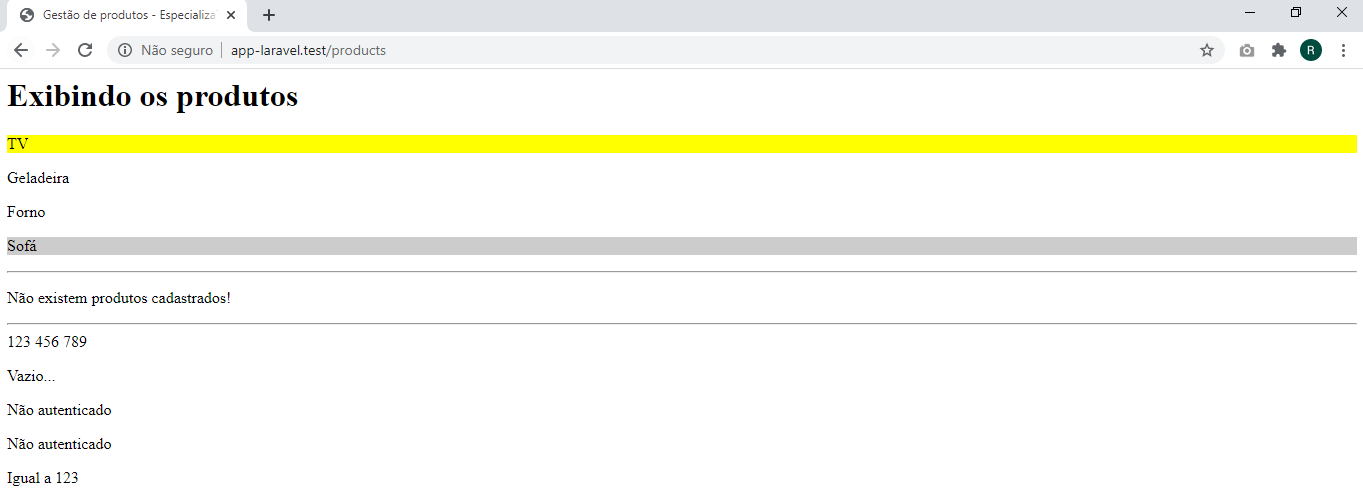
}

.first{

background: yellow;

}

</style>



# Aula 27 – Includes, components e slots no Blade Laravel

## Incluindo o Bootstrap via CDN

resources\views\admin\layouts\app.blade.php

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@yield('title') - EspecializaTi</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

</head>

<body>

<div class="container">

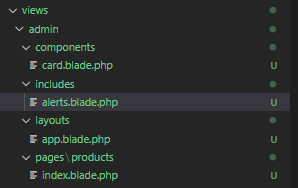
@yield('content')

</div>

</body>

</html>

## Includes



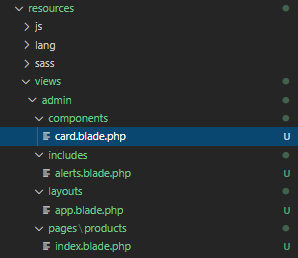
resources\views\admin\includes\alerts.blade.php

<div class="alert">

Alert - {{ $content ?? '' }}

</div>

## Components e Slots



resources\views\admin\components\card.blade.php

<div class="card">

<div class="card-header">

{{ $title }}

</div>

<div class="card-body">

{{ $slot }}

</div>

</div>

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

@component('admin.components.card')

@slot('title')

<h3>Título Card</h3>

@endslot

Um card de exemplo

@endcomponent

<hr>

@include('admin.includes.alerts', ['content' => 'Alerta de preços de produtos'])

<hr>

@if(isset($products))

@foreach ($products as $product)

<p class="@if($loop->last) last @elseif($loop->first) first @endif)">{{ $product }}</p>

@endforeach

@endif

<hr>

@if(isset($products))

@foreach ($products as $product)

<p class="@if($loop->last) last @elseif($loop->first) first @endif)">{{ $product }}</p>

@endforeach

@endif

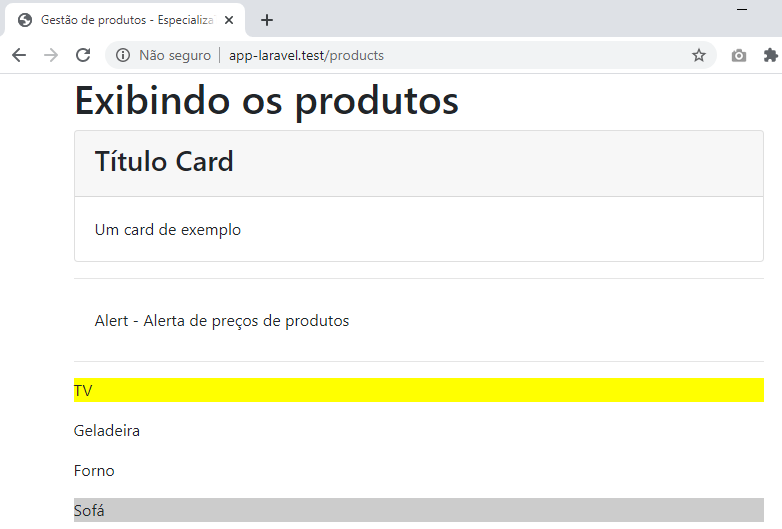
<hr>

.

.

.

@endsection



# Aula 28 – Stacks Blade Laravel

## stack e push

resources\views\admin\layouts\app.blade.php

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@yield('title') - EspecializaTi</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

@stack('styles')

</head>

<body>

<div class="container">

@yield('content')

</div>

@stack('scripts')

</body>

</html>

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

@component('admin.components.card')

@slot('title')

<h3>Título Card</h3>

@endslot

Um card de exemplo

@endcomponent

<hr>

@include('admin.includes.alerts', ['content' => 'Alerta de preços de produtos'])

<hr>

@if(isset($products))

@foreach ($products as $product)

<p class="@if($loop->last) last @elseif($loop->first) first @endif)">{{ $product }}</p>

@endforeach

@endif

<hr>

.

.

.

@endsection

@push('styles')

<style>

.last{

background: #CCC;

}

.first{

background: yellow;

}

</style>

@endpush

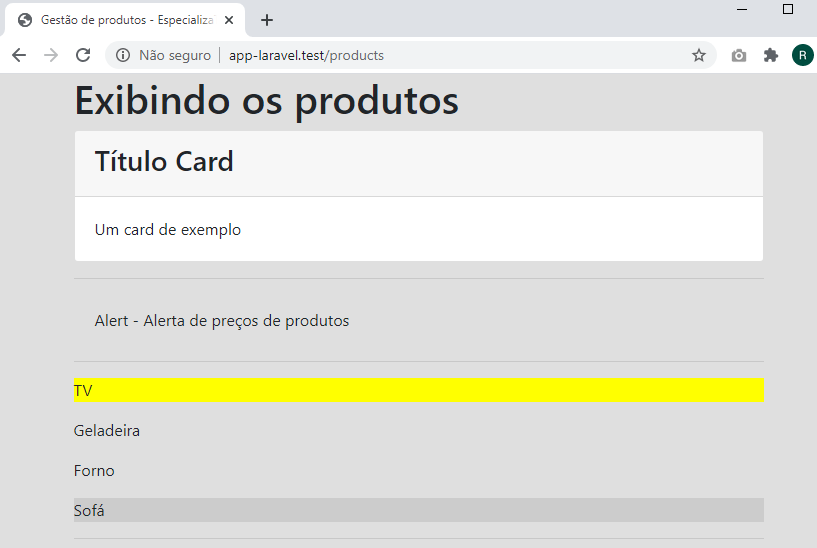
@push('scripts')

<script>

document.body.style.background = '#dfdfdf'

</script>

@endpush



# Aula 29 – Formulários no Laravel (csrf)

resources\views\admin\pages\products\create.blade.php

@extends('admin.layouts.app')

@section('title', 'Cadastrar novo produto')

@section('content')

<h1>Cadastrar novo produto</h1>

<form action="{{ route('products.store') }}" method="post">

@csrf

<input type="text" name="name" placeholder="Nome:">

<input type="text" name="description" placeholder="Descrição:">

<button type="submit">Enviar</button>

</form>

@endsection

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

public function \_\_construct(Request $request){

// $this->middleware('auth')->except([

// 'index', 'show'

// ]);

$this->request = $request;

}

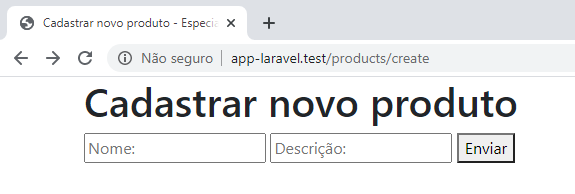
public function store(Request $request)

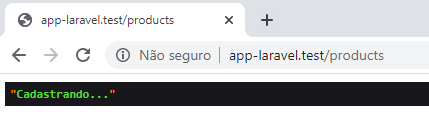
{

dd('Cadastrando...');

}

}





# Aula 30 – Formulários com Métodos de Envio no Laravel (verbo http)

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function edit($id)

{

return view('admin.pages.products.edit', compact('id'));

}

public function update(Request $request, $id)

{

dd("Editando o produto { $id }");

}

}

resources\views\admin\pages\products\edit.blade.php

@extends('admin.layouts.app')

@section('title', 'Editar produto')

@section('content')

<h1>Editar produto {{ $id }}</h1>

<form action="{{ route('products.update', $id) }}" method="post">

@method('PUT')

@csrf

<input type="text" name="name" placeholder="Nome:">

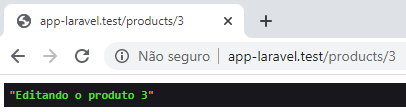
<input type="text" name="description" placeholder="Descrição:">

<button type="submit">Enviar</button>

</form>

@endsection





# Aula 31 – Outros recursos de Laravel

## Comentários no Blade

resources\views\admin\includes\alerts.blade.php

<div class="alert">

Alert - {{ $content ?? '' }}

{{-- Com essa notação, se a variável não existir será exibida uma string vazia (não será gerado um erro) --}}

</div>

## Limpando o cache das views

php artisan view:clear



# Aula 32 – Pegar dados do formulário com o Laravel

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

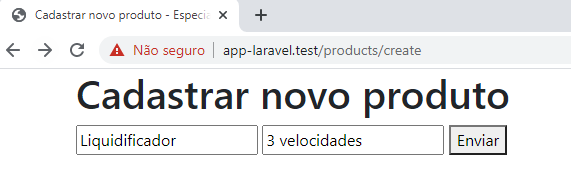
public function store(Request $request)

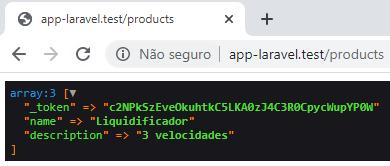
{

dd($request->all());

}

}





app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

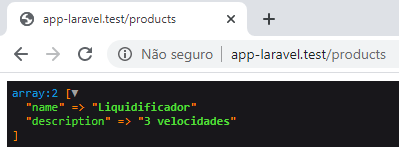
public function store(Request $request)

{

dd($request->only(['name', 'description']));

}

}



app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

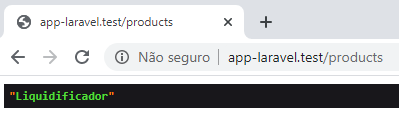
public function store(Request $request)

{

dd($request->name);

}

}



app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

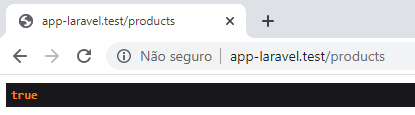
public function store(Request $request)

{

dd($request->has('name'));

}

}



app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

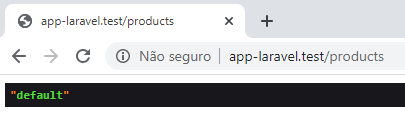
public function store(Request $request)

{

dd($request->input('teste', 'default'));

}

}



app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

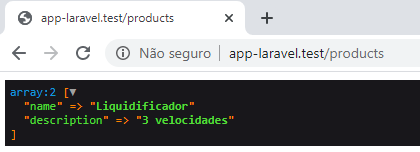
public function store(Request $request)

{

dd($request->except('\_token));

}

}



# Aula 33 – Upload de Arquivos no Laravel, com Request

resources\views\admin\pages\products\create.blade.php

@extends('admin.layouts.app')

@section('title', 'Cadastrar novo produto')

@section('content')

<h1>Cadastrar novo produto</h1>

<form action="{{ route('products.store') }}" method="post" enctype="multipart/form-data">

@csrf

<input type="text" name="name" placeholder="Nome:">

<input type="text" name="description" placeholder="Descrição:">

<input type="file" name="photo">

<button type="submit">Enviar</button>

</form>

@endsection

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

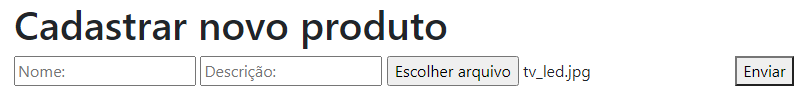
public function store(Request $request)

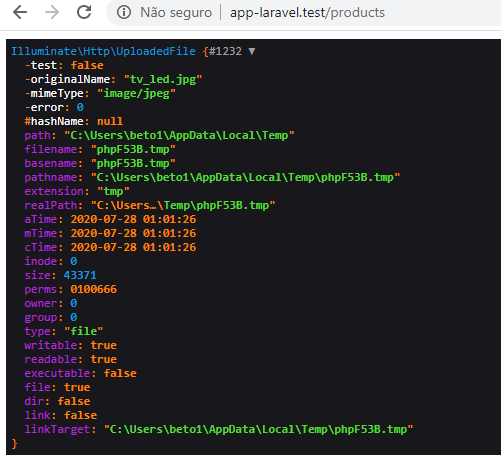
{

dd($request->file('photo'));

}

}





app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

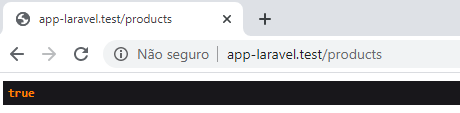
public function store(Request $request)

{

dd($request->file('photo')->isValid());

}

}



app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function store(Request $request)

{

public function store(Request $request)

{

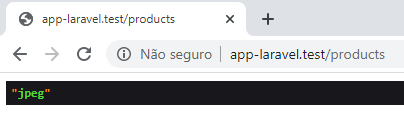
if($request->file('photo')->isValid()) {

dd($request->photo->extension());

}

}

}



app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function store(Request $request)

{

public function store(Request $request)

{

if($request->file('photo')->isValid()) {

dd($request->photo->getClientOriginalName());

}

}

}



Antes de fazer o upload é necessário fazer algumas configurações. Você pode salvar em local (na pasta "storage/app") ou em público.

## Trabalhando em configuração local

config\filesystems.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Default Filesystem Disk

|--------------------------------------------------------------------------

|

| Here you may specify the default filesystem disk that should be used

| by the framework. The "local" disk, as well as a variety of cloud

| based disks are available to your application. Just store away!

|

\*/

'default' => env('FILESYSTEM\_DRIVER', 'local'),

/\*

|--------------------------------------------------------------------------

| Default Cloud Filesystem Disk

|--------------------------------------------------------------------------

|

| Many applications store files both locally and in the cloud. For this

| reason, you may specify a default "cloud" driver here. This driver

| will be bound as the Cloud disk implementation in the container.

|

\*/

'cloud' => env('FILESYSTEM\_CLOUD', 's3'),

/\*

|--------------------------------------------------------------------------

| Filesystem Disks

|--------------------------------------------------------------------------

|

| Here you may configure as many filesystem "disks" as you wish, and you

| may even configure multiple disks of the same driver. Defaults have

| been setup for each driver as an example of the required options.

|

| Supported Drivers: "local", "ftp", "sftp", "s3"

|

\*/

'disks' => [

'local' => [

'driver' => 'local',

'root' => storage\_path('app'),

],

'public' => [

'driver' => 'local',

'root' => storage\_path('app/public'),

'url' => env('APP\_URL').'/storage',

'visibility' => 'public',

],

's3' => [

'driver' => 's3',

'key' => env('AWS\_ACCESS\_KEY\_ID'),

'secret' => env('AWS\_SECRET\_ACCESS\_KEY'),

'region' => env('AWS\_DEFAULT\_REGION'),

'bucket' => env('AWS\_BUCKET'),

'url' => env('AWS\_URL'),

'endpoint' => env('AWS\_ENDPOINT'),

],

],

];

## Fazendo o armazenamento do arquivo

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function store(Request $request)

{

public function store(Request $request)

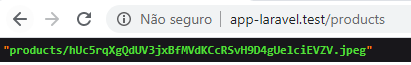
{

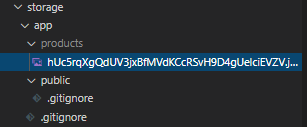
if($request->file('photo')->isValid()) {

dd($request->file('photo')->store('products')); }

}

}





### Definindo um nome customizado

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function store(Request $request)

{

if($request->file('photo')->isValid()) {

$nameFile = $request->name . '.' . $request->photo->extension();

dd($request->file('photo')->storeAs('products', $nameFile));

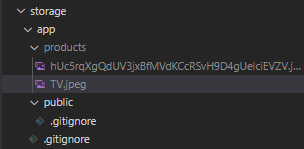
}

}

}







# Aula 34 – Upload de Arquivos Públicos no Laravel

## Trabalhando em configuração public

config\filesystems.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Default Filesystem Disk

|--------------------------------------------------------------------------

|

| Here you may specify the default filesystem disk that should be used

| by the framework. The "local" disk, as well as a variety of cloud

| based disks are available to your application. Just store away!

|

\*/

'default' => env('FILESYSTEM\_DRIVER', 'public'),

.

.

.

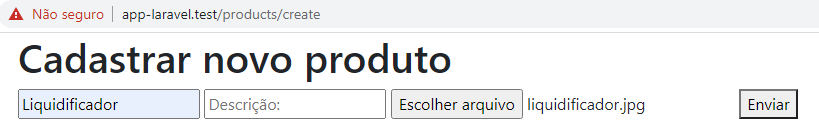
];

## Limpando configurações em cache

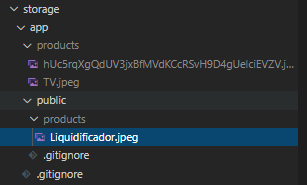
php artisan config:clear



## Armazenando um arquivo de imagem em “storage/app/public”



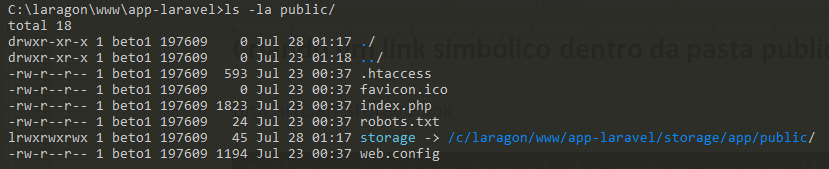


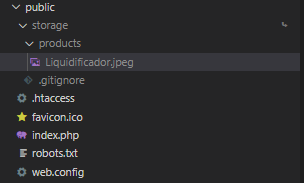


## Criando um link simbólico dentro da pasta public

php artisan storage:link









# Aula 35 – Validações no Laravel 6

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function store(Request $request)

{

$request->validate([

'name' => 'required|min:3|max:255',

'description' => 'nullable|min:3|max:10000',

'photo' => 'required|image'

]);

dd('ok');

}

}

resources\views\admin\pages\products\create.blade.php

@extends('admin.layouts.app')

@section('title', 'Cadastrar novo produto')

@section('content')

<h1>Cadastrar novo produto</h1>

@if ($errors->any())

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

@endif

<form action="{{ route('products.store') }}" method="post" enctype="multipart/form-data">

@csrf

<input type="text" name="name" placeholder="Nome:">

<input type="text" name="description" placeholder="Descrição:">

<input type="file" name="photo">

<button type="submit">Enviar</button>

</form>

@endsection

## Internacionalização (pt\_BR)

config\app.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Application Name

|--------------------------------------------------------------------------

|

| This value is the name of your application. This value is used when the

| framework needs to place the application's name in a notification or

| any other location as required by the application or its packages.

|

\*/

'name' => env('APP\_NAME', 'Laravel'),

/\*

|--------------------------------------------------------------------------

| Application Environment

|--------------------------------------------------------------------------

|

| This value determines the "environment" your application is currently

| running in. This may determine how you prefer to configure various

| services the application utilizes. Set this in your ".env" file.

|

\*/

'env' => env('APP\_ENV', 'production'),

/\*

|--------------------------------------------------------------------------

| Application Debug Mode

|--------------------------------------------------------------------------

|

| When your application is in debug mode, detailed error messages with

| stack traces will be shown on every error that occurs within your

| application. If disabled, a simple generic error page is shown.

|

\*/

'debug' => (bool) env('APP\_DEBUG', false),

/\*

|--------------------------------------------------------------------------

| Application URL

|--------------------------------------------------------------------------

|

| This URL is used by the console to properly generate URLs when using

| the Artisan command line tool. You should set this to the root of

| your application so that it is used when running Artisan tasks.

|

\*/

'url' => env('APP\_URL', 'http://localhost'),

'asset\_url' => env('ASSET\_URL', null),

/\*

|--------------------------------------------------------------------------

| Application Timezone

|--------------------------------------------------------------------------

|

| Here you may specify the default timezone for your application, which

| will be used by the PHP date and date-time functions. We have gone

| ahead and set this to a sensible default for you out of the box.

|

\*/

'timezone' => 'America/Sao\_Paulo',

/\*

|--------------------------------------------------------------------------

| Application Locale Configuration

|--------------------------------------------------------------------------

|

| The application locale determines the default locale that will be used

| by the translation service provider. You are free to set this value

| to any of the locales which will be supported by the application.

|

\*/

'locale' => 'pt\_BR',

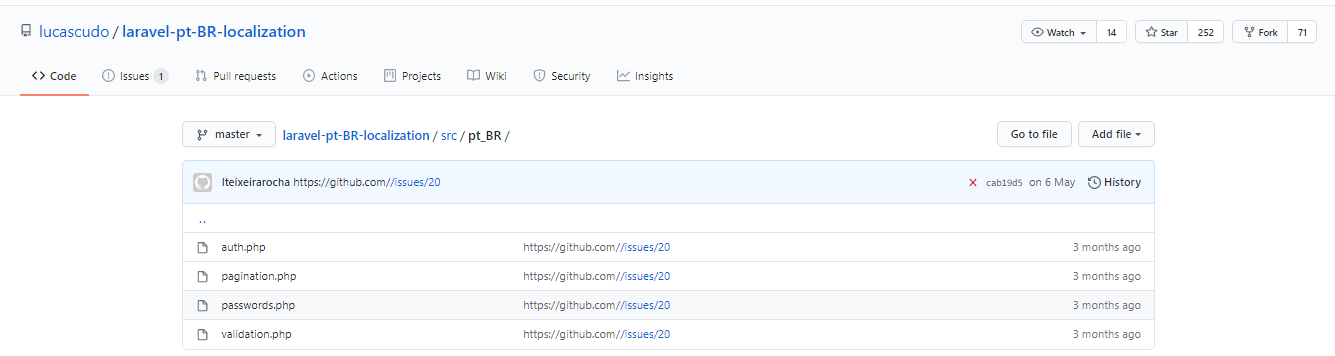
.

.

.

];

https://github.com/lucascudo/laravel-pt-BR-localization/tree/master/src/pt\_BR



resources\lang\pt\_BR\validation.php

<?php

return [

/\*

|--------------------------------------------------------------------------

| Validation Language Lines

|--------------------------------------------------------------------------

|

| The following language lines contain the default error messages used by

| the validator class. Some of these rules have multiple versions such

| as the size rules. Feel free to tweak each of these messages here.

|

\*/

'accepted' => 'O campo :attribute deve ser aceito.',

'active\_url' => 'O campo :attribute não é uma URL válida.',

'after' => 'O campo :attribute deve ser uma data posterior a :date.',

'after\_or\_equal' => 'O campo :attribute deve ser uma data posterior ou igual a :date.',

'alpha' => 'O campo :attribute só pode conter letras.',

'alpha\_dash' => 'O campo :attribute só pode conter letras, números e traços.',

'alpha\_num' => 'O campo :attribute só pode conter letras e números.',

'array' => 'O campo :attribute deve ser uma matriz.',

'before' => 'O campo :attribute deve ser uma data anterior :date.',

'before\_or\_equal' => 'O campo :attribute deve ser uma data anterior ou igual a :date.',

'between' => [

'numeric' => 'O campo :attribute deve ser entre :min e :max.',

'file' => 'O campo :attribute deve ser entre :min e :max kilobytes.',

'string' => 'O campo :attribute deve ser entre :min e :max caracteres.',

'array' => 'O campo :attribute deve ter entre :min e :max itens.',

],

'boolean' => 'O campo :attribute deve ser verdadeiro ou falso.',

'confirmed' => 'O campo :attribute de confirmação não confere.',

'date' => 'O campo :attribute não é uma data válida.',

'date\_equals' => 'O campo :attribute deve ser uma data igual a :date.',

'date\_format' => 'O campo :attribute não corresponde ao formato :format.',

'different' => 'Os campos :attribute e :other devem ser diferentes.',

'digits' => 'O campo :attribute deve ter :digits dígitos.',

'digits\_between' => 'O campo :attribute deve ter entre :min e :max dígitos.',

'dimensions' => 'O campo :attribute tem dimensões de imagem inválidas.',

'distinct' => 'O campo :attribute campo tem um valor duplicado.',

'email' => 'O campo :attribute deve ser um endereço de e-mail válido.',

'ends\_with' => 'O campo :attribute deve terminar com um dos seguintes: :values',

'exists' => 'O campo :attribute selecionado é inválido.',

'file' => 'O campo :attribute deve ser um arquivo.',

'filled' => 'O campo :attribute deve ter um valor.',

'gt' => [

'numeric' => 'O campo :attribute deve ser maior que :value.',

'file' => 'O campo :attribute deve ser maior que :value kilobytes.',

'string' => 'O campo :attribute deve ser maior que :value caracteres.',

'array' => 'O campo :attribute deve conter mais de :value itens.',

],

'gte' => [

'numeric' => 'O campo :attribute deve ser maior ou igual a :value.',

'file' => 'O campo :attribute deve ser maior ou igual a :value kilobytes.',

'string' => 'O campo :attribute deve ser maior ou igual a :value caracteres.',

'array' => 'O campo :attribute deve conter :value itens ou mais.',

],

'image' => 'O campo :attribute deve ser uma imagem.',

'in' => 'O campo :attribute selecionado é inválido.',

'in\_array' => 'O campo :attribute não existe em :other.',

'integer' => 'O campo :attribute deve ser um número inteiro.',

'ip' => 'O campo :attribute deve ser um endereço de IP válido.',

'ipv4' => 'O campo :attribute deve ser um endereço IPv4 válido.',

'ipv6' => 'O campo :attribute deve ser um endereço IPv6 válido.',

'json' => 'O campo :attribute deve ser uma string JSON válida.',

'lt' => [

'numeric' => 'O campo :attribute deve ser menor que :value.',

'file' => 'O campo :attribute deve ser menor que :value kilobytes.',

'string' => 'O campo :attribute deve ser menor que :value caracteres.',

'array' => 'O campo :attribute deve conter menos de :value itens.',

],

'lte' => [

'numeric' => 'O campo :attribute deve ser menor ou igual a :value.',

'file' => 'O campo :attribute deve ser menor ou igual a :value kilobytes.',

'string' => 'O campo :attribute deve ser menor ou igual a :value caracteres.',

'array' => 'O campo :attribute não deve conter mais que :value itens.',

],

'max' => [

'numeric' => 'O campo :attribute não pode ser superior a :max.',

'file' => 'O campo :attribute não pode ser superior a :max kilobytes.',

'string' => 'O campo :attribute não pode ser superior a :max caracteres.',

'array' => 'O campo :attribute não pode ter mais do que :max itens.',

],

'mimes' => 'O campo :attribute deve ser um arquivo do tipo: :values.',

'mimetypes' => 'O campo :attribute deve ser um arquivo do tipo: :values.',

'min' => [

'numeric' => 'O campo :attribute deve ser pelo menos :min.',

'file' => 'O campo :attribute deve ter pelo menos :min kilobytes.',

'string' => 'O campo :attribute deve ter pelo menos :min caracteres.',

'array' => 'O campo :attribute deve ter pelo menos :min itens.',

],

'not\_in' => 'O campo :attribute selecionado é inválido.',

'not\_regex' => 'O campo :attribute possui um formato inválido.',

'numeric' => 'O campo :attribute deve ser um número.',

'password' => 'A senha está incorreta.',

'present' => 'O campo :attribute deve estar presente.',

'regex' => 'O campo :attribute tem um formato inválido.',

'required' => 'O campo :attribute é obrigatório.',

'required\_if' => 'O campo :attribute é obrigatório quando :other for :value.',

'required\_unless' => 'O campo :attribute é obrigatório exceto quando :other for :values.',

'required\_with' => 'O campo :attribute é obrigatório quando :values está presente.',

'required\_with\_all' => 'O campo :attribute é obrigatório quando :values está presente.',

'required\_without' => 'O campo :attribute é obrigatório quando :values não está presente.',

'required\_without\_all' => 'O campo :attribute é obrigatório quando nenhum dos :values estão presentes.',

'same' => 'Os campos :attribute e :other devem corresponder.',

'size' => [

'numeric' => 'O campo :attribute deve ser :size.',

'file' => 'O campo :attribute deve ser :size kilobytes.',

'string' => 'O campo :attribute deve ser :size caracteres.',

'array' => 'O campo :attribute deve conter :size itens.',

],

'starts\_with' => 'O campo :attribute deve começar com um dos seguintes valores: :values',

'string' => 'O campo :attribute deve ser uma string.',

'timezone' => 'O campo :attribute deve ser uma zona válida.',

'unique' => 'O campo :attribute já está sendo utilizado.',

'uploaded' => 'Ocorreu uma falha no upload do campo :attribute.',

'url' => 'O campo :attribute tem um formato inválido.',

'uuid' => 'O campo :attribute deve ser um UUID válido.',

/\*

|--------------------------------------------------------------------------

| Custom Validation Language Lines

|--------------------------------------------------------------------------

|

| Here you may specify custom validation messages for attributes using the

| convention "attribute.rule" to name the lines. This makes it quick to

| specify a specific custom language line for a given attribute rule.

|

\*/

'custom' => [

'attribute-name' => [

'rule-name' => 'custom-message',

],

],

/\*

|--------------------------------------------------------------------------

| Custom Validation Attributes

|--------------------------------------------------------------------------

|

| The following language lines are used to swap our attribute placeholder

| with something more reader friendly such as "E-Mail Address" instead

| of "email". This simply helps us make our message more expressive.

|

\*/

'attributes' => [

'address' => 'endereço',

'age' => 'idade',

'body' => 'conteúdo',

'city' => 'cidade',

'country' => 'país',

'date' => 'data',

'day' => 'dia',

'excerpt' => 'resumo',

'first\_name'=> 'primeiro nome',

'gender' => 'gênero',

'hour' => 'hora',

'last\_name' => 'sobrenome',

'message' => 'mensagem',

'minute' => 'minuto',

'mobile' => 'celular',

'month' => 'mês',

'name' => 'nome',

'password' => 'senha',

'phone' => 'telefone',

'second' => 'segundo',

'sex' => 'sexo',

'state' => 'estado',

'subject' => 'assunto',

'text' => 'texto',

'time' => 'hora',

'title' => 'título',

'username' => 'usuário',

'year' => 'ano',

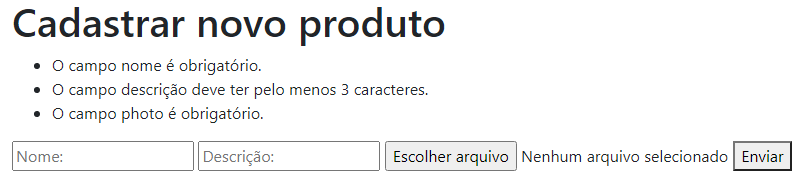
'description' => 'descrição',

'password\_confirmation' => 'confirmação da senha',

],

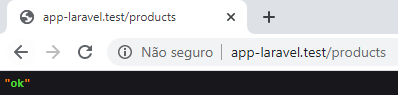
];





### Cadastrando um produto sem erro de validação





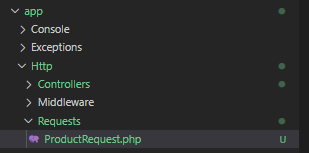
# Aula 36 – Validações no Laravel 6 com Form Request

É a forma correta de fazer validação no Laravel.

## Criando um request

php artisan make:request ProductRequest





app\Http\Requests\ProductRequest.php

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class ProductRequest extends FormRequest

{

public function authorize()

{

return true;

}

public function rules()

{

return [

'name' => 'required|min:3|max:255',

'description' => 'nullable|min:3|max:10000',

'photo' => 'required|image'

];

}

public function messages(){

return[

'name.required' => 'Nome é obrigatório!',

'name.min' => 'Nome deve conter pelo menos 3 caracteres!',

'photo.required' => 'Ops! Uma imagem deve ser informada!'

];

}

}

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use App\Http\Requests\ProductRequest;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

public function \_\_construct(Request $request){

$this->request = $request;

}

public function store(ProductRequest $request)

{

dd('Ok');

}

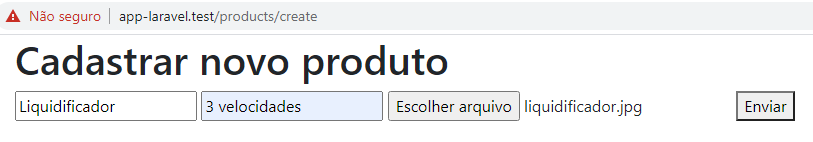
public function update(ProductRequest $request, $id)

{

dd("Editando o produto {$id}");

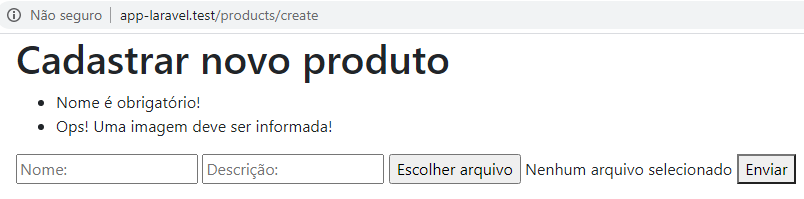
}

}









resources\views\admin\pages\products\create.blade.php

@extends('admin.layouts.app')

@section('title', 'Cadastrar novo produto')

@section('content')

<h1>Cadastrar novo produto</h1>

@if ($errors->any())

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

@endif

<form action="{{ route('products.store') }}" method="post" enctype="multipart/form-data">

@csrf

<input type="text" name="name" placeholder="Nome:" value={{ old('name') }}>

<input type="text" name="description" placeholder="Descrição:" value={{ old('description') }}>

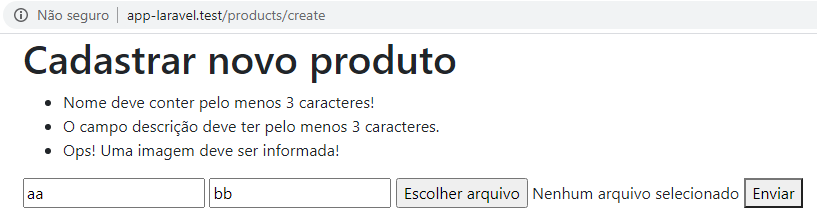
<input type="file" name="photo">

<button type="submit">Enviar</button>

</form>

@endsection





# Aula 37 – Conectar o Laravel 6.x ao Banco de Dados

.env

APP\_NAME=Laravel

APP\_ENV=local

APP\_KEY=base64:+4asd3CqMBOVOIBxcRFx6oBOOHS/wd3lbV/towibWPQ=

APP\_DEBUG=true

APP\_URL=http://app-laravel.test/

LOG\_CHANNEL=stack

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=curso\_laravel\_6

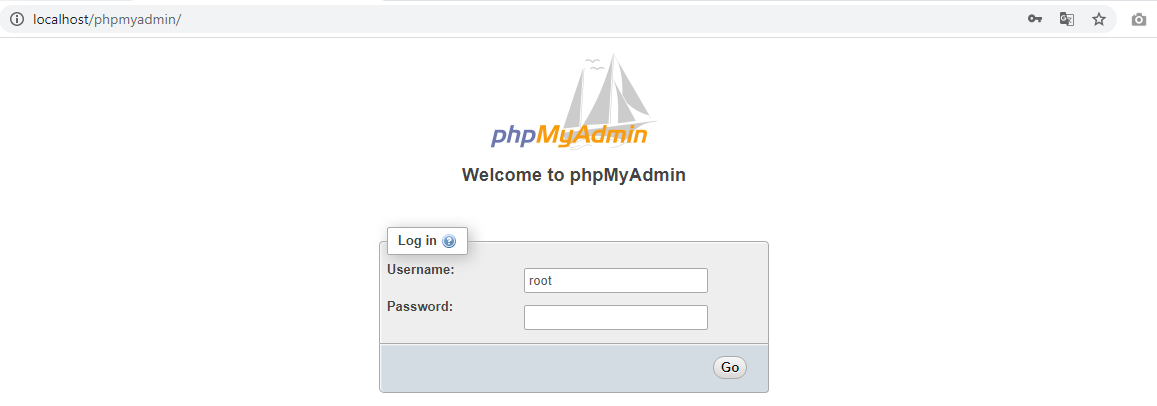
DB\_USERNAME=root

DB\_PASSWORD=

.

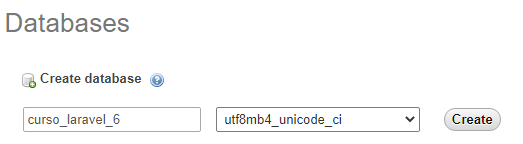
.

.



## Criando o banco de dados

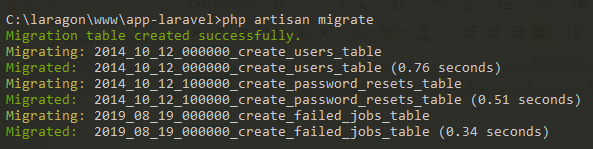
- No PHPMyAdmin:

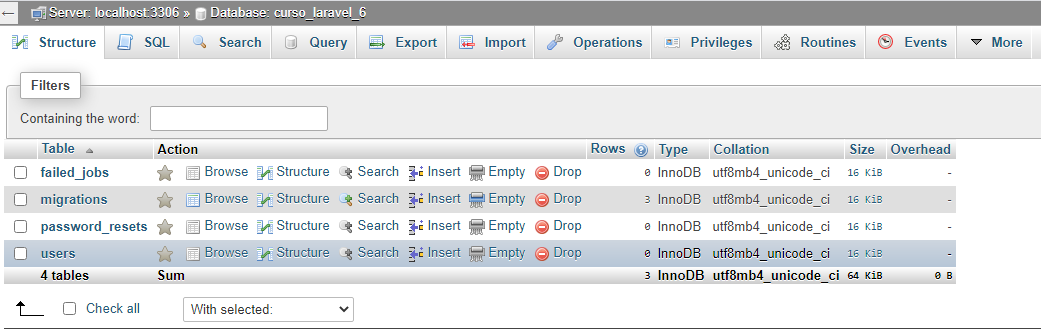


# Aula 38 – Migrations no Laravel 6.x

## Criando as tabelas

php artisan migrate



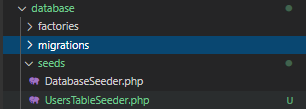


# Aula 39 – Seeders no Laravel 6.x

## Criando um seeder

php artisan make:seeder UsersTableSeeder





## Criando um usuário padrão com seeders

database\seeds\UsersTableSeeder.php

<?php

use Illuminate\Database\Seeder;

use App\User;

class UsersTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

User::create([

'name' => 'Carlos Ferreira',

'email' => 'carlos@especializati.com.br',

'password' => bcrypt('123456')

]);

}

}

database\seeds\DatabaseSeeder.php

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

/\*\*

\* Seed the application's database.

\*

\* @return void

\*/

public function run()

{

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

}

}

app\User.php

<?php

namespace App;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

class User extends Authenticatable

{

use Notifiable;

/\*\*

\* The attributes that are mass assignable.

\*

\* @var array

\*/

protected $fillable = [

'name', 'email', 'password',

];

/\*\*

\* The attributes that should be hidden for arrays.

\*

\* @var array

\*/

protected $hidden = [

'password', 'remember\_token',

];

/\*\*

\* The attributes that should be cast to native types.

\*

\* @var array

\*/

protected $casts = [

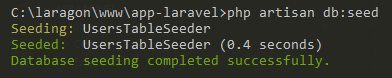
'email\_verified\_at' => 'datetime',

];

}

## Rodando o seeder e criando o usuário

php artisan db:seed





## Rodando apenas um seeder específico

php artisan db:seed --class=ProductsTableSeeder

# Aula 40 – Factory no Laravel 6.x

database\factories\UserFactory.php

<?php

/\*\* @var \Illuminate\Database\Eloquent\Factory $factory \*/

use App\User;

use Faker\Generator as Faker;

use Illuminate\Support\Str;

/\*

|--------------------------------------------------------------------------

| Model Factories

|--------------------------------------------------------------------------

|

| This directory should contain each of the model factory definitions for

| your application. Factories provide a convenient way to generate new

| model instances for testing / seeding your application's database.

|

\*/

$factory->define(User::class, function (Faker $faker) {

return [

'name' => $faker->name,

'email' => $faker->unique()->safeEmail,

'email\_verified\_at' => now(),

// 'password' => '$2y$10$92IXUNpkjO0rOQ5byMi.Ye4oKoEa3Ro9llC/.og/at2.uheWG/igi',

'password' => bcrypt('123456'),

'remember\_token' => Str::random(10),

];

});

## Criando 10 usuários fakers

database\seeds\UsersTableSeeder.php

<?php

use Illuminate\Database\Seeder;

use App\User;

class UsersTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

factory(User::class, 10)->create();

/\*

User::create([

'name' => 'Carlos Ferreira',

'email' => 'carlos@especializati.com.br',

'password' => bcrypt('123456')

]);

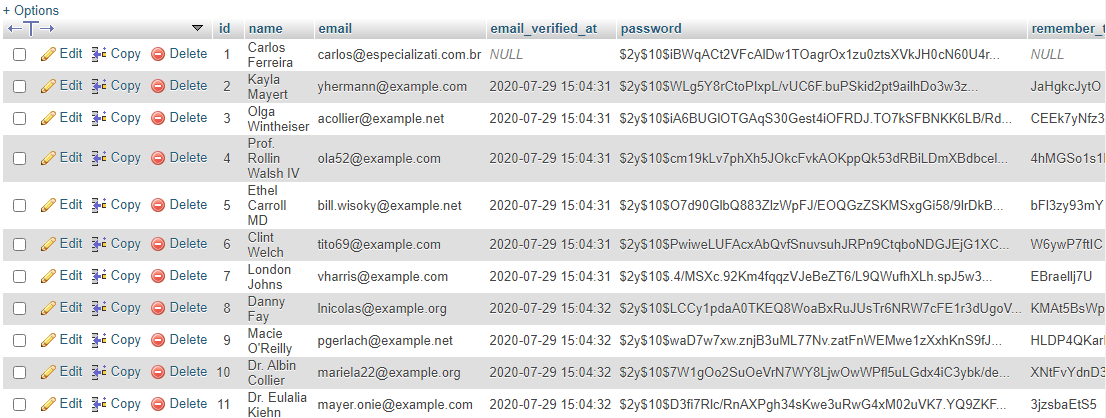
\*/

}

}

php artisan db:seed --class=UsersTableSeeder





## Criando um novo factory

Apenas à título de exemplo, seria:

php artisan make:factory ProductFactory --model=Product

composer dump-upload

# Aula 41 – Introdução ao Eloquent ORM do Laravel

## Criando model e migration simultaneamente

php artisan make:model Product -m



app\Product.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Product extends Model

{

// protected $table = 'products';

}

database\migrations\2020\_07\_30\_000215\_create\_products\_table.php

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateProductsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->bigIncrements('id');

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

$table->double('price', 10, 2);

$table->text('description');

$table->string('image')->nullable();

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('products');

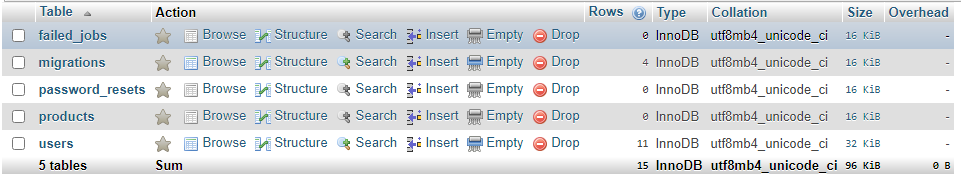
}

}

## Criando a tabela products

php artisan migrate





# Aula 42 – Criando Factory de Produtos no Laravel 6x

## Criando factory de produtos

php artisan make:factory ProductFactory --model=Product



database\factories\ProductFactory.php

<?php

/\*\* @var \Illuminate\Database\Eloquent\Factory $factory \*/

use App\Product;

use Faker\Generator as Faker;

$factory->define(Product::class, function (Faker $faker) {

return [

'name' => $faker->unique()->word,

'description' => $faker->sentence(),

'price' => 12.2

];

});

## Criando seeder de produtos

php artisan make:seeder ProductsTableSeeder



database\seeds\ProductsTableSeeder.php

<?php

use App\Product;

use Illuminate\Database\Seeder;

class ProductsTableSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

factory(Product::class, 100)->create();

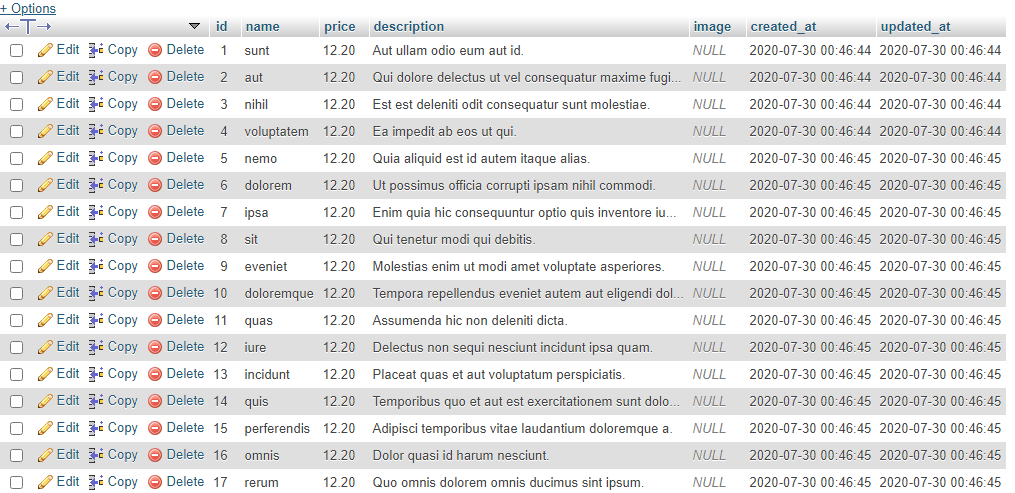
}

}

## Gerando os produtos fakers

php artisan db:seed --class=ProductsTableSeeder





# Aula 43 – Listar/Paginar Registros no Laravel 6x

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use App\Product;

use App\Http\Requests\ProductRequest;

use Illuminate\Http\Request;

public function index()

{

// $products = Product::all();

$products = Product::paginate();

return view('admin.pages.products.index', [

'products' => $products

]);

}

}

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

<a href="{{ route('products.create') }}">Cadastrar</a>

<hr>

<table border="1">

<thead>

<tr>

<th>Nome</th>

<th>Preço</th>

<th>Ações</th>

</tr>

</thead>

<tbody>

@forelse ($products as $product)

<tr>

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

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

<td>

<a href="">Detalhes</a>

</td>

</tr>

@empty

<tr>

<td class="text-leftr"><h5>Não há produtos cadastrados!</h5></td>

<td></td>

<td></td>

</tr>

@endforelse

</tbody>

</table>

<br>

{!! $products->links() !!}

@endsection



# Aula 44 – Incluindo o Bootstrap no Laravel 6x (via CDN)

resources\views\admin\layouts\app.blade.php

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@yield('title') - EspecializaTi</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

@stack('styles')

</head>

<body>

<div class="container">

@yield('content')

</div>

@stack('scripts')

</body>

</html>

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

<a class="btn btn-primary" href="{{ route('products.create') }}">Cadastrar</a>

<hr>

<table class="table table-striped">

<thead>

<tr>

<th>Nome</th>

<th>Preço</th>

<th width='100'>Ações</th>

</tr>

</thead>

<tbody>

@forelse ($products as $product)

<tr>

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

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

<td>

<a href="">Detalhes</a>

</td>

</tr>

@empty

<tr>

<td class="text-leftr"><h5>Não há produtos cadastrados!</h5></td>

<td></td>

<td></td>

</tr>

@endforelse

</tbody>

</table>

<br>

{!! $products->links() !!}

@endsection



# Aula 45 – Listando Detalhes de um registro especifico no Laravel 6x

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use App\Product;

use App\Http\Requests\ProductRequest;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function show($id)

{

// $products = Product::where('id', $id)->first;

if(!$product = Product::find($id)){

return redirect()->back();

}

return view('admin.pages.products.show', [

'product' => $product

]);

}

}

resources\views\admin\pages\products\show.blade.php

@extends('admin.layouts.app')

@section('title', 'Detalhes do produto')

@section('content')

<h1>Produto {{ $product->name }} <a href="{{ route('products.index') }}"><<</a></h1>

<ul>

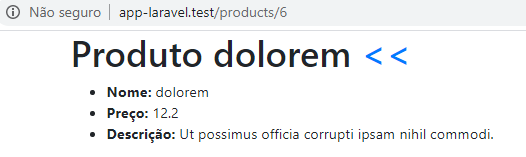
<li><strong>Nome:</strong> {{ $product->name}}</li>

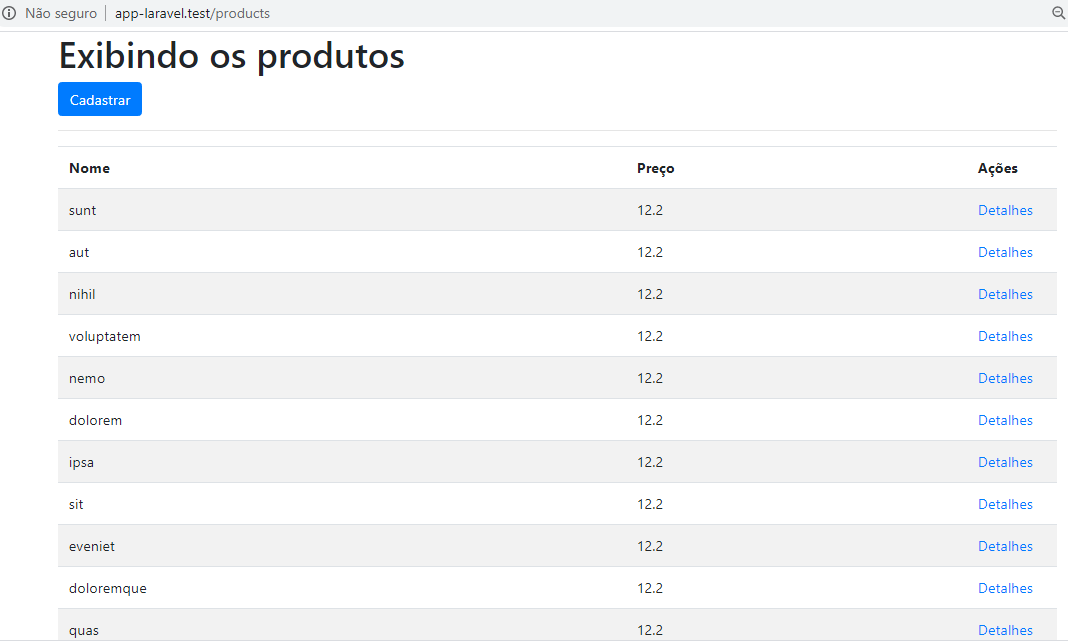
<li><strong>Preço:</strong> {{ $product->price}}</li>

<li><strong>Descrição:</strong> {{ $product->description}}</li>

</ul>

@endsection





# Aula 46 – Inserindo Novo Registro no Banco de Dados com o Laravel 6

resources\views\admin\pages\products\create.blade.php

@extends('admin.layouts.app')

@section('title', 'Cadastrar novo produto')

@section('content')

<h1>Cadastrar novo produto</h1>

@include('admin.includes.alerts')

<form action="{{ route('products.store') }}" method="post" enctype="multipart/form-data">

@csrf

<div class="form-group">

<input class="form-control" type="text" name="name" placeholder="Nome:" value={{ old('name') }}>

</div>

<div class="form-group">

<input class="form-control" type="text" name="price" placeholder="Preço:" value={{ old('price') }}>

</div>

<div class="form-group">

<textarea class="form-control" type="text" name="description" rows="5" placeholder="Descrição:" value={{ old('description') }}></textarea>

</div>

<div class="form-group">

<input class="form-control" type="file" name="image">

</div>

<div class="form-group">

<button class="btn btn-success" type="submit">Enviar</button>

</div>

</form>

@endsection

resources\views\admin\includes\alerts.blade.php

@if ($errors->any())

<ul>

@foreach ($errors->all() as $error)

<li>{{ $error }}</li>

@endforeach

</ul>

@endif

app\Product.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Product extends Model

{

protected $fillable = [

'name', 'description', 'price', 'image'

];

}

app\Http\Requests\ProductRequest.php

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class ProductRequest extends FormRequest

{

public function authorize()

{

return true;

}

public function rules()

{

return [

'name' => 'required|min:3|max:255',

'description' => 'required|min:3|max:10000',

'price' => 'required',

'image' => 'nullable| image'

];

}

}

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use App\Product;

use App\Http\Requests\ProductRequest;

use Illuminate\Http\Request;

class ProductController extends Controller

{

public function index()

{

// $products = Product::all();

$products = Product::latest()->paginate();

return view('admin.pages.products.index', [

'products' => $products

]);

public function create()

{

return view('admin.pages.products.create');

}

public function store(ProductRequest $request)

{

$data = $request->only('name', 'description', 'price');

$product = Product::create($data);

return redirect()->route('products.index');

}

public function show($id)

{

if(!$product = Product::find($id)){

return redirect()->back();

}

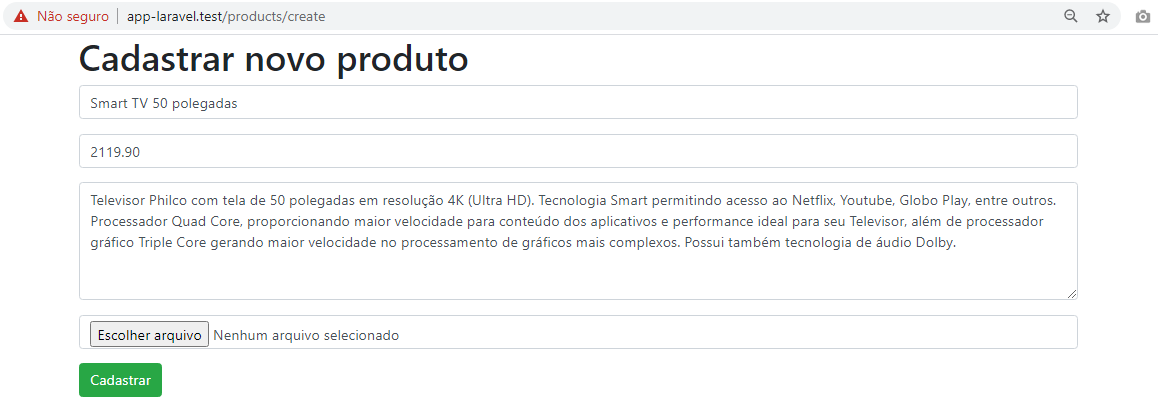
return view('admin.pages.products.show', [

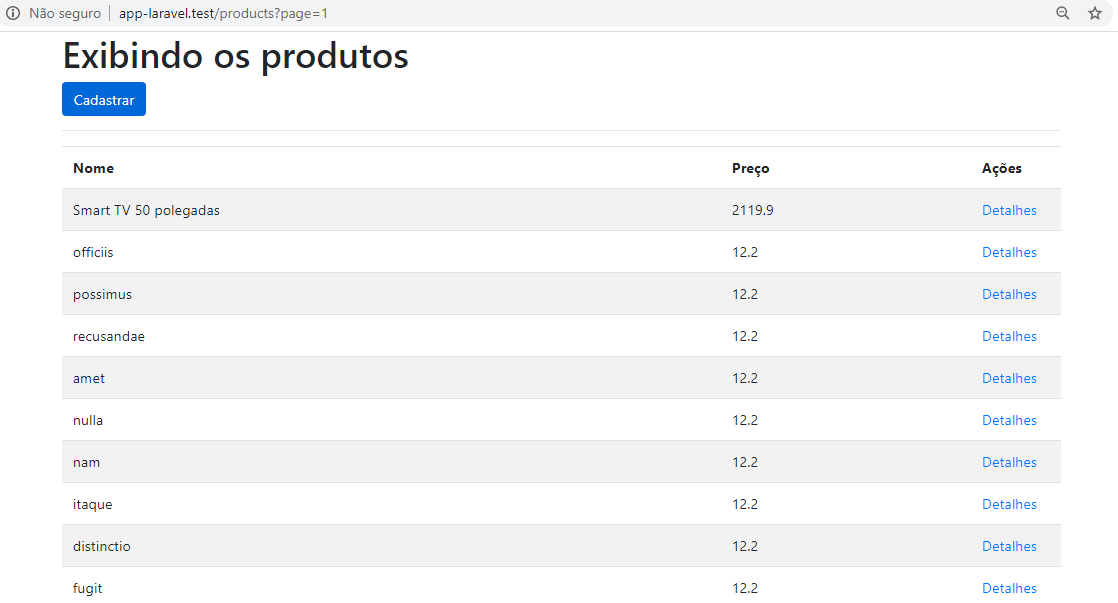
'product' => $product

]);

}

}





# Aula 47 – Deletar um registro no Laravel 6

resources\views\admin\pages\products\show.blade.php

@extends('admin.layouts.app')

@section('title', 'Detalhes do produto')

@section('content')

<h1>Produto {{ $product->name }} <a href="{{ route('products.index') }}"><<</a></h1>

<ul>

<li><strong>Nome:</strong> {{ $product->name}}</li>

<li><strong>Preço:</strong> {{ $product->price}}</li>

<li><strong>Descrição:</strong> {{ $product->description}}</li>

</ul>

<form action="{{ route('products.destroy', $product->id) }}" method="POST">

@csrf

@method('DELETE')

<button class="btn btn-danger" type="submit">Deletar</button>

</form>

@endsection

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use App\Product;

use App\Http\Requests\ProductRequest;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

private $repository;

public function \_\_construct(Request $request, Product $product){

$this->request = $request;

$this->repository = $product;

}

public function index()

{

$products = Product::latest()->paginate();

return view('admin.pages.products.index', [

'products' => $products

]);

}

public function create()

{

return view('admin.pages.products.create');

}

public function store(ProductRequest $request)

{

$data = $request->only('name', 'description', 'price');

$product = $this->repository->create($data);

return redirect()->route('products.index');

}

public function show($id)

{

// $products = Product::where('id', $id)->first;

if(!$product = $this->repository->find($id)){

return redirect()->back();

}

return view('admin.pages.products.show', [

'product' => $product

]);

}

public function destroy($id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

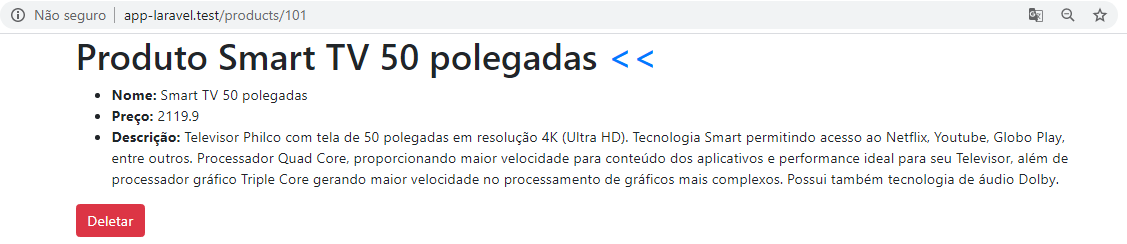
}

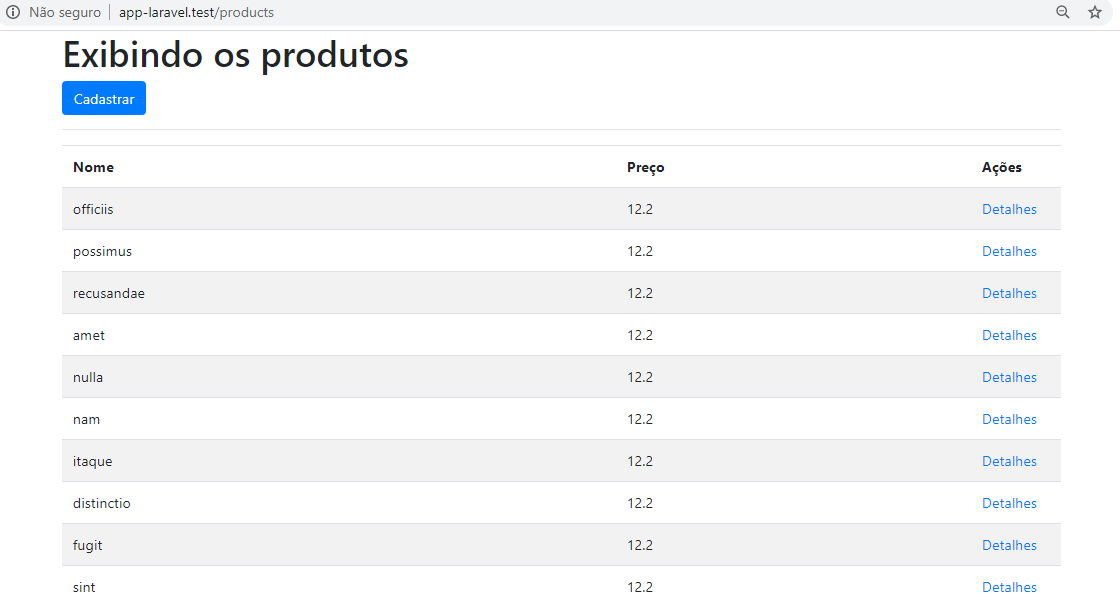
$product->delete();

return redirect()->route('products.index');

}

}





# Aula 48 – Editar um registro no Laravel 6.x

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

<a class="btn btn-primary" href="{{ route('products.create') }}">Cadastrar</a>

<hr>

<table class="table table-striped">

<thead>

<tr>

<th>Nome</th>

<th>Preço</th>

<th width='100'>Ações</th>

</tr>

</thead>

<tbody>

@forelse ($products as $product)

<tr>

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

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

<td>

<a href="{{ route('products.show', $product->id) }}">Detalhes</a>

<a href="{{ route('products.edit', $product->id) }}">Editar</a>

</td>

</tr>

@empty

<tr>

<td class="text-leftr"><h5>Não há produtos cadastrados!</h5></td>

<td></td>

<td></td>

</tr>

@endforelse

</tbody>

</table>

<br>

{!! $products->links() !!}

@endsection

resources\views\admin\pages\products\edit.blade.php

@extends('admin.layouts.app')

@section('title', 'Editar produto {{ $product->name }}')

@section('content')

<h1>Editar produto {{ $product->name }}</h1>

<form action="{{ route('products.update', $product->id) }}" method="post">

@method('PUT')

@include('admin.pages.products.\_partials.form')

</form>

@endsection

resources\views\admin\pages\products\create.blade.php

@extends('admin.layouts.app')

@section('title', 'Cadastrar novo produto')

@section('content')

<h1>Cadastrar novo produto</h1>

<form action="{{ route('products.store') }}" method="post" enctype="multipart/form-data">

@include('admin.pages.products.\_partials.form')

</form>

@endsection

resources\views\admin\pages\products\\_partials\form.blade.php

@include('admin.includes.alerts')

@csrf

<div class="form-group">

<input class="form-control" type="text" name="name" placeholder="Nome:" value={{ $product->name ?? old('name') }}>

</div>

<div class="form-group">

<input class="form-control" type="text" name="price" placeholder="Preço:" value={{ $product->price ?? old('price') }}>

</div>

<div class="form-group">

<textarea class="form-control" type="text" name="description" rows="5" placeholder="Descrição:" >{{ $product->description ?? old('description') }}</textarea>

</div>

<div class="form-group">

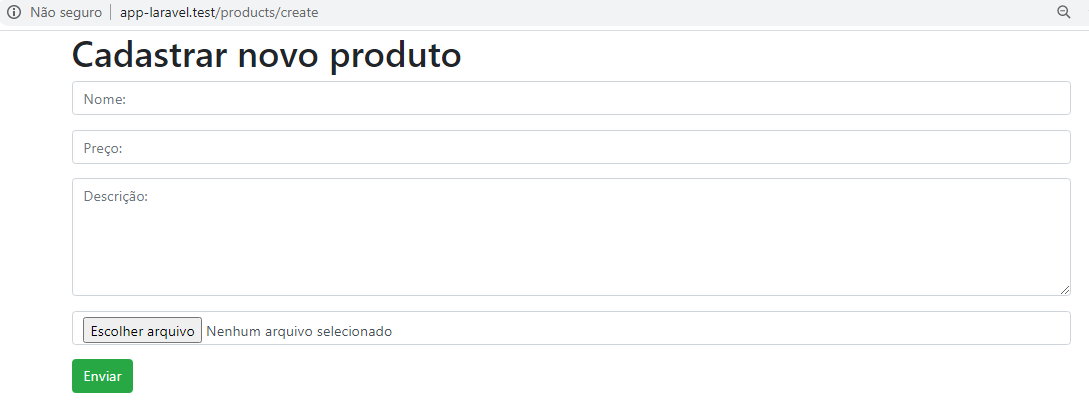
<input class="form-control" type="file" name="image">

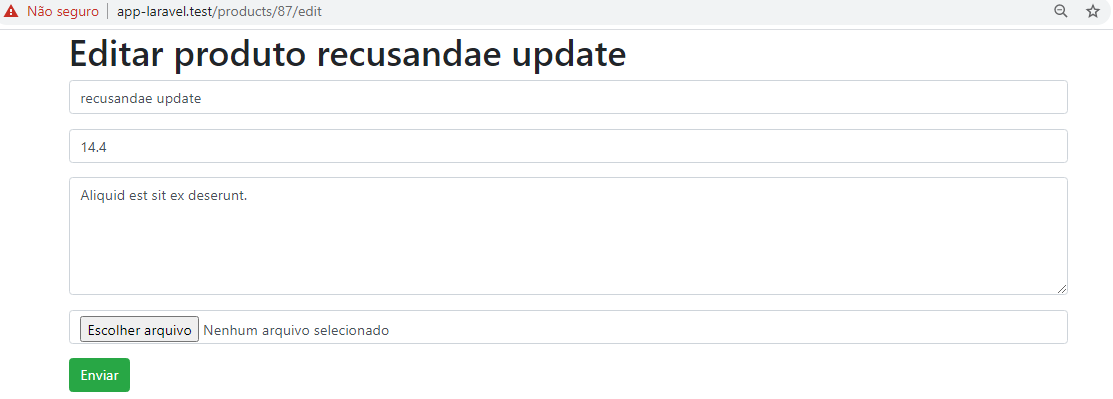
</div>

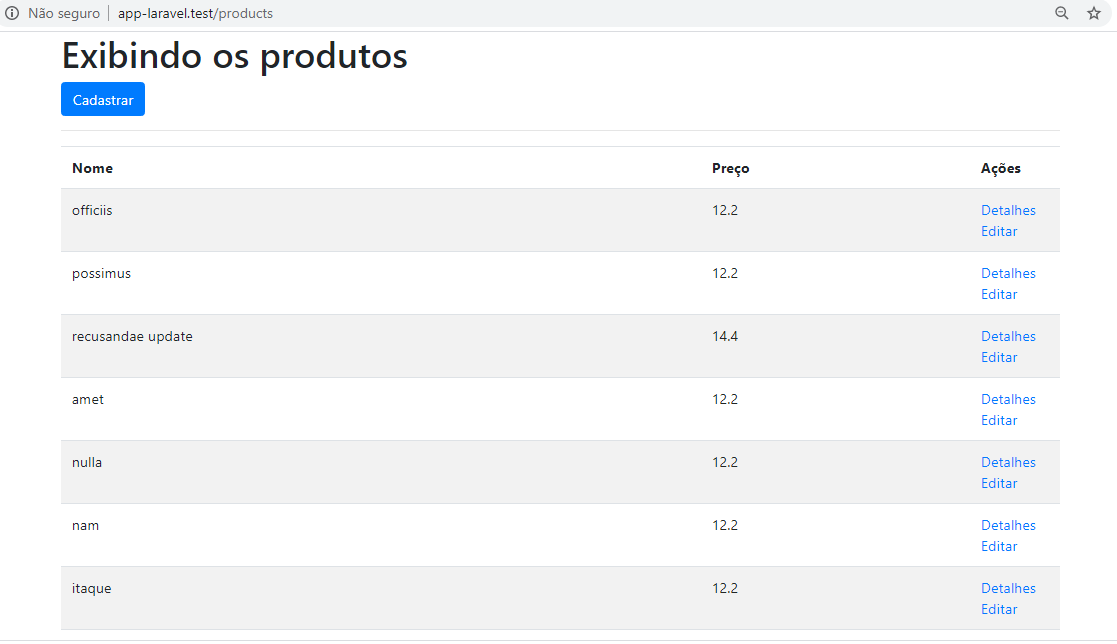
<div class="form-group">

<button class="btn btn-success" type="submit">Enviar</button>

</div>







# Aula 49 – Validações de Produtos no Laravel 6 com Form Request

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use App\Product;

use App\Http\Requests\ProductRequest;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

private $repository;

public function \_\_construct(Request $request, Product $product){

$this->request = $request;

$this->repository = $product;

}

public function index()

{

// $products = Product::all();

$products = Product::latest()->paginate();

return view('admin.pages.products.index', [

'products' => $products

]);

}

public function create()

{

return view('admin.pages.products.create');

}

public function store(ProductRequest $request)

{

$data = $request->only('name', 'description', 'price');

$product = $this->repository->create($data);

return redirect()->route('products.index');

}

public function show($id)

{

// $products = Product::where('id', $id)->first;

if(!$product = $this->repository->find($id)){

return redirect()->back();

}

return view('admin.pages.products.show', [

'product' => $product

]);

}

public function edit($id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

}

return view('admin.pages.products.edit', [

'id' => $id,

'product' => $product

]);

}

public function update(ProductRequest $request, $id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

}

$product->update($request->all());

return redirect()->route('products.index');

}

public function destroy($id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

}

$product->delete();

return redirect()->route('products.index');

}

}

app\Http\Requests\ProductRequest.php

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class ProductRequest extends FormRequest

{

public function authorize()

{

return true;

}

public function rules()

{

$id = $this->segment(2);

return [

'name' => "required|min:3|max:255|unique:products,name,{$id},id",

'description' => 'required|min:3|max:10000',

'price' => "required|regex:/^\d+(\.\d{1,2})?$/",

'image' => 'nullable|image'

];

}

}

resources\views\admin\includes\alerts.blade.php

@if ($errors->any())

<div class="alert alert-warning">

<ul>

@foreach ($errors->all() as $error)

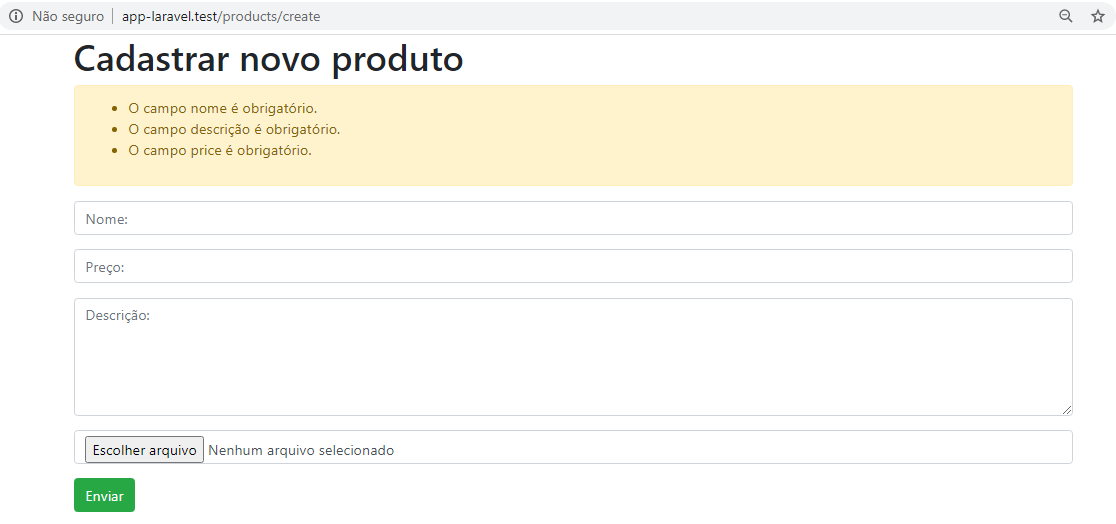
<li>{{ $error }}</li>

@endforeach

</ul>

</div>

@endif



# Aula 50 – Filtrar Registros no Laravel 6x

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

<a class="btn btn-primary" href="{{ route('products.create') }}">Cadastrar</a>

<hr>

<form action="{{ route('products.search') }}" method="post" class="form form-inline">

@csrf

<input type="text" name="filter" class="form-control" value={{ $filters['filter'] ?? '' }}>

<button type="submit" class="btn btn-info">Pesquisar</button>

</form>

<table class="table table-striped">

<thead>

<tr>

<th>Nome</th>

<th>Preço</th>

<th width='100'>Ações</th>

</tr>

</thead>

<tbody>

@forelse ($products as $product)

<tr>

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

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

<td>

<a href="{{ route('products.show', $product->id) }}">Detalhes</a>

<a href="{{ route('products.edit', $product->id) }}">Editar</a>

</td>

</tr>

@empty

<tr>

<td class="text-leftr"><h5>Não há produtos cadastrados!</h5></td>

<td></td>

<td></td>

</tr>

@endforelse

</tbody>

</table>

<br>

@if (isset($filters))

{!! $products->appends($filters)->links() !!}

@else

{!! $products->links() !!}

@endif

routes\web.php

<?php

Route::any('products/search', 'ProductController@search')->name('products.search');

// CRUD

Route::resource('/products', 'ProductController');

app\Product.php

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Product extends Model

{

// protected $table = 'products';

protected $fillable = [

'name', 'description', 'price', 'image'

];

public function search($filter = null){

$results = $this->where(function($query) use($filter){

if($filter){

$query->where('name', 'LIKE', "%{$filter}%");

}

})//->toSql();

->paginate();

// dd($results);

return $results;

}

}

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use App\Product;

use App\Http\Requests\ProductRequest;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

private $repository;

public function \_\_construct(Request $request, Product $product){

$this->request = $request;

$this->repository = $product;

}

public function index()

{

$products = Product::latest()->paginate();

return view('admin.pages.products.index', [

'products' => $products

]);

}

public function create()

{

return view('admin.pages.products.create');

}

public function store(ProductRequest $request)

{

$data = $request->only('name', 'description', 'price');

$product = $this->repository->create($data);

return redirect()->route('products.index');

}

public function show($id)

{

// $products = Product::where('id', $id)->first;

if(!$product = $this->repository->find($id)){

return redirect()->back();

}

return view('admin.pages.products.show', [

'product' => $product

]);

}

public function edit($id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

}

return view('admin.pages.products.edit', [

'id' => $id,

'product' => $product

]);

}

public function update(ProductRequest $request, $id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

}

$product->update($request->all());

return redirect()->route('products.index');

}

public function destroy($id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

}

$product->delete();

return redirect()->route('products.index');

}

public function search(Request $request){

$filters = $request->except('\_token');

$products = $this->repository->search($request->filter);

return view('admin.pages.products.index', [

'products' => $products,

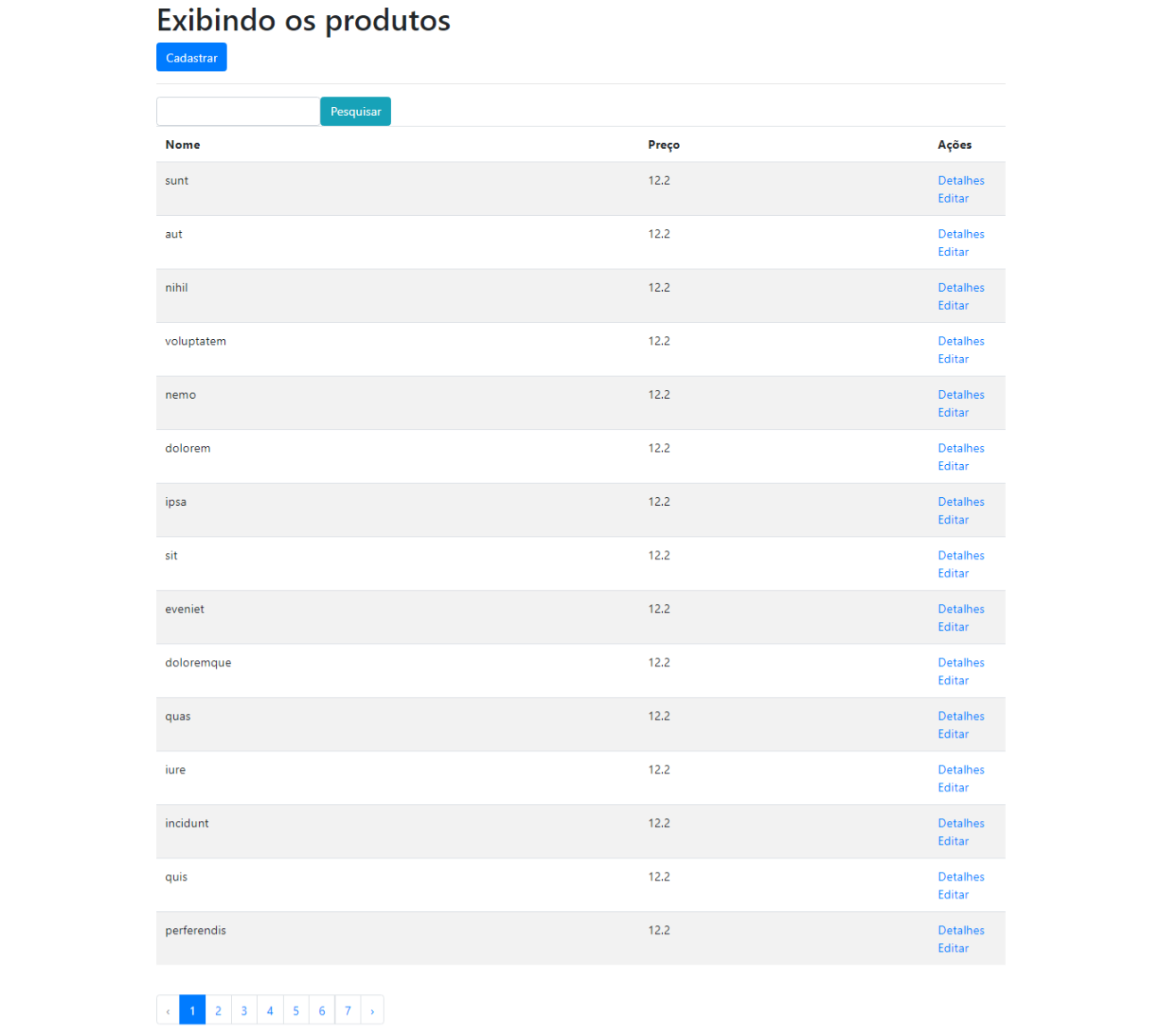
'filters' => $filters

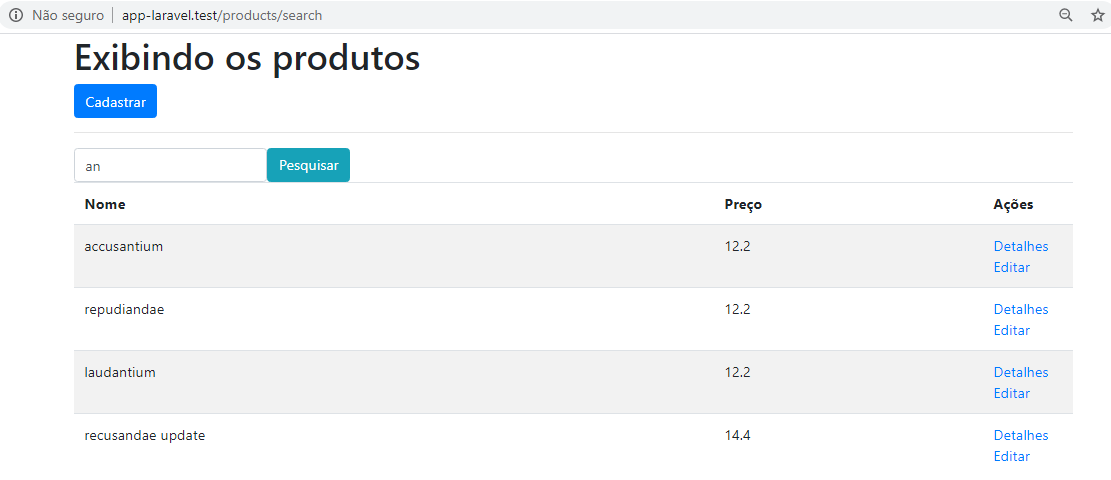
]);

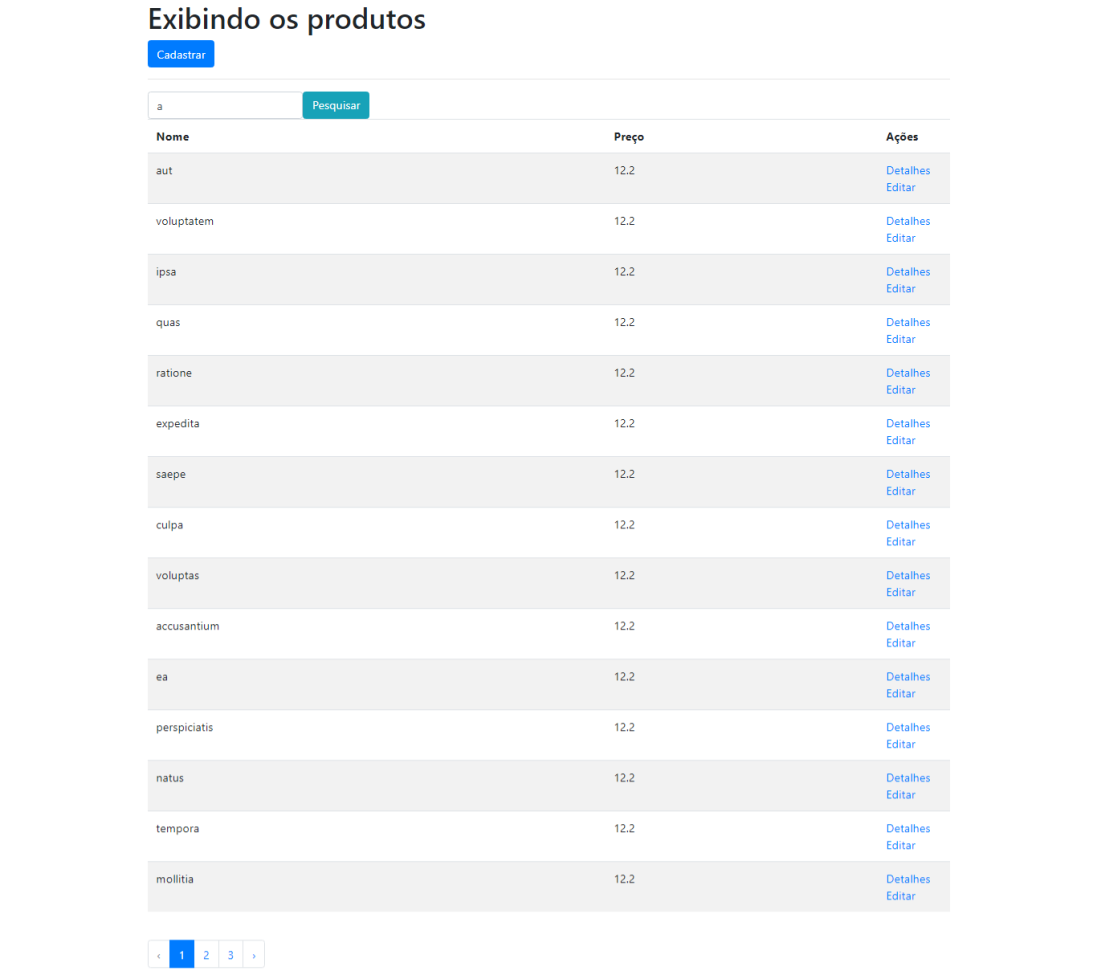
// dd($request->all());

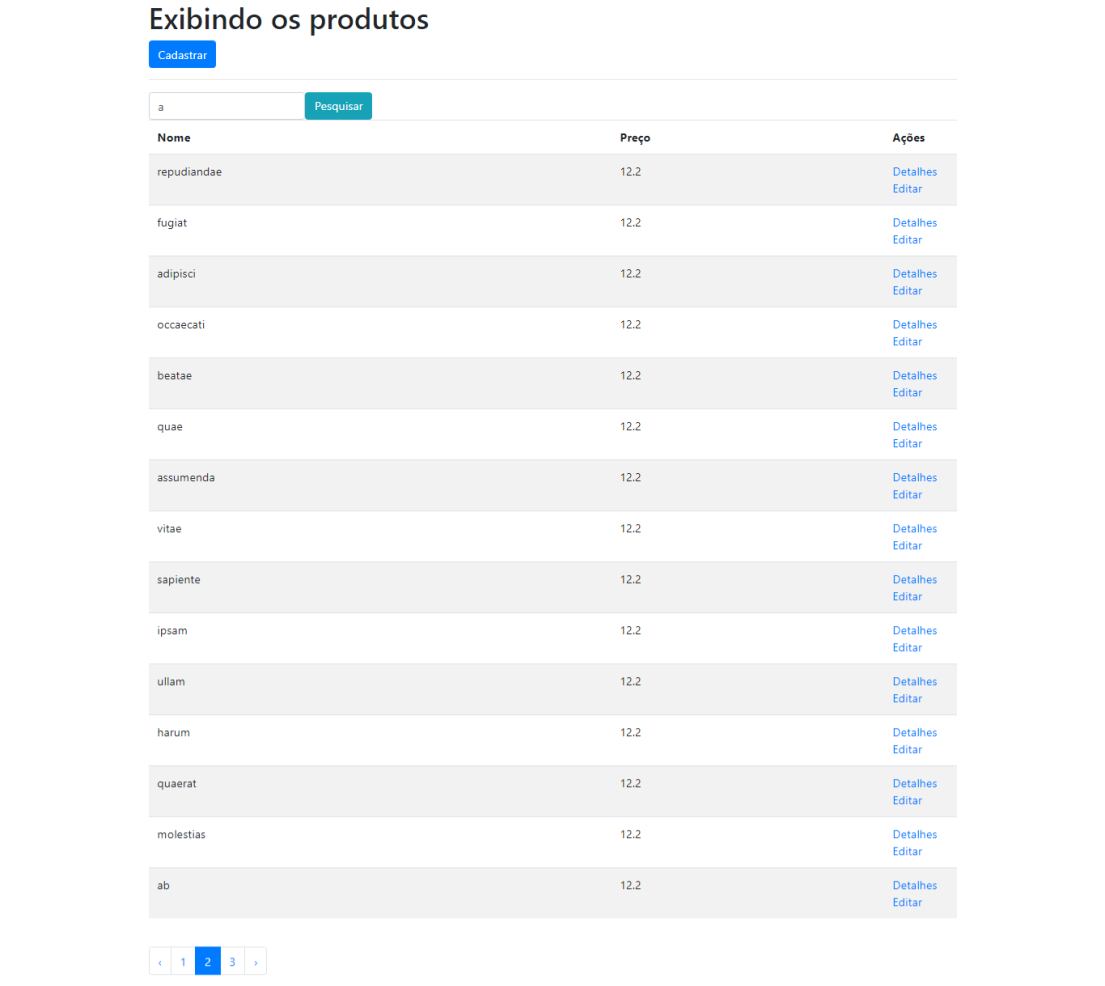
}

}









# Aula 51 – Upload de Imagem para Produto no Laravel

resources\views\admin\pages\products\edit.blade.php

@extends('admin.layouts.app')

@section('title', 'Editar produto {{ $product->name }}')

@section('content')

<h1>Editar produto {{ $product->name }}</h1>

<form action="{{ route('products.update', $product->id) }}" method="post" enctype="multipart/form-data">

@method('PUT')

@include('admin.pages.products.\_partials.form')

</form>

@endsection

resources\views\admin\pages\products\index.blade.php

@extends('admin.layouts.app')

@section('title', 'Gestão de produtos')

@section('content')

<h1>Exibindo os produtos</h1>

<a class="btn btn-primary" href="{{ route('products.create') }}">Cadastrar</a>

<hr>

<form action="{{ route('products.search') }}" method="post" class="form form-inline">

@csrf

<input type="text" name="filter" class="form-control" value={{ $filters['filter'] ?? '' }}>

<button type="submit" class="btn btn-info">Pesquisar</button>

</form>

<table class="table table-striped">

<thead>

<tr>

<th width="100">Imagem</th>

<th>Nome</th>

<th>Preço</th>

<th width='100'>Ações</th>

</tr>

</thead>

<tbody>

@forelse ($products as $product)

<tr>

<td>

@if ($product->image)

<img src="{{ url("storage/{$product->image}") }}" alt="{{ $product->name }}" style="max-width: 100px;">

@endif

</td>

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

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

<td>

<a href="{{ route('products.show', $product->id) }}">Detalhes</a>

<a href="{{ route('products.edit', $product->id) }}">Editar</a>

</td>

</tr>

@empty

<tr>

<td class="text-leftr"><h5>Não há produtos cadastrados!</h5></td>

<td></td>

<td></td>

</tr>

@endforelse

</tbody>

</table>

<br>

@if (isset($filters))

{!! $products->appends($filters)->links() !!}

@else

{!! $products->links() !!}

@endif

app\Http\Controllers\ProductController.php

<?php

namespace App\Http\Controllers;

use App\Product;

use App\Http\Requests\ProductRequest;

use Illuminate\Http\Request;

class ProductController extends Controller

{

protected $request;

private $repository;

public function \_\_construct(Request $request, Product $product){

$this->request = $request;

$this->repository = $product;

}

public function index()

{

// $products = Product::all();

$products = Product::latest()->paginate();

return view('admin.pages.products.index', [

'products' => $products

]);

}

public function create()

{

return view('admin.pages.products.create');

}

public function store(ProductRequest $request)

{

$data = $request->only('name', 'description', 'price');

if ($request->hasFile('image') && $request->image->isValid()) {

$imagePath = $request->image->store('products');

$data['image'] = $imagePath;

}

$product = $this->repository->create($data);

return redirect()->route('products.index');

}

public function show($id)

{

// $products = Product::where('id', $id)->first;

if(!$product = $this->repository->find($id)){

return redirect()->back();

}

return view('admin.pages.products.show', [

'product' => $product

]);

// return "Detalhes do produto de id {$id}";

}

public function edit($id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

}

return view('admin.pages.products.edit', [

'id' => $id,

'product' => $product

]);

}

public function update(ProductRequest $request, $id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

}

$data = $request->all();

if ($request->hasFile('image') && $request->image->isValid()) {

if ($product->image && Storage::exists($product->image)) {

Storage::delete($product->image);

}

$imagePath = $request->image->store('products');

$data['image'] = $imagePath;

}

$product->update($data);

return redirect()->route('products.index');

}

public function destroy($id)

{

$product = $this->repository->where('id', $id)->first();

if(!$product){

return redirect()->back();

}

if ($product->image && Storage::exists($product->image)) {

Storage::delete($product->image);

}

$product->delete();

return redirect()->route('products.index');

}

public function search(Request $request){

$filters = $request->except('\_token');

$products = $this->repository->search($request->filter);

return view('admin.pages.products.index', [

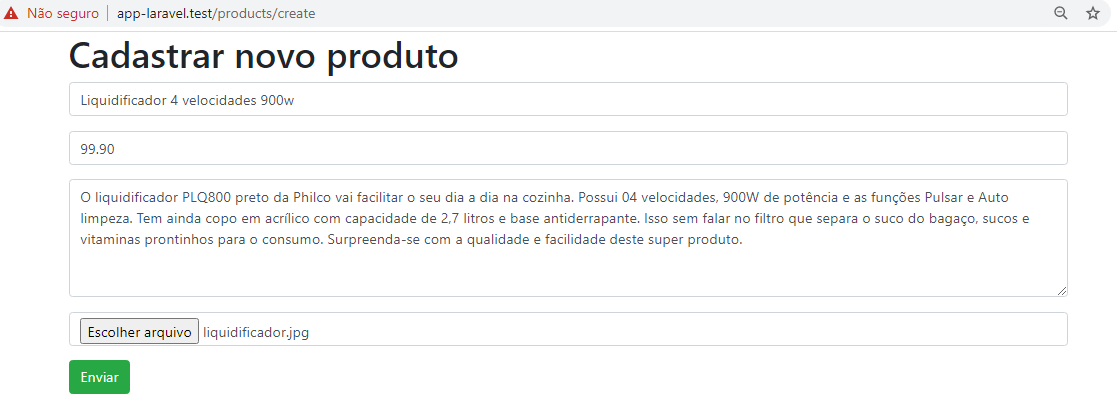
'products' => $products,

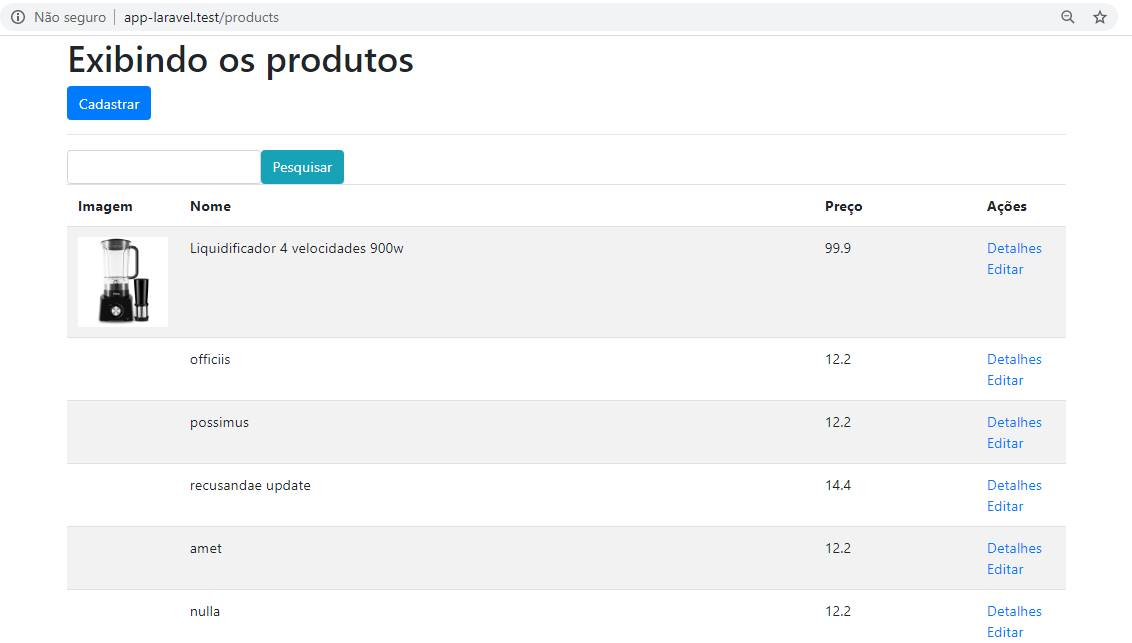
'filters' => $filters

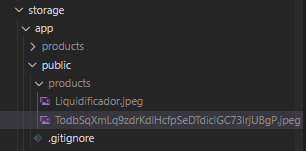
]);

}

}







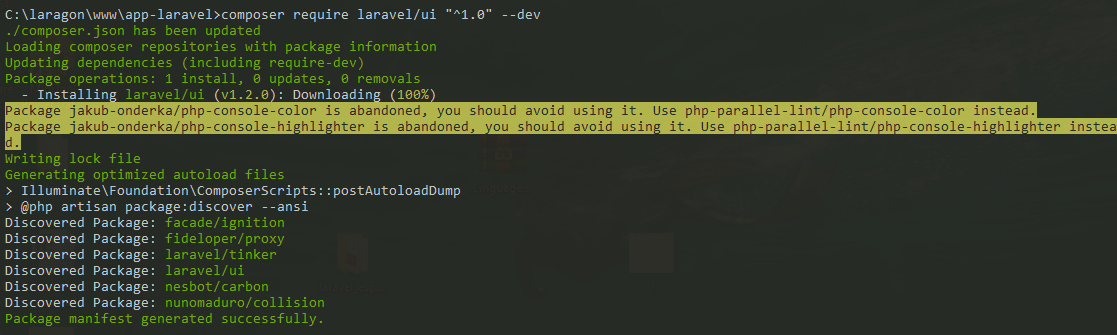


# Aula 52 – Autenticação no Laravel (5, 6 e 7)

## Fazendo a instalação do pacote de autenticação do Laravel

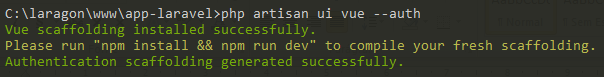
https://laravel.com/docs/6.x/authentication

composer require laravel/ui "^1.0" --dev



## Gerando o scaffold

php artisan ui vue --auth



## Rotas

routes\web.php

<?php

Route::any('products/search', 'ProductController@search')->name('products.search');

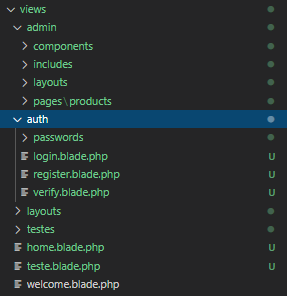
// CRUD

Route::resource('/products', 'ProductController');

Auth::routes();

Route::get('/home', 'HomeController@index')->name('home');

## Views



resources\views\welcome.blade.php

<!DOCTYPE html>

<html lang="{{ str\_replace('\_', '-', app()->getLocale()) }}">

<head>

<meta charset="utf-8">

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

<title>Laravel</title>

<!-- Fonts -->

<link href="https://fonts.googleapis.com/css?family=Nunito:200,600" rel="stylesheet">

<!-- Styles -->

<style>

html, body {

background-color: #fff;

color: #636b6f;

font-family: 'Nunito', sans-serif;

font-weight: 200;

height: 100vh;

margin: 0;

}

.full-height {

height: 100vh;

}

.flex-center {

align-items: center;

display: flex;

justify-content: center;

}

.position-ref {

position: relative;

}

.top-right {

position: absolute;

right: 10px;

top: 18px;

}

.content {

text-align: center;

}

.title {

font-size: 84px;

}

.links > a {

color: #636b6f;

padding: 0 25px;

font-size: 13px;

font-weight: 600;

letter-spacing: .1rem;

text-decoration: none;

text-transform: uppercase;

}

.m-b-md {

margin-bottom: 30px;

}

</style>

</head>

<body>

<div class="flex-center position-ref full-height">

@if (Route::has('login'))

<div class="top-right links">

@auth

<a href="{{ url('/home') }}">Home</a>

@else

<a href="{{ route('login') }}">Login</a>

@if (Route::has('register'))

<a href="{{ route('register') }}">Register</a>

@endif

@endauth

</div>

@endif

<div class="content">

<div class="title m-b-md">

Laravel

</div>

<div class="links">

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

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

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

<a href="https://blog.laravel.com">Blog</a>

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

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

<a href="https://vapor.laravel.com">Vapor</a>

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

</div>

</div>

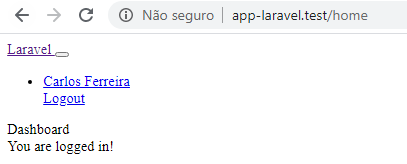
</div>

</body>

</html>







## Desabilitando o register e aplicando a autenticação

routes\web.php

<?php

Route::any('products/search', 'ProductController@search')->name('products.search')->middleware('auth');

// CRUD

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

Auth::routes(['register' => false]);

Route::get('/home', 'HomeController@index')->name('home');



## Redirecionando para a página de entrada

app\Providers\RouteServiceProvider.php

<?php

namespace App\Providers;

use Illuminate\Foundation\Support\Providers\RouteServiceProvider as ServiceProvider;

use Illuminate\Support\Facades\Route;

class RouteServiceProvider extends ServiceProvider

{

protected $namespace = 'App\Http\Controllers';

// public const HOME = '/home';

public const HOME = '/products';

public function boot()

{

//

parent::boot();

}

public function map()

{

$this->mapApiRoutes();

$this->mapWebRoutes();

//

}

protected function mapWebRoutes()

{

Route::middleware('web')

->namespace($this->namespace)

->group(base\_path('routes/web.php'));

}

protected function mapApiRoutes()

{

Route::prefix('api')

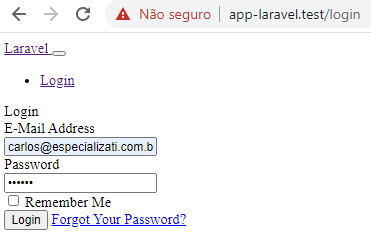
->middleware('api')

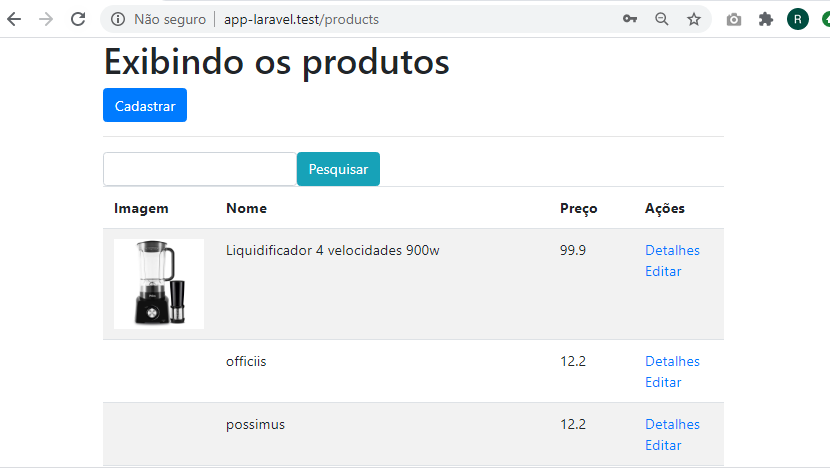
->namespace($this->namespace)

->group(base\_path('routes/api.php'));

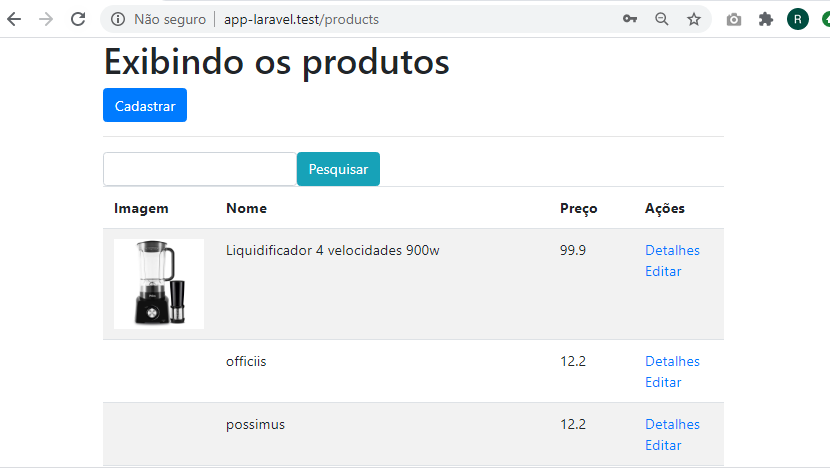
}

}









# Aula 53 – Compilar Assets com o Laravel Mix

- Baixar o node (versão mais atual), para poder usar o npm.

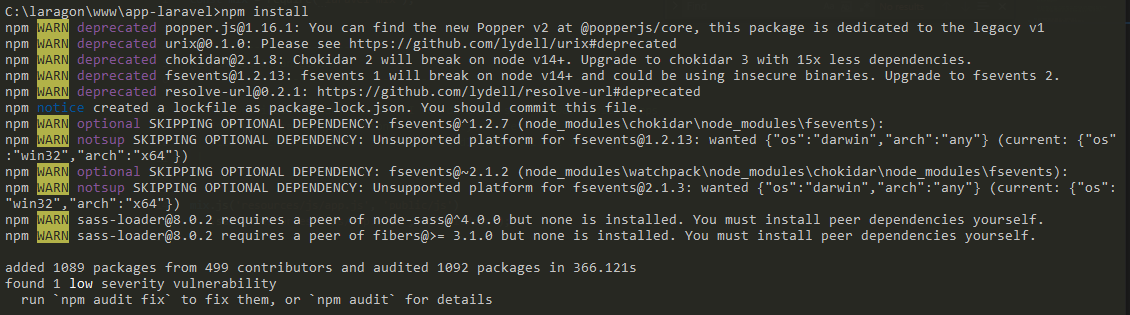
npm --version

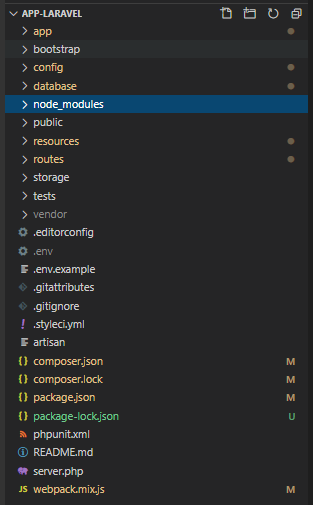


## Instalando as dependências

Execute o comando a seguir. Será criada a pasta node\_modules com todas as dependências.

npm install





webpack.mix.js

const mix = require('laravel-mix');

/\*

|--------------------------------------------------------------------------

| Mix Asset Management

|--------------------------------------------------------------------------

|

| Mix provides a clean, fluent API for defining some Webpack build steps

| for your Laravel application. By default, we are compiling the Sass

| file for the application as well as bundling up all the JS files.

|

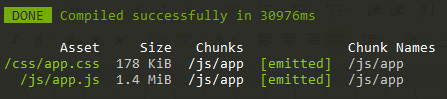
\*/

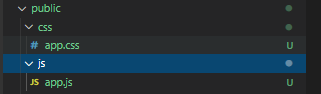
mix.js('resources/js/app.js', 'public/js')

.sass('resources/sass/app.scss', 'public/css');

## Compilando

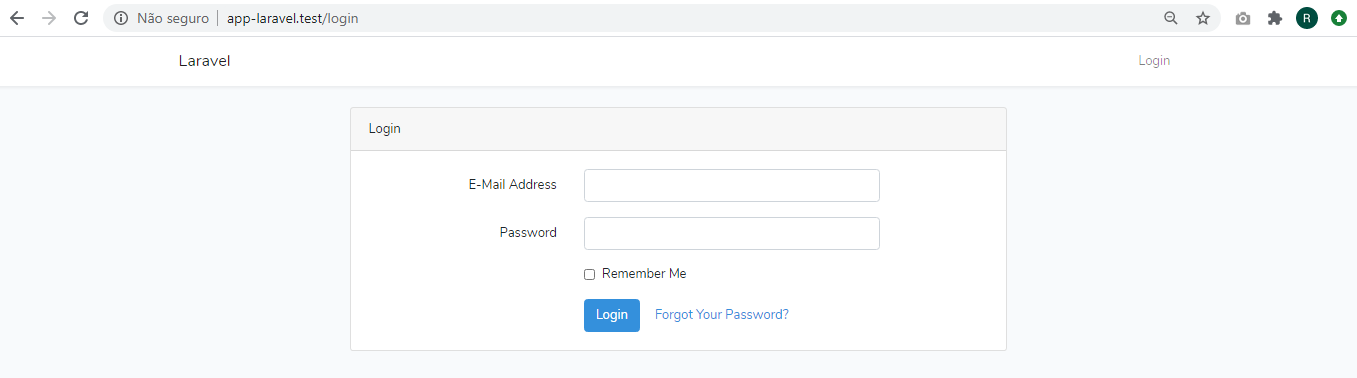
npm run dev

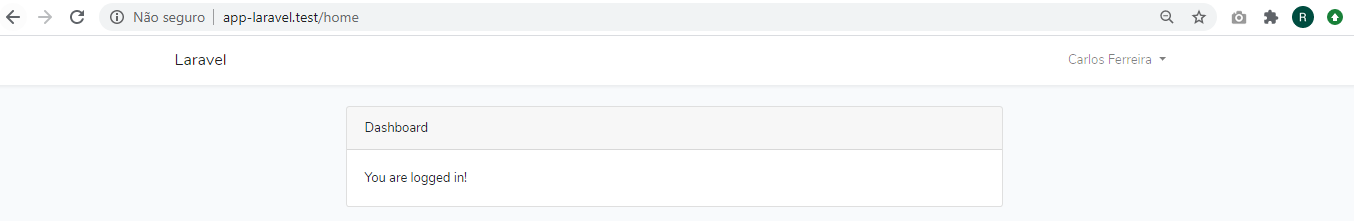




## Compilando em tempo real

npm run dev --watch





# Aula 54 – Finalizando a autenticação no Laravel

- Remova a rota "/home"

- Remova o controller "HomeController"

- Remova a view "home.blade.php".

## Incluindo uma sidebar no template

resources\views\layouts\app.blade.php

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>@yield('title') - EspecializaTi</title>

{{-- <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"> --}}

<!-- Scripts -->

<script src="{{ asset('js/app.js') }}" defer></script>

<!-- Fonts -->

<link rel="dns-prefetch" href="//fonts.gstatic.com">

<link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet">

<!-- Styles -->

<link href="{{ asset('css/app.css') }}" rel="stylesheet">

@stack('styles')

</head>

<body>

<nav class="navbar navbar-expand-md navbar-light bg-white shadow-sm">

<div class="container">

<a class="navbar-brand" href="{{ url('/') }}">

{{ config('app.name', 'Laravel') }}

</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="{{ \_\_('Toggle navigation') }}">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<!-- Left Side Of Navbar -->

<ul class="navbar-nav mr-auto">

</ul>

<!-- Right Side Of Navbar -->

<ul class="navbar-nav ml-auto">

<!-- Authentication Links -->

@guest

<li class="nav-item">

<a class="nav-link" href="{{ route('login') }}">{{ \_\_('Login') }}</a>

</li>

@if (Route::has('register'))

<li class="nav-item">

<a class="nav-link" href="{{ route('register') }}">{{ \_\_('Register') }}</a>

</li>

@endif

@else

<li class="nav-item dropdown">

<a id="navbarDropdown" class="nav-link dropdown-toggle" href="#" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" v-pre>

{{ Auth::user()->name }} <span class="caret"></span>

</a>

<div class="dropdown-menu dropdown-menu-right" aria-labelledby="navbarDropdown">

<a class="dropdown-item" href="{{ route('logout') }}"

onclick="event.preventDefault();

document.getElementById('logout-form').submit();">

{{ \_\_('Logout') }}

</a>

<form id="logout-form" action="{{ route('logout') }}" method="POST" style="display: none;">

@csrf

</form>

</div>

</li>

@endguest

</ul>

</div>

</div>

</nav>

<div class="container">

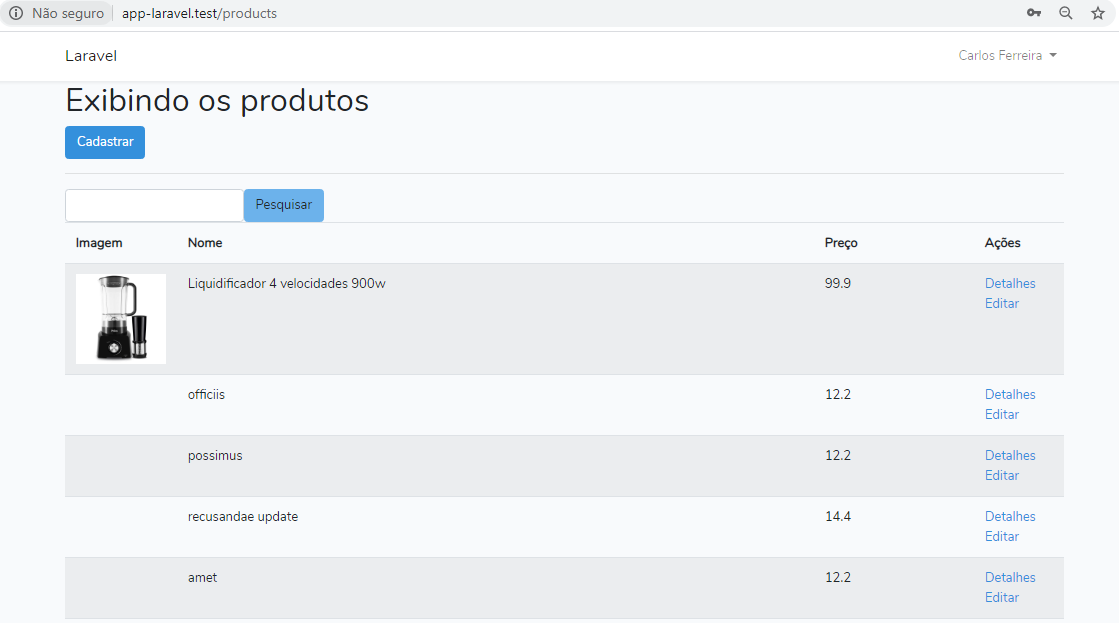
@yield('content')

</div>

@stack('scripts')

</body>

</html>



# Aula 55 – Middlewares (filtros) no Laravel

Middlewares são filtros no Laravel.

As definições de middlewares ficam no arquivo "app\Http\Kernel.php".

## Criando um middleware

php artisan make:middleware CheckIsAdminMiddleware



## Registrando o middleware

app\Http\Kernel.php

<?php

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

class Kernel extends HttpKernel

{

/\*\*

\* The application's global HTTP middleware stack.

\*

\* These middleware are run during every request to your application.

\*

\* @var array

\*/

protected $middleware = [

\App\Http\Middleware\TrustProxies::class,

\App\Http\Middleware\CheckForMaintenanceMode::class,

\Illuminate\Foundation\Http\Middleware\ValidatePostSize::class,

\App\Http\Middleware\TrimStrings::class,

\Illuminate\Foundation\Http\Middleware\ConvertEmptyStringsToNull::class,

];

/\*\*

\* The application's route middleware groups.

\*

\* @var array

\*/

protected $middlewareGroups = [

'web' => [

\App\Http\Middleware\EncryptCookies::class,

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,

\Illuminate\Session\Middleware\StartSession::class,

// \Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\App\Http\Middleware\VerifyCsrfToken::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,

],

'api' => [

'throttle:60,1',

\Illuminate\Routing\Middleware\SubstituteBindings::class,

],

];

/\*\*

\* The application's route middleware.

\*

\* These middleware may be assigned to groups or used individually.

\*

\* @var array

\*/

protected $routeMiddleware = [

'auth' => \App\Http\Middleware\Authenticate::class,

'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,

'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,

'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,

'can' => \Illuminate\Auth\Middleware\Authorize::class,

'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,

'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,

'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,

'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,

'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,

'check.is.admin' => \App\Http\Middleware\CheckIsAdminMiddleware::class

];

/\*\*

\* The priority-sorted list of middleware.

\*

\* This forces non-global middleware to always be in the given order.

\*

\* @var array

\*/

protected $middlewarePriority = [

\Illuminate\Session\Middleware\StartSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\App\Http\Middleware\Authenticate::class,

\Illuminate\Routing\Middleware\ThrottleRequests::class,

\Illuminate\Session\Middleware\AuthenticateSession::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,

\Illuminate\Auth\Middleware\Authorize::class,

];

}

routes\web.php

<?php

Route::any('products/search', 'ProductController@search')->name('products.search')->middleware('auth');

// CRUD

Route::resource('/products', 'ProductController')->middleware(['auth', 'check.is.admin']);

Auth::routes(['register' => false]);

Route::get('/home', 'HomeController@index')->name('home');

app\Http\Middleware\CheckIsAdminMiddleware.php

<?php

namespace App\Http\Middleware;

use Closure;

class CheckIsAdminMiddleware

{

/\*\*

\* Handle an incoming request.

\*

\* @param \Illuminate\Http\Request $request

\* @param \Closure $next

\* @return mixed

\*/

public function handle($request, Closure $next)

{

// dd('Estou aqui');

$user = auth()->user();

if (!in\_array($user->email, ['carlos@especializati.com.br', 'yhermann@example.com'])) {

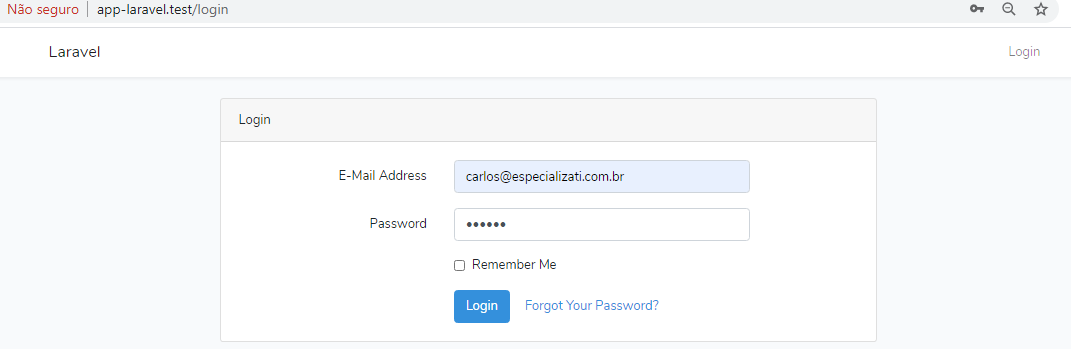
return redirect('/');

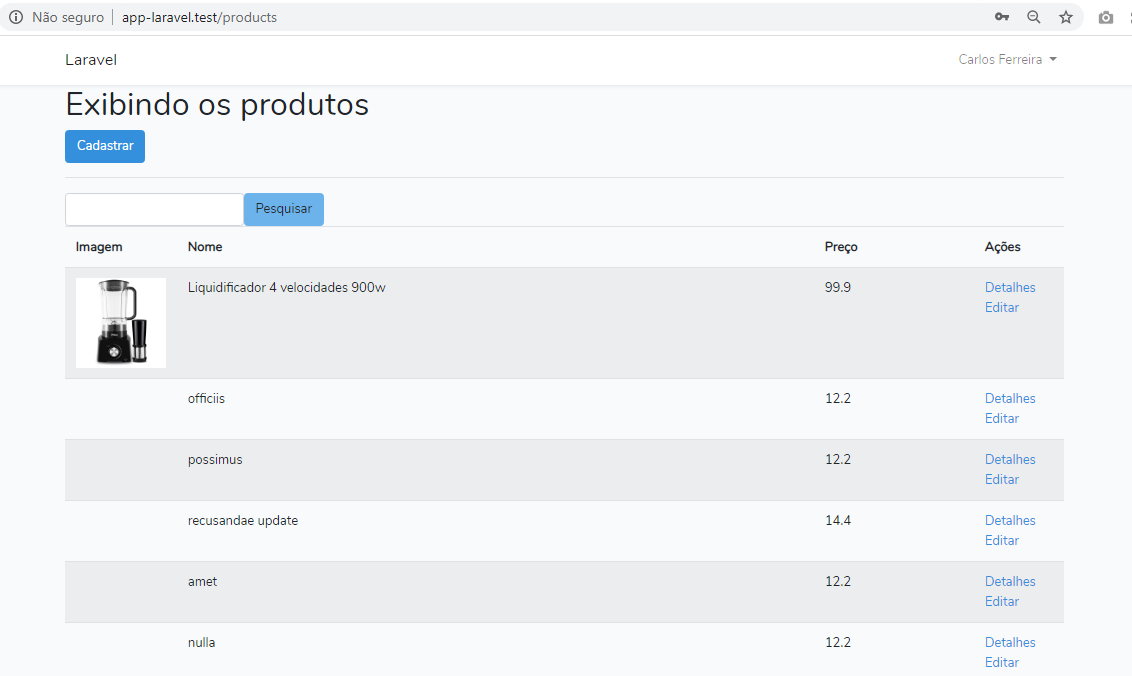
}

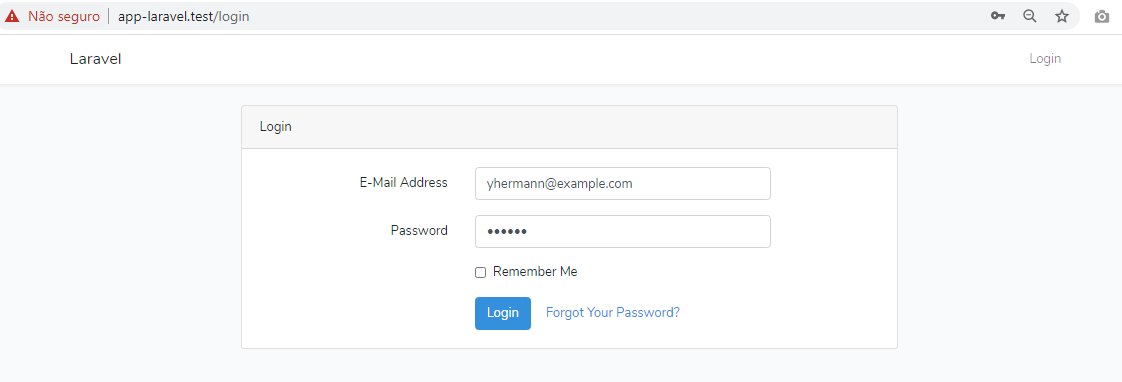
return $next($request);

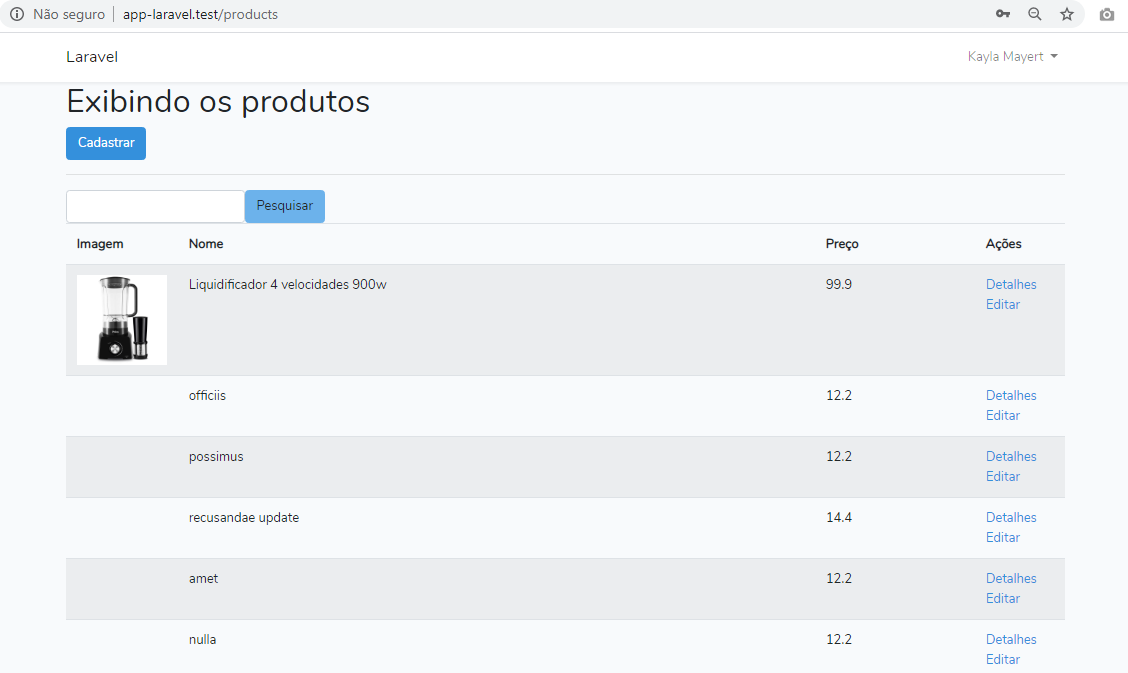
}

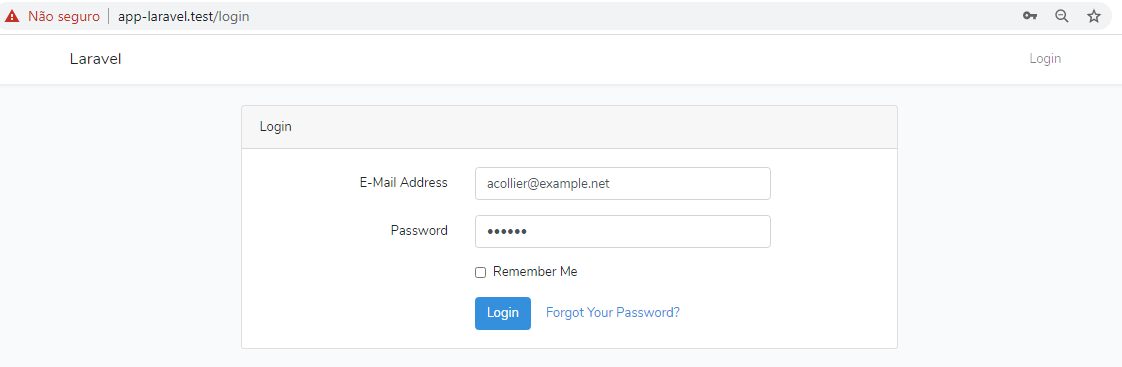
}

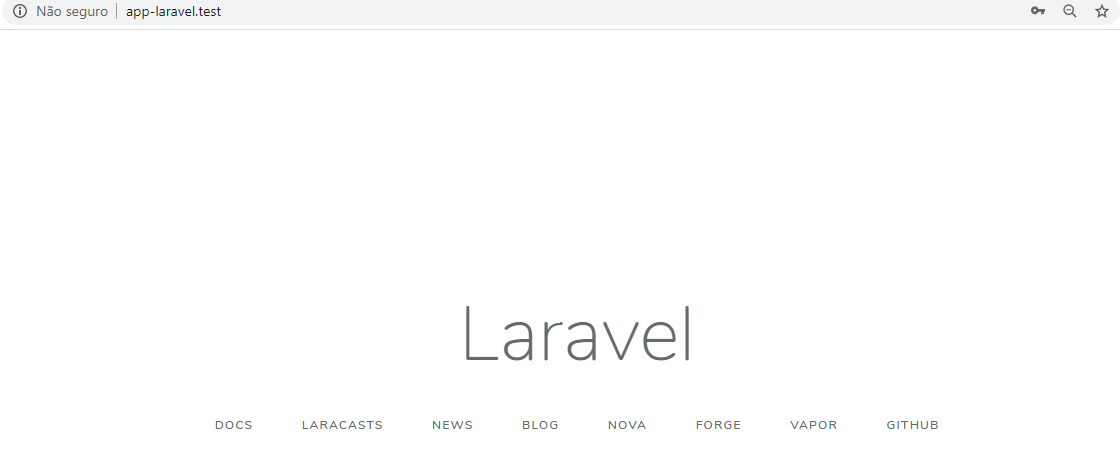












<https://academy.especializati.com.br/aula/ferramentas-para-trabalhar-com-laravel-6>