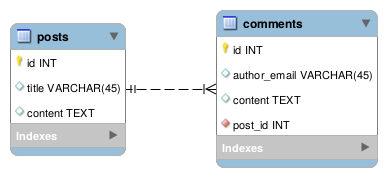
**LARAVEL 5.6**

**USP dev (Leandro Ramos)**

<https://www.youtube.com/watch?v=lO_udEy9I4U&list=PLIFOx3X8xDut2X2ydevlVvFKkqFxZ4SX9>

Resumo do curso feito por Roberto Pinheiro

# Aula 01 - hasMany Relationships e Tinker



## Configurando o acesso ao banco de dados

DB\_CONNECTION=mysql

DB\_HOST=localhost

DB\_PORT=3306

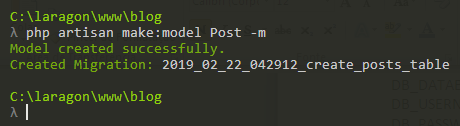
DB\_DATABASE=blog

DB\_USERNAME=root

DB\_PASSWORD=

## Criando os models e migrations

php artisan make:model Post -m



**database/migrations/2019\_02\_22\_042912\_create\_posts\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreatePostsTable extends Migration

{

public function up()

{

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

$table->increments('id');

$table->string('title');

$table->text('content');

$table->timestamps();

});

}

public function down()

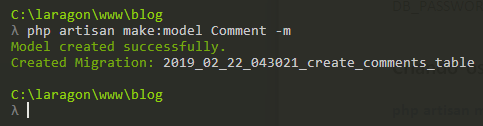
{

Schema::dropIfExists('posts');

}

}

php artisan make:model Comment -m



**database/migrations/2019\_02\_22\_043021\_create\_comments\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateCommentsTable extends Migration

{

public function up()

{

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

$table->increments('id');

$table->string('author\_email');

$table->text('content');

$table->integer('post\_id')->unsigned();

$table->timestamps();

$table->foreign('post\_id')->references('id')->on('posts');

});

}

public function down()

{

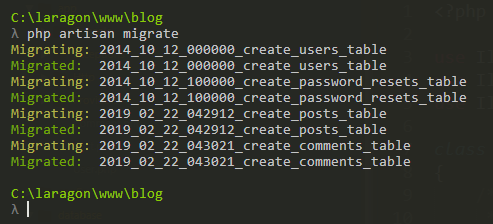
Schema::dropIfExists('comments');

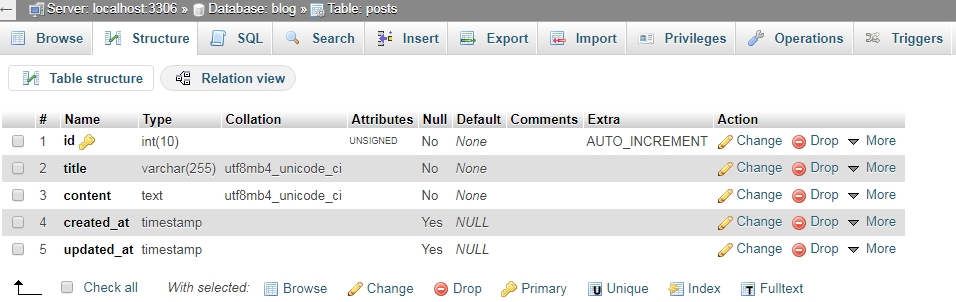
}

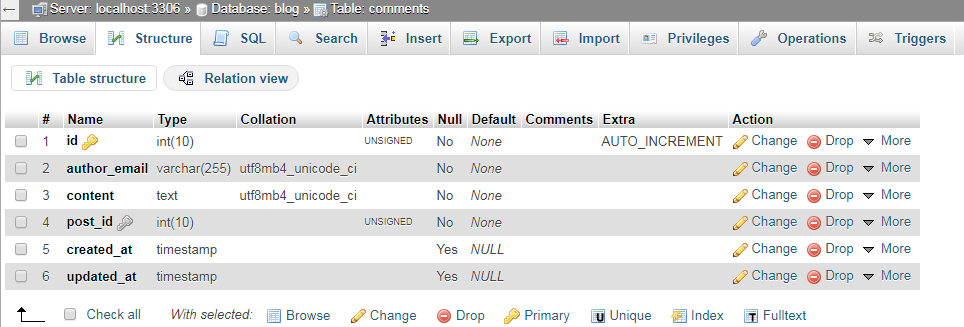
}

## Criando as tabelas

php artisan migrate







**app/Post.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Post extends Model

{

public function comments()

{

// Um post tem vários comentários

return $this->hasMany('App\Comment');

}

}

**app/Comment.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Comment extends Model

{

public function post()

{

// Um comentário pertence a um post

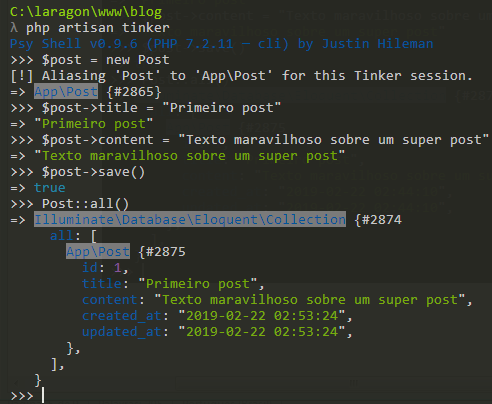
return $this->belongsTo('App\Post');

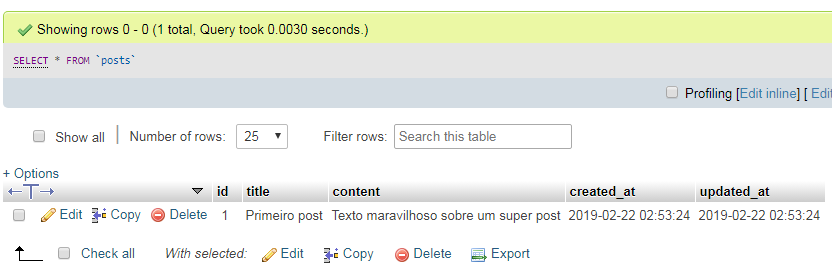
}

}

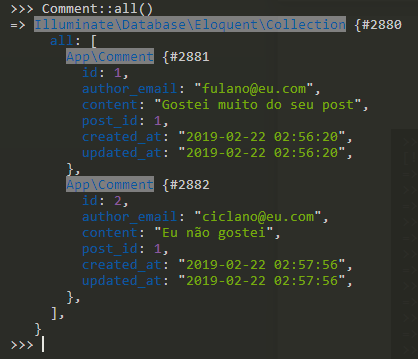
## Testando os relacionamentos com o Tinker

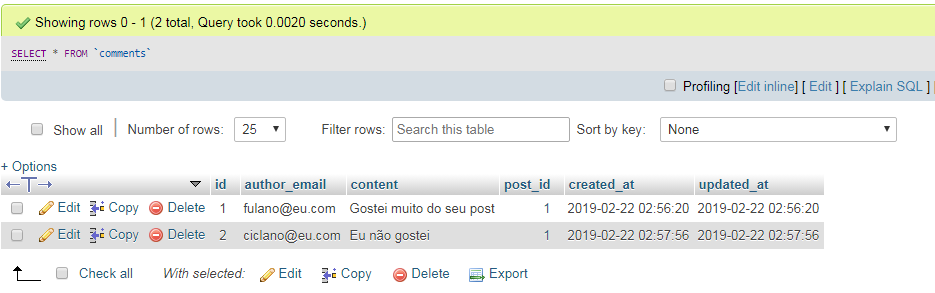
php artisan tinker

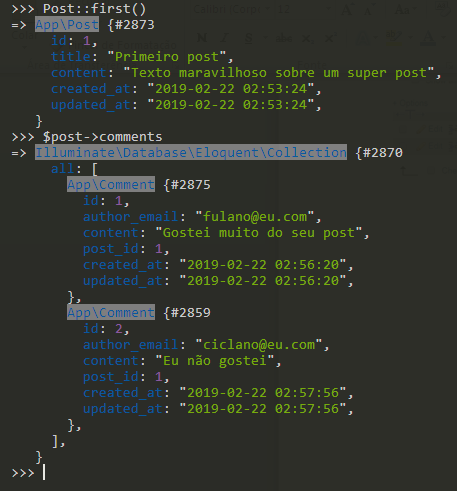


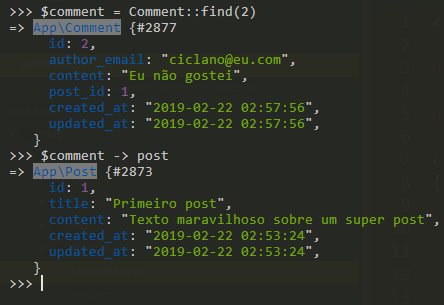


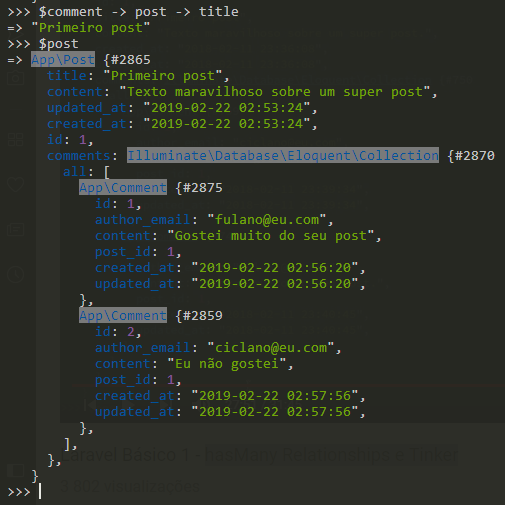


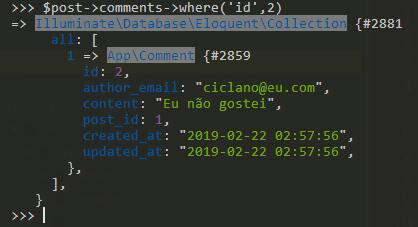


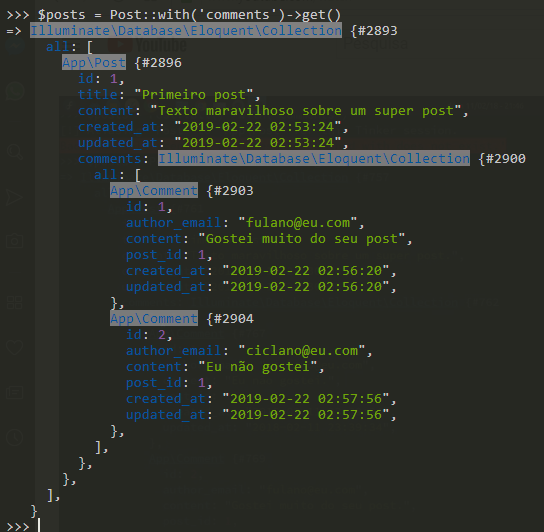




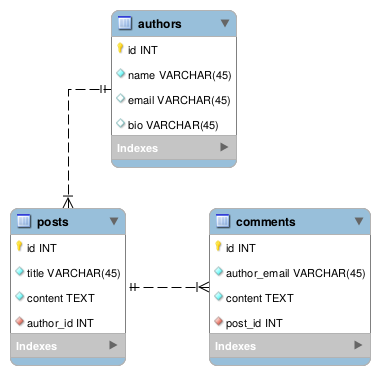






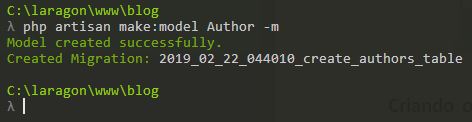


# Aula 02 - hasManyThrough Relationships



## Criando o model para os autores

php artisan make:model Author -m



**database/migrations/2019\_02\_22\_044010\_create\_authors\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateAuthorsTable extends Migration

{

public function up()

{

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

$table->increments('id');

$table->string('name');

$table->string('email');

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

$table->timestamps();

});

}

public function down()

{

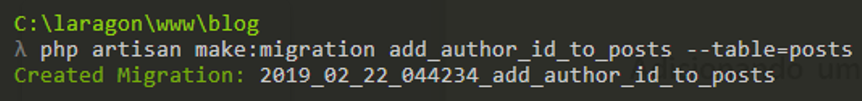
Schema::dropIfExists('authors');

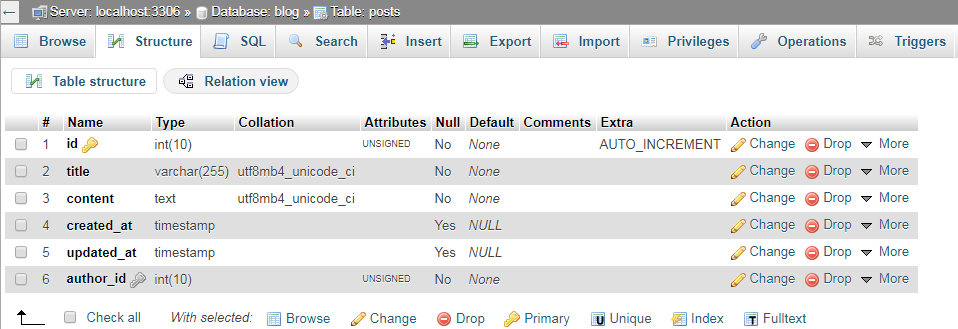
}

}

## Criando uma migration para adicionar um campo na tabela posts

php artisan make:migration add\_author\_id\_to\_posts --table=posts





**database/migrations/2019\_02\_22\_044234\_add\_author\_id\_to\_posts.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class AddAuthorIdToPosts extends Migration

{

public function up()

{

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

$table->integer('author\_id')->unsigned();

$table->foreign('author\_id')->references('id')->on('authors');

});

}

public function down()

{

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

$table->dropForeign('posts\_author\_id\_foreign');

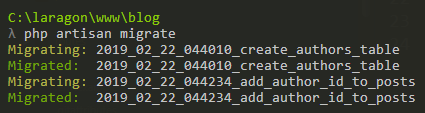
});

}

}

## Criando as tabelas

php artisan migrate



**app/Post.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Post extends Model

{

public function comments()

{

// Um post tem vários comentários

return $this->hasMany('App\Comment');

}

public function author()

{

// Um post pertence a um autor

return $this->belongsTo('App\Author');

}

}

**app/Author.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Author extends Model

{

public function posts()

{

// Um autor tem muitos posts

return $this->hasMany('App\Post');

}

public function comments()

{

// Um autor tem muitos comentários, passando pelos posts

return $this->hasManyThrough('App\Comment', 'App\Post');

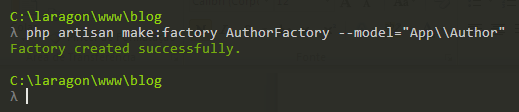
}

}

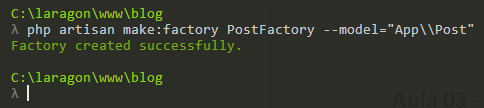
# Aula 03 - Factories

## Criando os factories

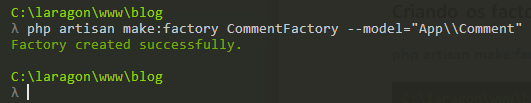
php artisan make:factory AuthorFactory --model="App\\Author"



php artisan make:factory PostFactory --model="App\\Post"



php artisan make:factory CommentFactory --model="App\\Comment"



**database/factory/AuthorFactory.php**

<?php

use Faker\Generator as Faker;

$factory->define(App\Author::class, function (Faker $faker) {

return [

'name' => $faker->name,

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

'bio' => $faker->paragraph(1),

];

});

**database/factory/PostFactory.php**

<?php

use App\Author;

use Faker\Generator as Faker;

$factory->define(App\Post::class, function (Faker $faker) {

return [

'title' => $faker->sentence(4),

'content' => $faker->paragraph(4),

'author\_id' => function () {

return Author::orderByRaw("RAND()")

->take(1)

->first()

->id;

}

];

});

**database/factory/CommentFactory.php**

<?php

use App\Post;

use Faker\Generator as Faker;

$factory->define(App\Comment::class, function (Faker $faker) {

return [

'author\_email' => $faker->unique()->safeEmail,

'content' => $faker->paragraph(2),

'post\_id' => function () {

return Post::orderByRaw("RAND()")

->take(1)

->first()

->id;

}

];

});

**database/seeds/DatabaseSeeder.php**

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

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

echo "Creating 10 authors...\n";

factory(App\Author::class, 10)->create();

echo "Creating 36 posts related to random authors...\n";

factory(App\Post::class, 36)->create();

echo "Creating 67 comments related to random posts...\n";

factory(App\Comment::class, 67)->create();

}

}

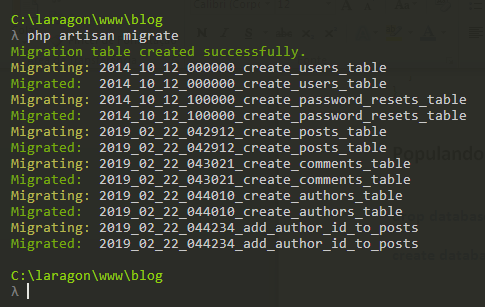
## Populando as tabelas

drop database blog;

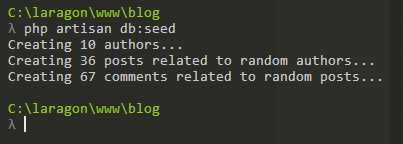
create database blog;

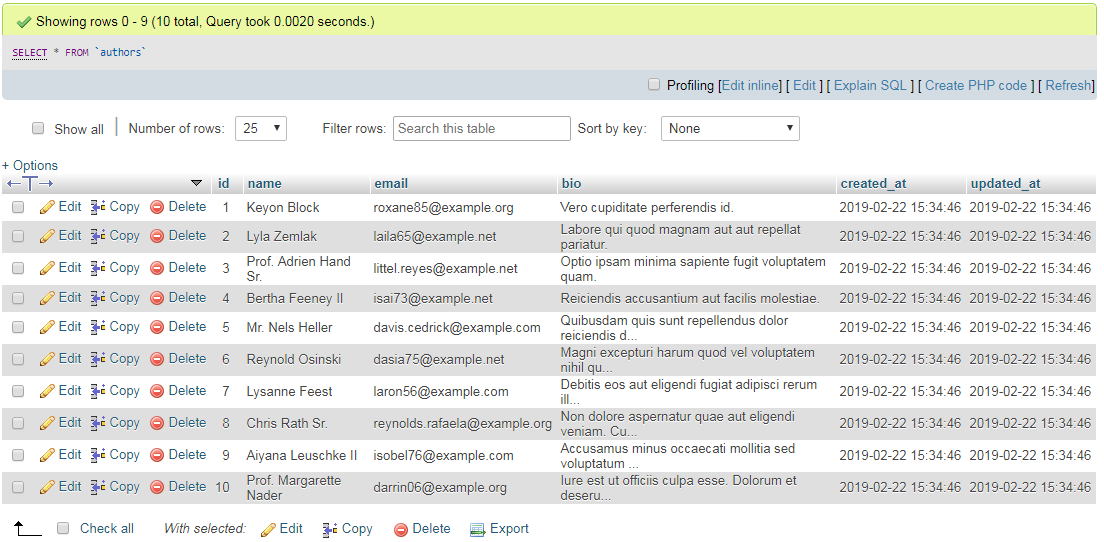
No terminal:

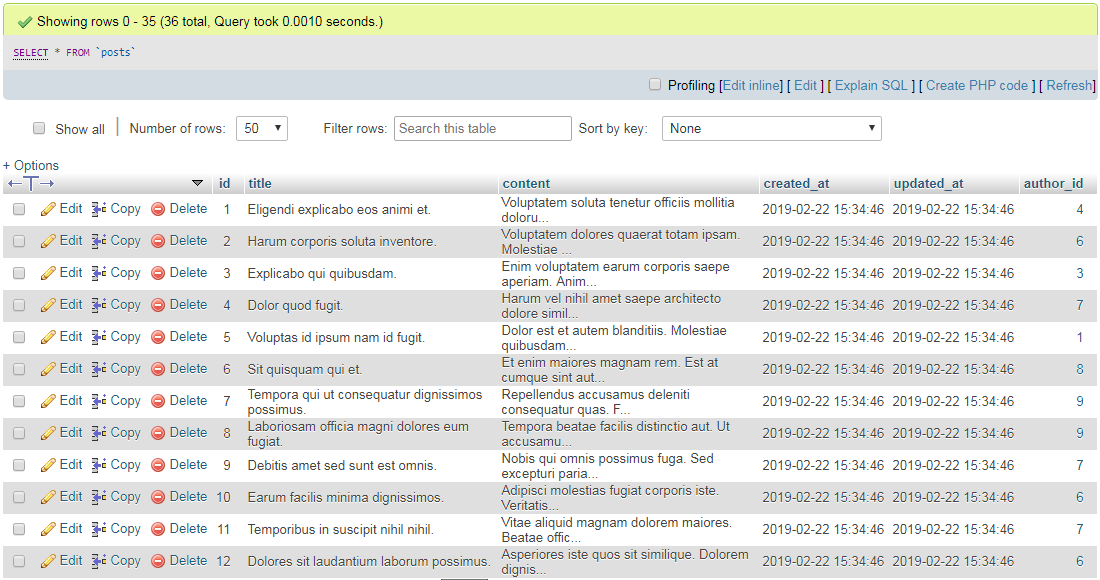
php artisan migrate

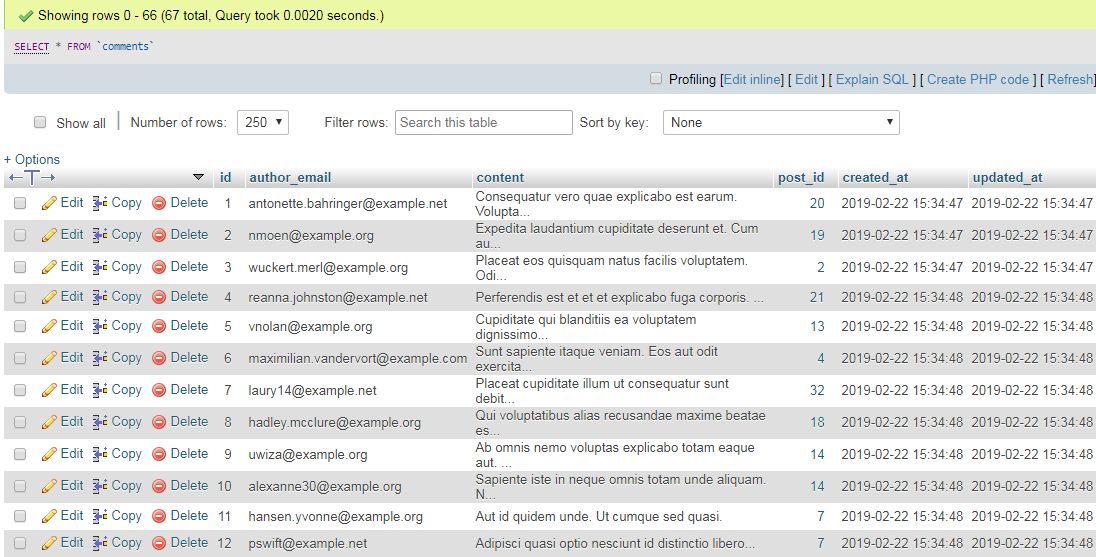


php artisan db:seed





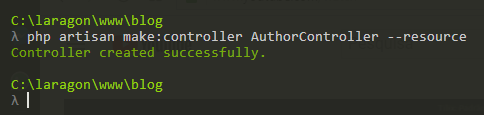




# Aula 04 - Controllers e Rotas

## Criando o controller para os autores

php artisan make:controller AuthorController --resource



**routes/web.php**

Route::resource('authors','AuthorController');

**app/Http/Controllers/AuthorController.php**

<?php

namespace App\Http\Controllers;

use App\Author;

use Illuminate\Support\Facades\DB;

use Illuminate\Http\Request;

class AuthorController extends Controller

{

public function index()

{

//$authors = json\_encode(Author::all(), JSON\_PRETTY\_PRINT);

$authors = Author::join('posts', 'posts.author\_id', '=', 'authors.id')

->groupBy('authors.id')

->get([

'authors.id',

'authors.name',

'authors.email',

DB::raw('count(posts.id) as posts')

]

);

$authors = json\_encode($authors, JSON\_PRETTY\_PRINT);

return "<pre>$authors</pre>";

}

public function show(Author $author)

{

$posts = $author->posts->count();

$text = <<<TEXT

ID: $author->id

Name: $author->name

E-mail: $author->email

Bio: $author->bio

Posts: $posts

TEXT;

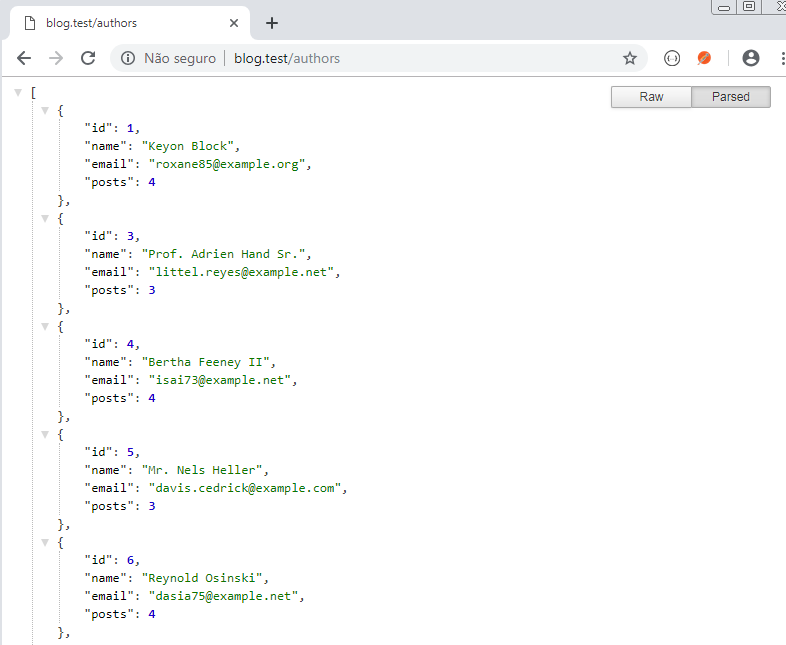
/\* A marca de fim do heredoc (TEXT;)

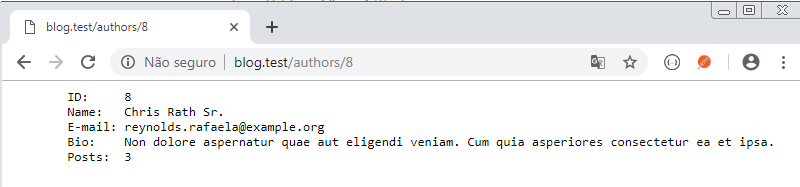
deve ficar na coluna zero da linha \*/

return "<pre>$text</pre>";

}

}

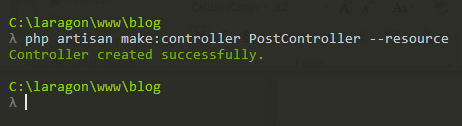




# Aula 05 - PostController

## Criando o controller para os posts

php artisan make:controller PostController --resource



**routes/web.php**

Route::resources([

'authors' => 'AuthorController',

'posts' => 'PostController'

]);

**app/Http/Controllers/PostController.php**

<?php

namespace App\Http\Controllers;

use App\Post;

use Illuminate\Support\Facades\DB;

use Illuminate\Http\Request;

class PostController extends Controller

{

public function index()

{

$posts = json\_encode(Post::all(), JSON\_PRETTY\_PRINT);

return "<pre>$posts</pre>";

}

public function show(Post $post)

{

$author = $post->author->name;

$text = <<<TEXT

ID: $post->id

Title: $post->title

Content: $post->content

Author: $author

TEXT;

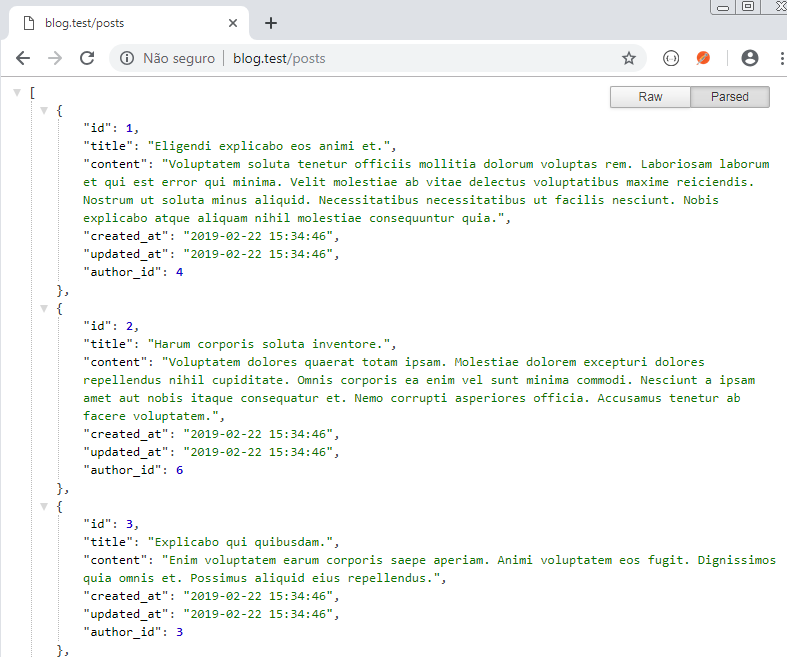
// A marca do fim do heredoc (TEXT;) deve ficar na coluna zero da linha.

// ou teremos um erro - Unexpected end of file

return "<pre>$text</pre>";

}

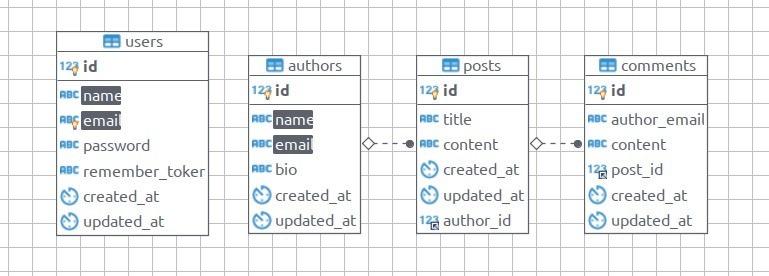
}





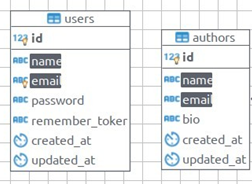
# Aula 06 - Login

## Fazer login dos autores



## Alterar o model Author para relacionar com User

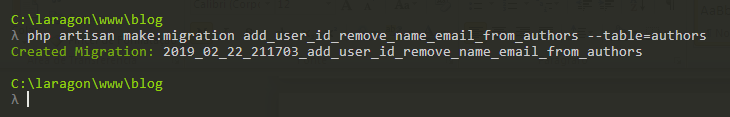
* Tabelas users e authors
* Tabelas users e authors com relacionamento 1:1



## Alterar a tabela authors

Criar migration para adicionar user\_id foreign e remover as colunas name e email

php artisan make:migration add\_user\_id\_remove\_name\_email\_from\_authors --table=authors



## Código para a migration

**database/migrations/add\_user\_id\_remove\_name\_email\_from\_authors.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class AddUserIdRemoveNameEmailFromAuthors extends Migration

{

public function up()

{

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

$table->dropColumn('name');

$table->dropColumn('email');

$table->integer('user\_id')->unsigned();

$table->foreign('user\_id')

->references('id')

->on('users')

->onDelete('cascade');

});

}

public function down()

{

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

$table->string('name');

$table->string('email');

$table->dropForeign('authors\_user\_id\_foreign');

$table->dropColumn('user\_id');

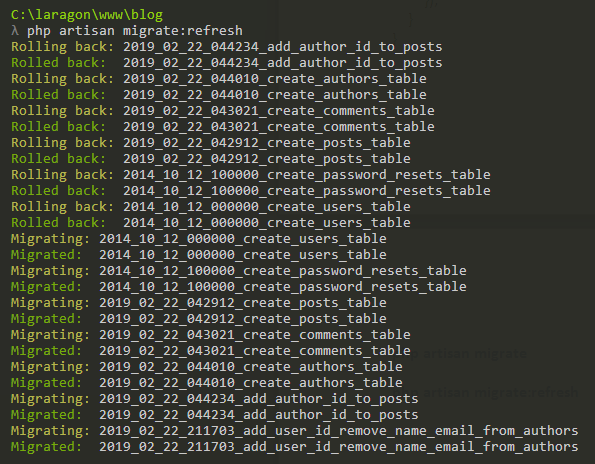
});

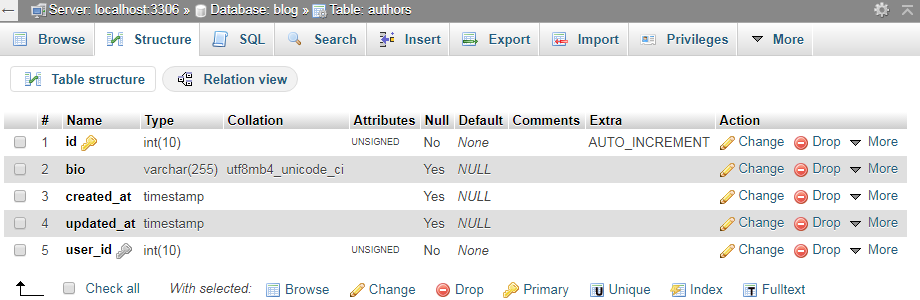
}

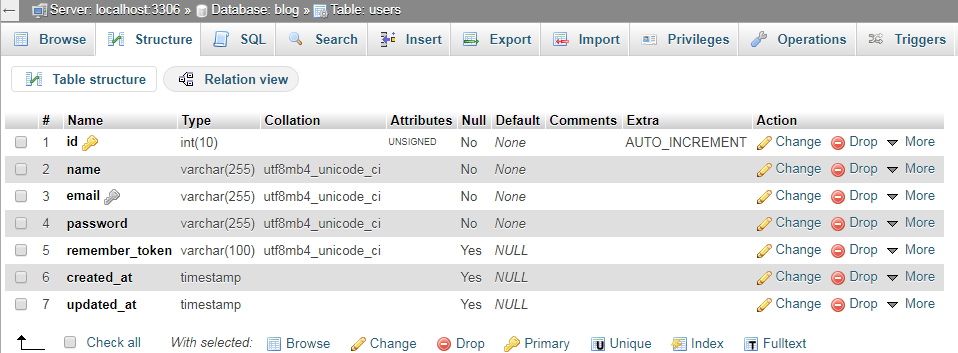
}

php artisan migrate

php artisan migrate:refresh







## Ajustar os models User e Author

Setar o relacionamento: adicionar em app\User.php

public function author()

{

return $this->hasOne('App\Author');

}

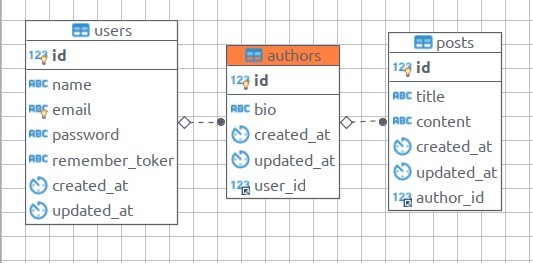
Setar o relacionamento: adicionar em app\Author.php

public function user()

{

return $this->belongsTo('App\User');

}



### Inserir user e author no Banco de Dados

php artisan tinker

No tinker:

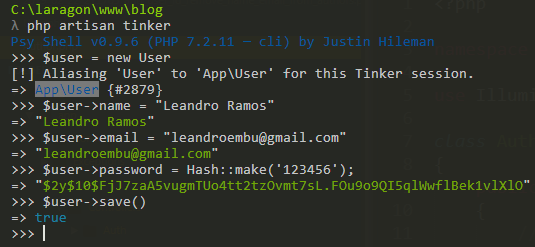
>>> $user = new User

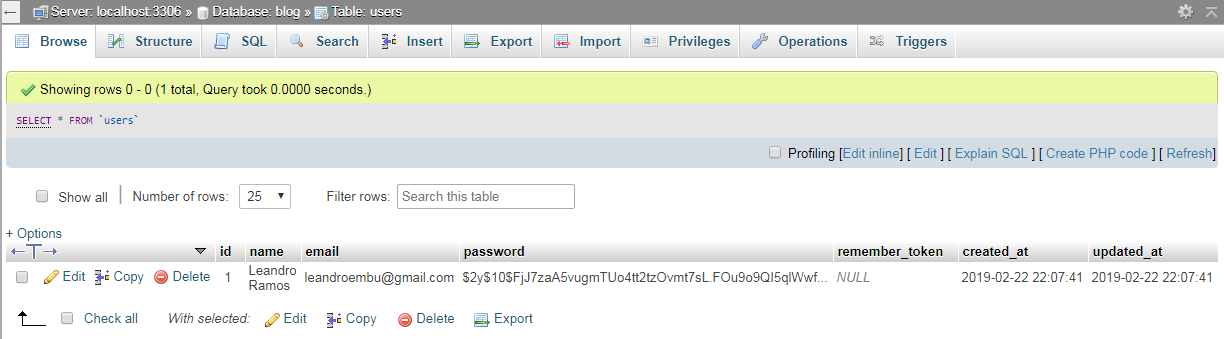
>>> $user->name = "Leandro Ramos"

>>> $user->email = "leandroembu@gmail.com"

>>> $user->password = Hash::make('123456');

>>> $user->save()

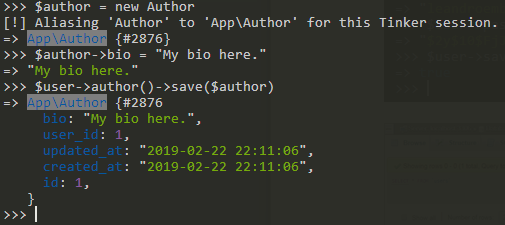


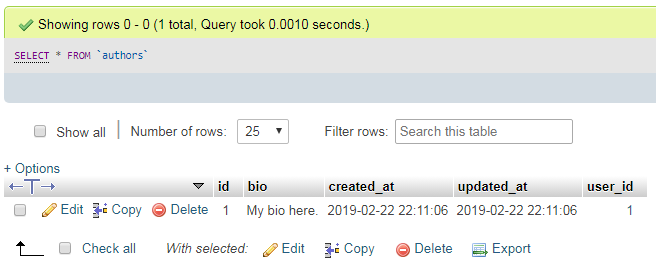


>>> $author = new Author

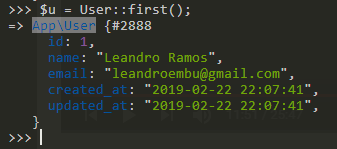
>>> $author->bio = "My bio here."

>>> $user->author()->save($author)





>>> $u- = User::first();

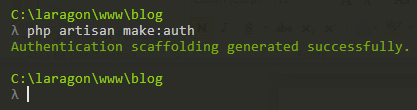


>>> $u- -> author -> bio;



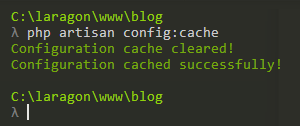
## Criar a autenticação

php artisan make:auth



## Limpar o cache

php artisan config:cache



**routes/web.php**

Route::resources([

'authors' => 'AuthorController',

'posts' => 'PostController'

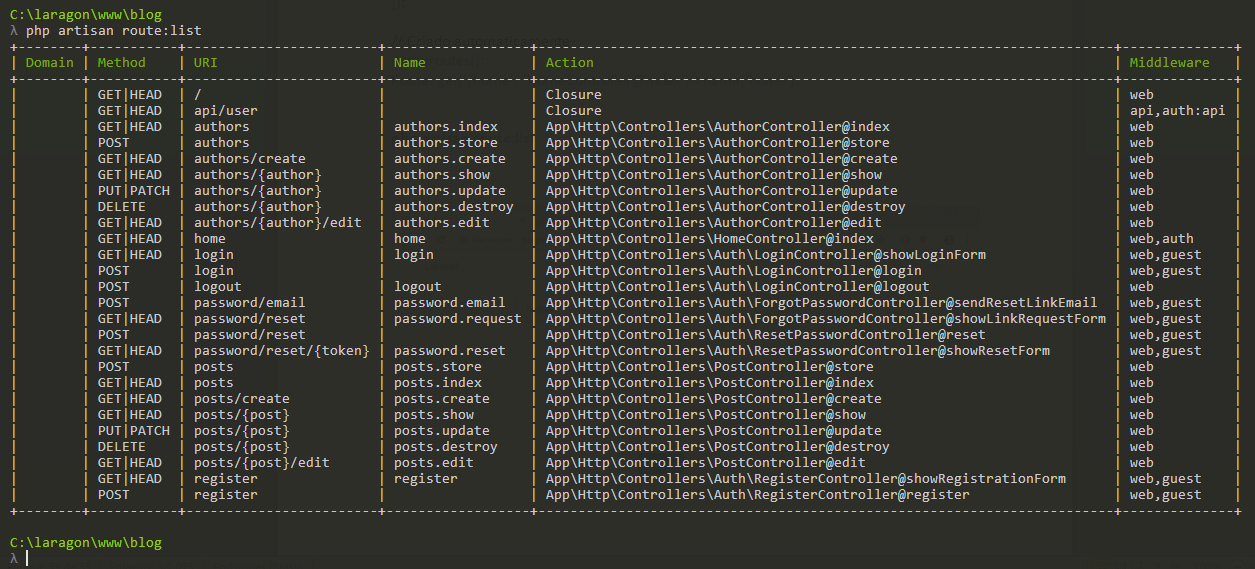
]);

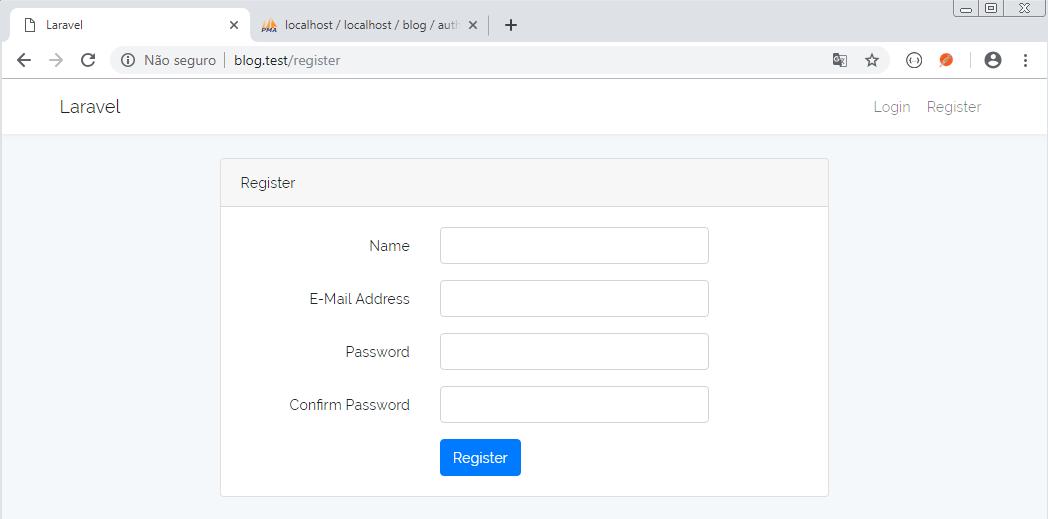
// Criado automaticamente

Auth::routes();

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

**php artisan route:list**





- Adicionar o campo "bio" ao form de registro

- Adicionar ao formulário (antes do campo password)

**resources/views/auth/register.blade.php:**

<div class="form-group row">

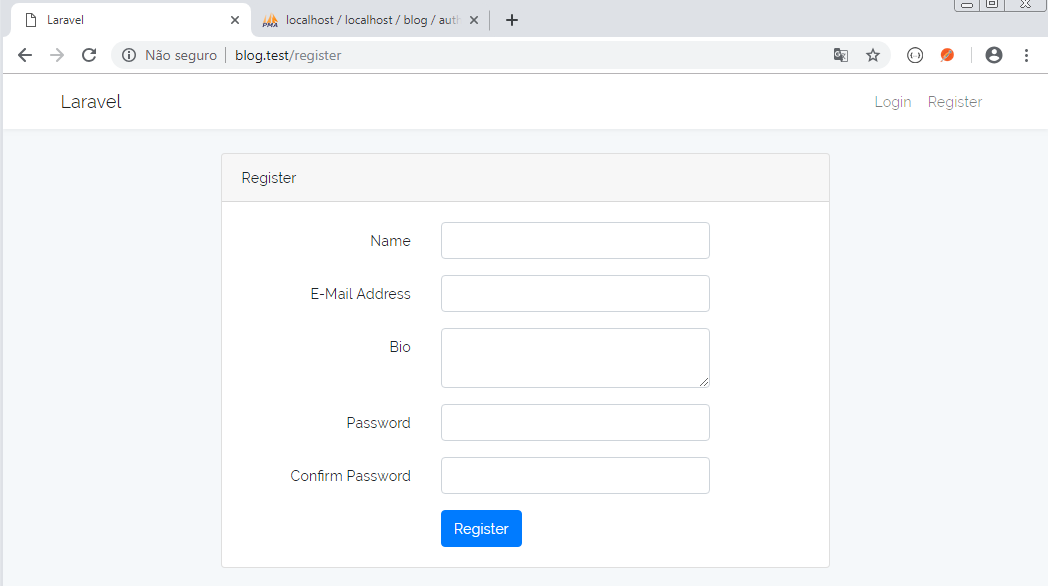
<label for="bio" class="col-md-4 col-form-label text-md-right">Bio</label>

<div class="col-md-6">

<textarea id="bio" class="form-control" name="bio" value="{{ old('bio') }}"></textarea>

</div>

</div>



## Ajustar o controller

**app/Http/Controllers/Auth/RegisterController.php**

// Coloque abaixo do namespace

use App\User;

// ... aqui tem um monte de código

// troque o código do método create por isso:

protected function create(array $data)

{

$user = User::create([

'name' => $data['name'],

'email' => $data['email'],

'password' => bcrypt($data['password']),

]);

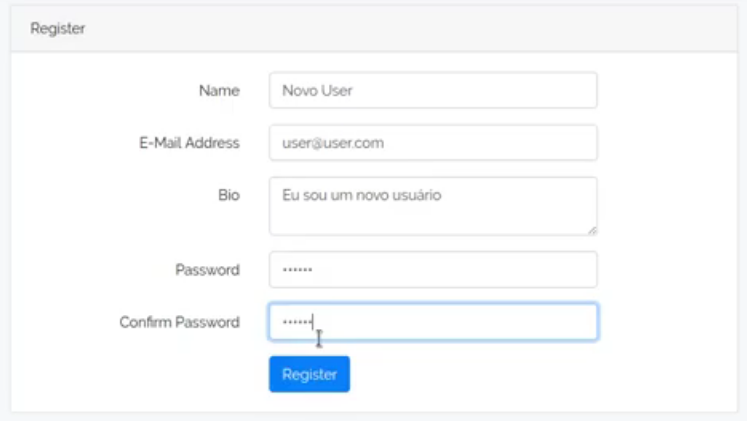
$author = new Author;

$author->bio = $data['bio'];

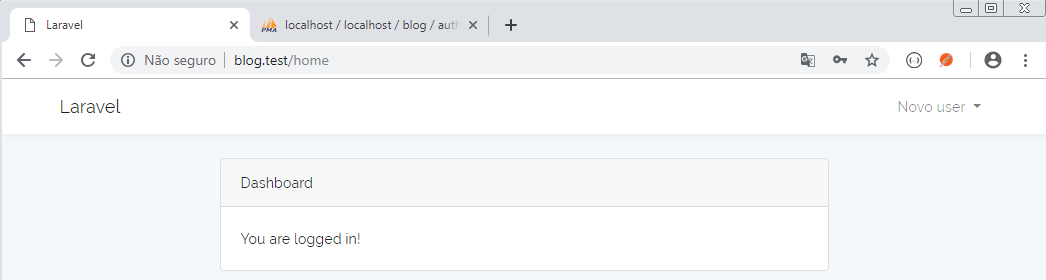
$user->author()->save($author);

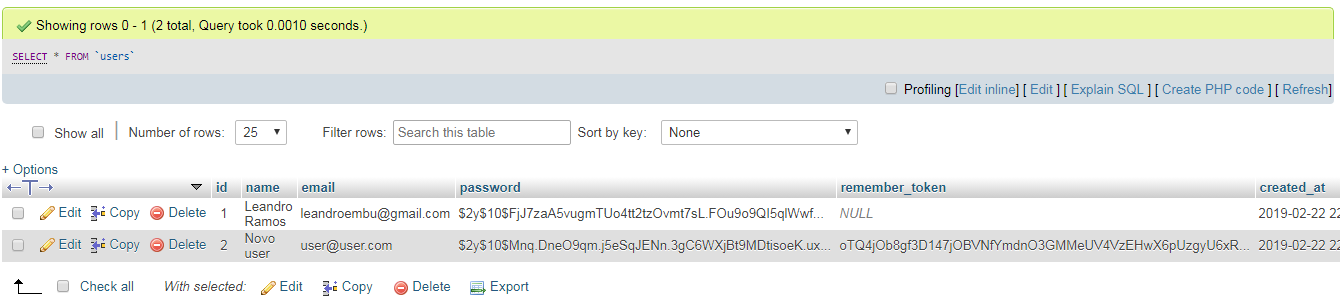
return $user;

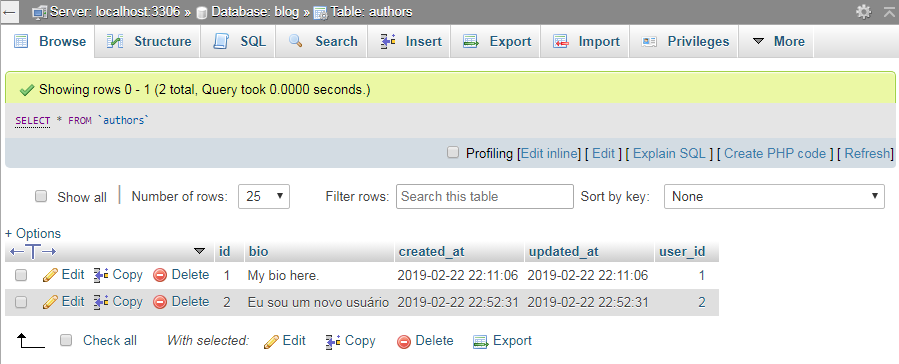
}

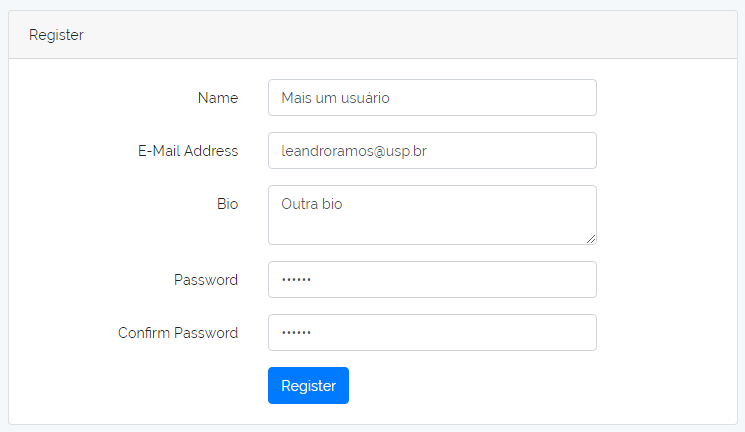


Senha: 123456

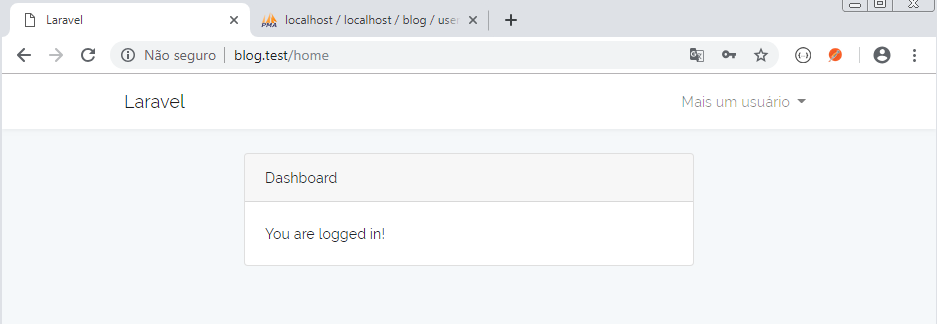


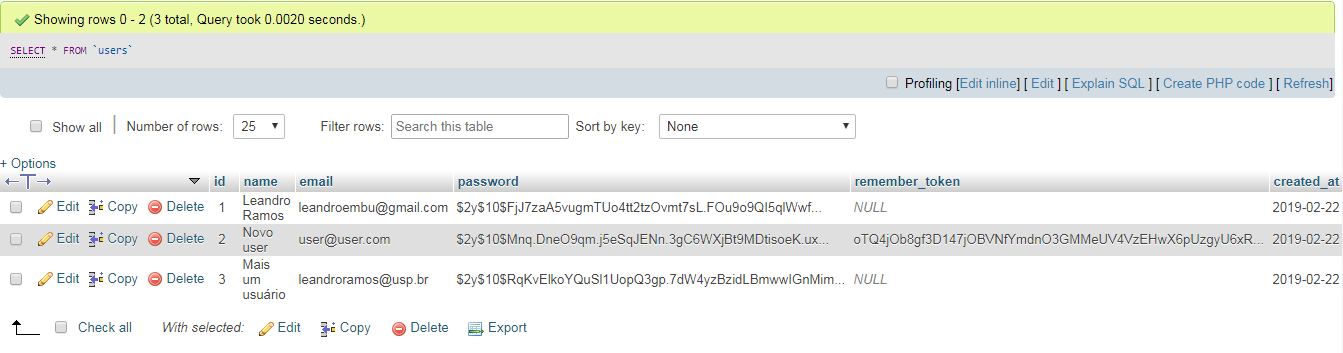


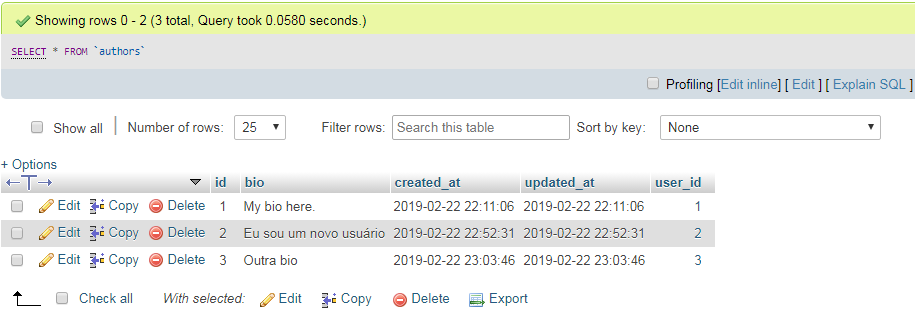




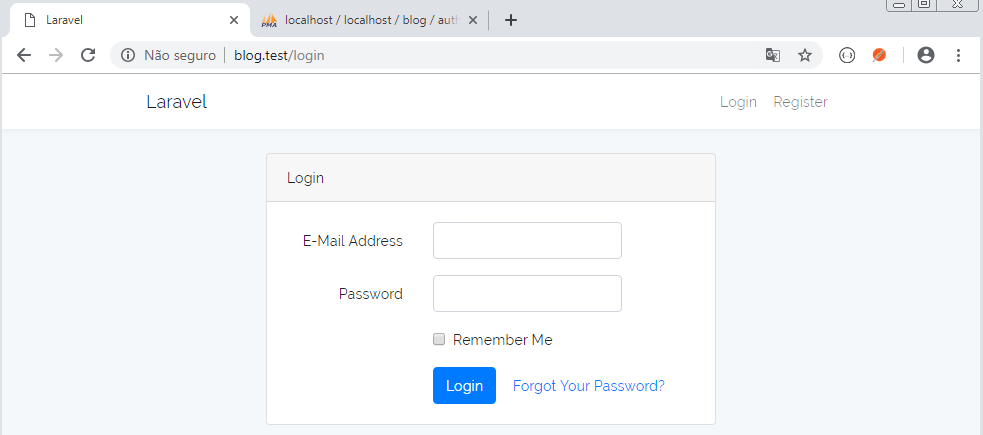
Senha: 654321

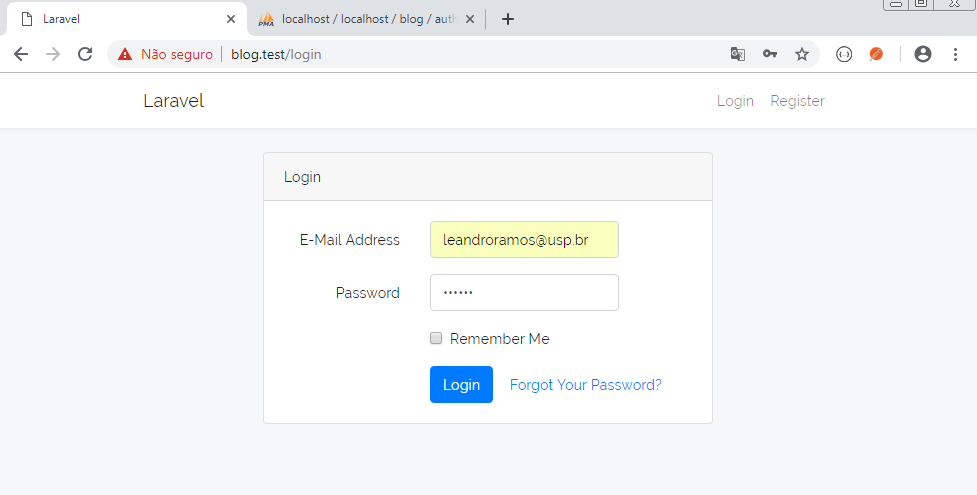


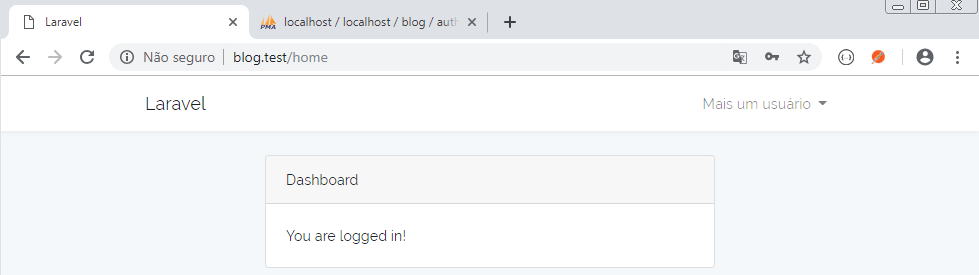




Fazer login:







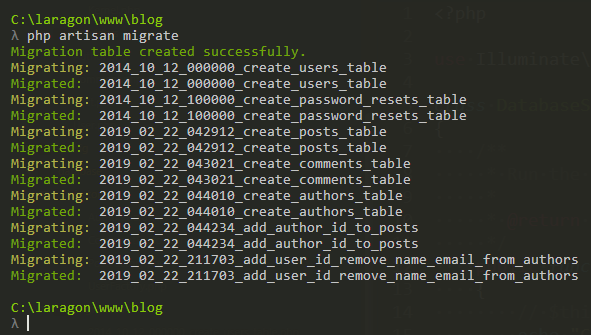
# Aula 07 - Views

## Populando o banco de dados

- Apague o banco de dados blog e em seguida o recrie.

- Recrie as tabelas:

php artisan migrate



- Altere os arquivos a seguir:

**database/factories/AuthorFactory.php**

<?php

use Faker\Generator as Faker;

$factory->define(App\Author::class, function (Faker $faker) {

return [

// 'name' => $faker->name,

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

'bio' => $faker->paragraph(1),

'user\_id' => $faker->unique()->numberBetween(1,10)

];

});

**database/factories/UserFactory.php**

<?php

use Faker\Generator as Faker;

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

return [

'name' => $faker->name,

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

'password' => '$2y$10$TKh8H1.PfQx37YgCzwiKb.KjNyWgaHb9cbcoQgdIVFlYg7B77UdFm', // secret

'remember\_token' => str\_random(10),

];

});

**database/seeds/DatabaseSeeder.php**

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

public function run()

{

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

echo "Creating 10 users...\n";

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

echo "Creating 10 authors...\n";

factory(App\Author::class, 10)->create();

echo "Creating 36 posts related to random authors...\n";

factory(App\Post::class, 36)->create();

echo "Creating 67 comments related to random posts...\n";

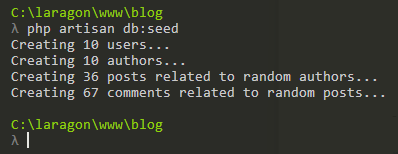
factory(App\Comment::class, 67)->create();

}

}

- Popule o banco de dados.

php artisan db:seed



## Listando todos os posts

**app/Controllers/PostController.php**

public function index()

{

$posts = Post::all();

return view('posts.index', compact('posts'));

}

**resources/views/posts/index.blade.php**

@extends('layouts.app')

@section('content')

<div class="row justify-content-center">

<div class="col-md-8">

<div class="card card-default">

@foreach ($posts as $post)

<div class="card-header">

<h3>{{ $post->title }} :: por {{ $post->author->user->name }}</h3>

</div>

<div class="card-body">

<p>

{{ substr($post->content, 0, 60) }}...

<a href="{{ action('PostController@show', $post->id) }}" title="Ler o post">Ler o post</a>

</p>

</div>

@endforeach

</div>

</div>

</div>

@endsection

**resources/views/posts/show.blade.php**

@extends('layouts.app')

@section('content')

<div class="row justify-content-center">

<div class="col-md-8">

<div class="card card-default">

<div class="card-header">

<h3>{{ $post->title }} :: por {{ $post->author->user->name }}</h3>

</div>

<div class="card-body">

<p>

{{ $post->content }}

</p>

<h4>Comentários:</h4>

<!-- Comentários do post -->

@foreach($post->comments as $comment)

<div class="card card-default">

<div class="card-header">

{{ $comment->author\_email }}

</div>

<div class="card-body">

{{ $comment->content }}

</div>

</div>

<br>

@endforeach

</div>

</div>

</div>

</div>

@endsection

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

<!DOCTYPE html>

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

<head>

<meta charset="utf-8">

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

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

<!-- CSRF Token -->

<meta name="csrf-token" content="{{ csrf\_token() }}">

<title>{{ config('app.name', 'Laravel') }}</title>

<!-- Scripts -->

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

<!-- Fonts -->

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

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

<!-- Styles -->

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

</head>

<body>

<div id="app">

<nav class="navbar navbar-expand-md navbar-light navbar-laravel">

<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>

<li class="nav-item">

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

</li>

@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>

<main class="py-4">

@yield('content')

</main>

</div>

</body>

</html>

**app/Controllers/PostController.php**

<?php

namespace App\Http\Controllers;

use App\Post;

use Illuminate\Support\Facades\DB;

use Illuminate\Http\Request;

class PostController extends Controller

{

public function index()

{

$posts = Post::all();

return view('posts.index', compact('posts'));

}

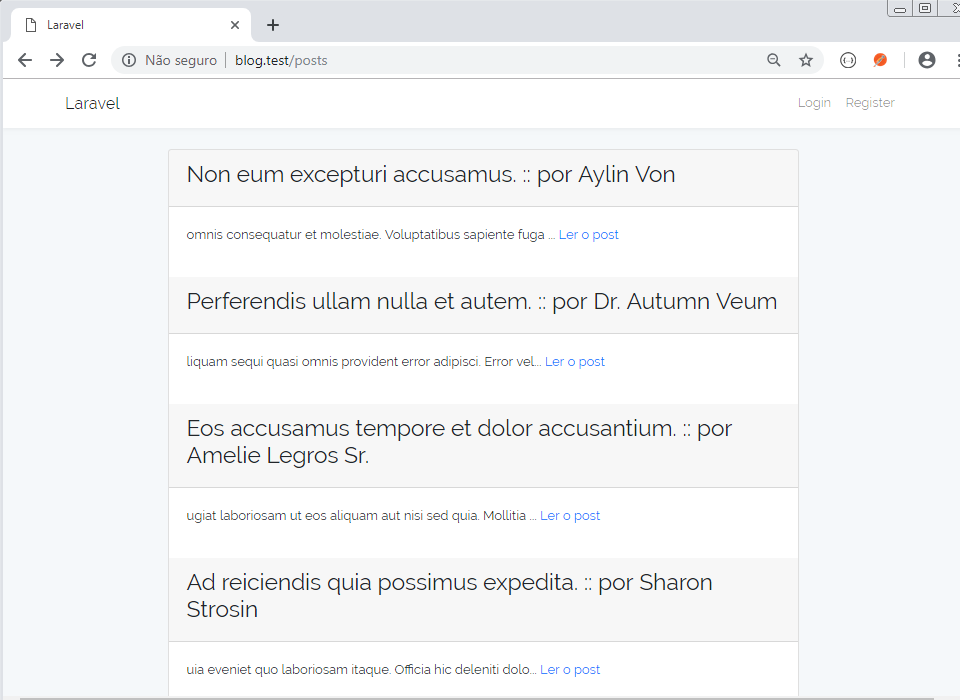
public function show(Post $post)

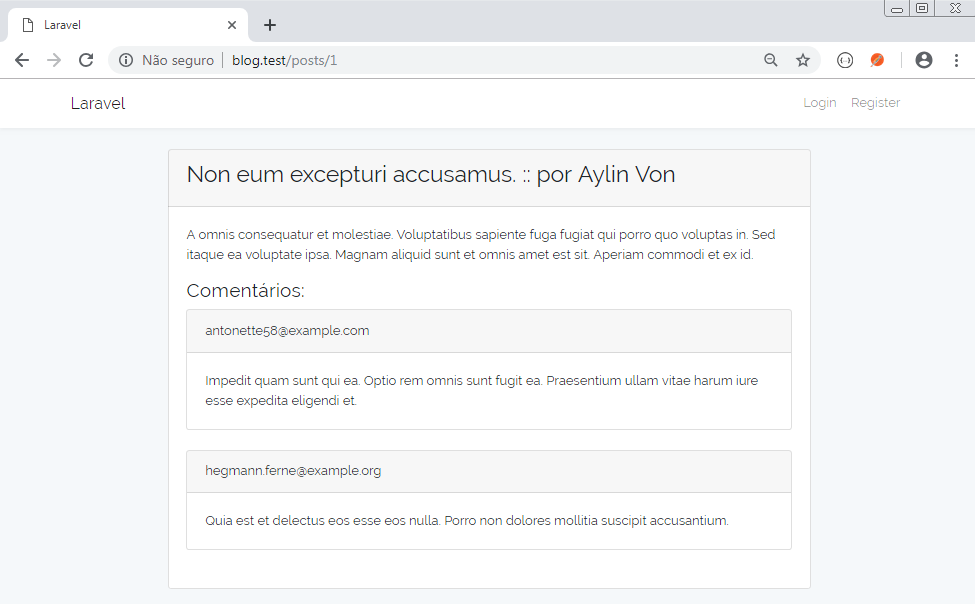
{

return view('posts.show', compact('post'));

}

}





# Aula 08 - Views - Criando Posts

## Logar o usuário

O user (que também é um author) só poderá criar e editar posts se estiver logado.

Ajustar o PostController para autenticar o usuário

Criar método \_\_construct em:

**app/Http/Controllers/PostController.php**

public function \_\_construct()

{

$this->middleware('auth')->except(['index', 'show']);

}

## Criando o formulário

**resources/views/forms/post.blade.php**

@extends('layouts.app')

@section('content')

<div class="row justify-content-center">

<div class="col-md-8">

<div class="card card-default">

<div class="card-header">

<h3>Post</h3>

</div>

<div class="card-body">

<form method="post" action="{{ $action or url('posts') }}">

{{csrf\_field()}}

@isset($post) {{method\_field('patch')}} @endisset

<input name="author\_id" type="hidden" value="{{ encrypt(Auth::User()->author->id) }}">

<div class="form-group row">

<label for="title" class="col-md-4 col-form-label text-md-right">Title</label>

<div class="col-md-6">

<input id="title" class="form-control" name="title" type="text" value="{{ $post->title or ''}}">

</div>

</div>

<div class="form-group row">

<label for="content" class="col-md-4 col-form-label text-md-right">Content</label>

<div class="col-md-6">

<textarea id="content" class="form-control" name="content">{{ $post->content or '' }}</textarea>

</div>

</div>

<div class="form-group row mb-0">

<div class="col-md-6 offset-md-4">

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

Save

</button>

</div>

</div>

</form>

</div>

</div>

</div>

</div>

@endsection

**resources/views/posts/create.blade.php**

@include('forms.post')

**resources/views/posts/edit.blade.php**

@include('forms.post')

### Ajustando o app/Http/Controllers/PostController.php

Ajustar o controller para efetuar as operações corretamente

**Método index**

public function index()

{

$posts = Post::orderBy('created\_at', 'desc')->get();

return view('posts.index', compact('posts'));

}

**Método create**

public function create()

{

return view('posts.create');

}

**Método store**

public function store(Request $request)

{

$post = new Post;

$post->title = $request->title;

$post->content = $request->content;

$post->author\_id = decrypt($request->author\_id);

$post->save();

$request->session()->flash('alert-success', 'Post criado com sucesso!');

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

}

**Método edit**

public function edit(Post $post)

{

$action = action('PostController@update', $post->id);

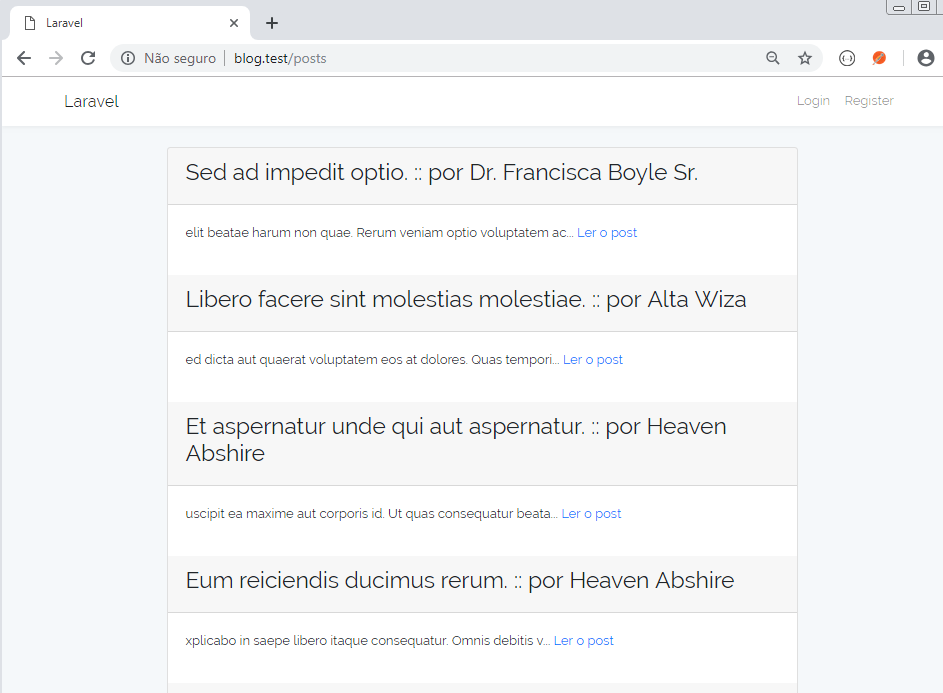
return view('posts.edit', compact('post', "action"));

}

### View para edição do Post

**resources/views/posts/edit.blade.php**

@include('forms.post')





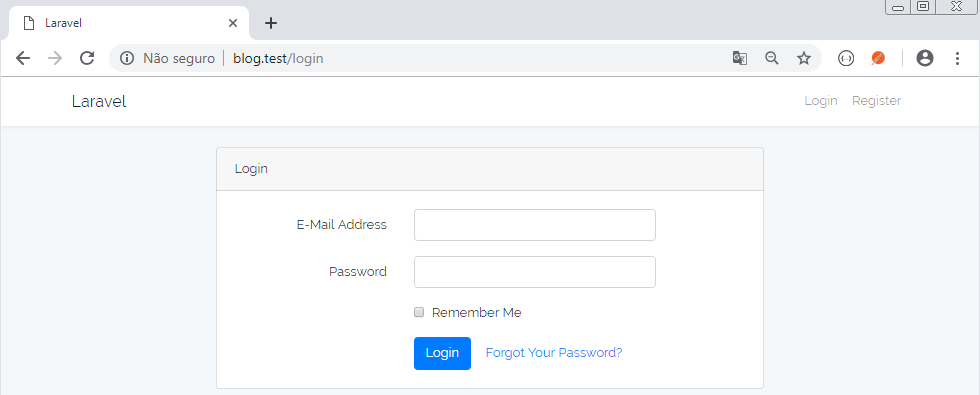
Entrando com:

http://blog.test/posts/create

ou

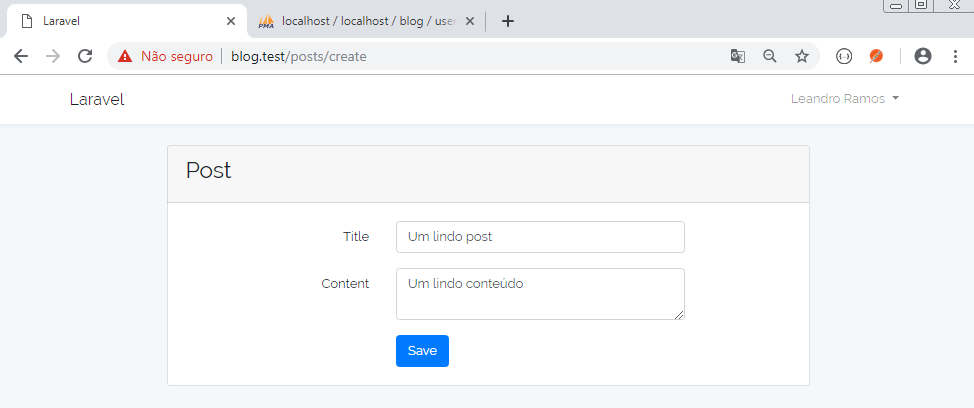
http://blog.test/posts/5/edit

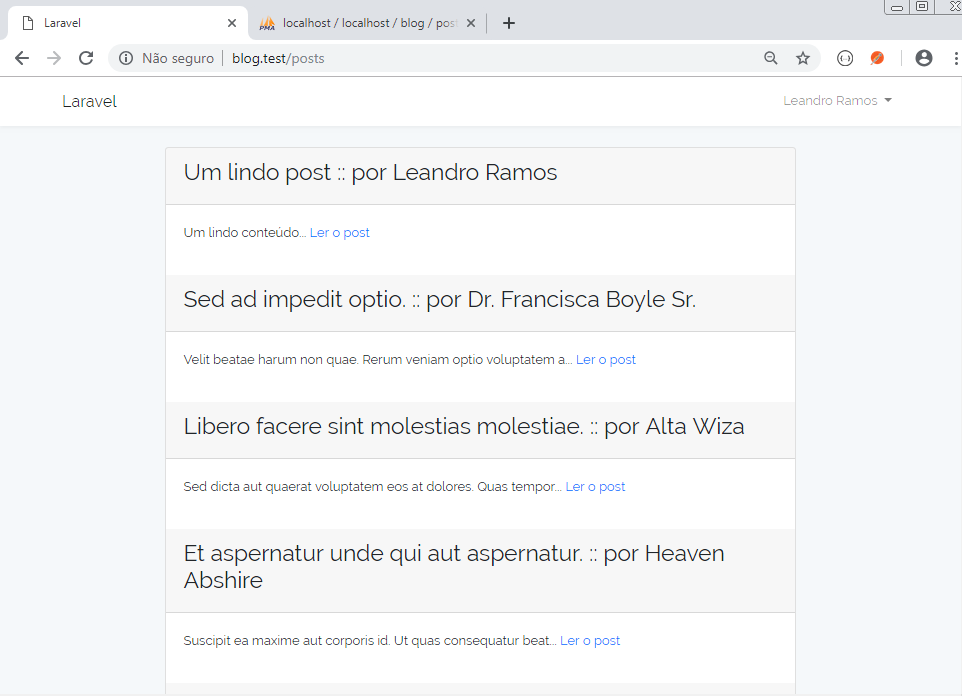
Vai para a tela de login

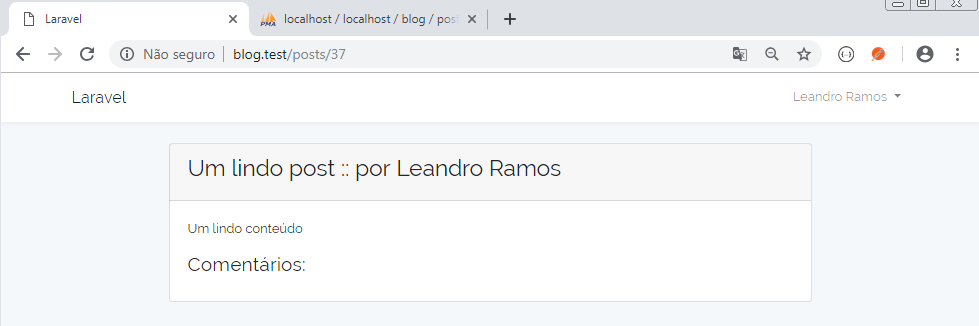


## Criando um novo post

**blog.test/posts/create**







## Editando um post

### Editando o post 1



# Aula 09 - Listando posts do autor

## Alterar os seeders do banco de dados

**database/seeds/DatabaseSeeder.php**

<?php

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

/\*\*

\* Run the database seeds.

\*

\* @return void

\*/

public function run()

{

echo "Creating 10 users...\n";

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

echo "Creating 10 authors...\n";

factory(App\Author::class, 10)->create();

echo "Creating 36 posts related to random authors...\n";

factory(App\Post::class, 36)->create();

echo "Creating 67 comments related to random posts...\n";

factory(App\Comment::class, 67)->create();

}

}

**database/factories/AuthorFactory.php**

<?php

use Faker\Generator as Faker;

$factory->define(App\Author::class, function (Faker $faker) {

return [

'bio' => $faker->paragraph(1),

'user\_id' => $faker->unique()->numberBetween(1, 10)

];

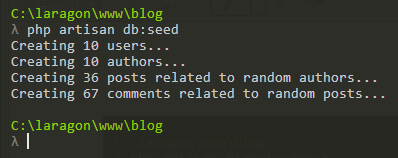
});

## "Zerar"o banco de dados e popular com o novo seeder

php artisan migrate:fresh



php artisan db:seed



## Criar a rota para a view dos posts do autor

**routes/web.php**

Route::resources([

'authors' => 'AuthorController',

'posts' => 'PostController'

]);

Auth::routes();

Route::get('authors/{author}/posts', 'AuthorController@posts');

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

## Criar o método que lista os posts do autor

**app/Http/Controllers/AuthorController.php**

public function posts(Author $author)

{

$posts = $author->posts;

return view('authors.posts', compact('posts'));

}

## Criar a view que lista os posts do autor

**resources/views/authors/posts.blade.php**

@extends('layouts.app')

@section('content')

<div class="row justify-content-center">

<div class="col-md-8">

<h2>Meus Posts</h2>

<div class="flash-message">

@foreach (['danger', 'warning', 'success', 'info'] as $msg)

@if(Session::has('alert-' . $msg))

<p class="alert alert-{{ $msg }}">{{ Session::get('alert-' . $msg) }} <a href="#" class="close" data-dismiss="alert" aria-label="fechar">&times;</a></p>

@endif

@endforeach

</div>

@foreach ($posts as $post)

<div class="card card-default">

<div class="card-header">

<h3>{{ $post->title }} :: <small>por {{ $post->author->user->name }}</small></h3>

</div>

<div class="card-body">

<p>

{{ $post->content }} <br>

<a class="btn btn-danger" href="{{ action('PostController@destroy', $post->id) }}" title="Apagar o post">Apagar</a><br>

<a class="btn btn-primary" href="{{ action('PostController@edit', $post->id) }}" title="Editar o post">Editar</a><br>

</p>

</div>

</div>

<br>

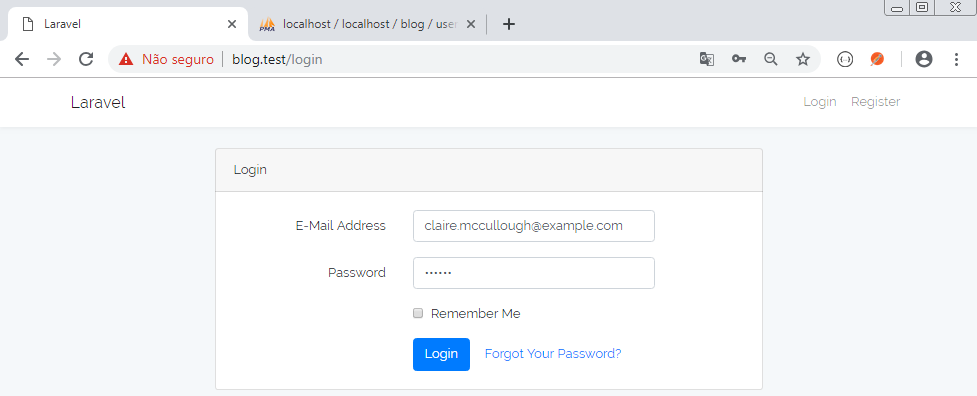
@endforeach

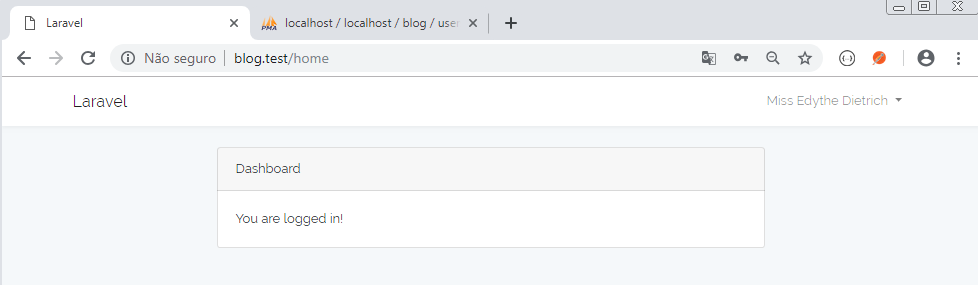
</div>

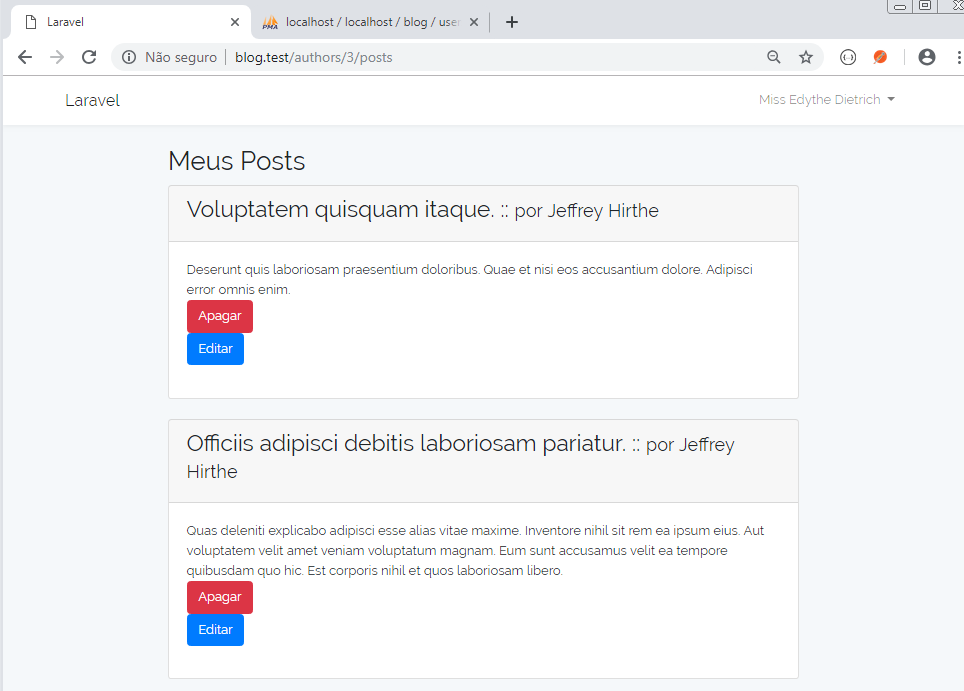
</div>

@endsection

A senha de todos os usuários do banco é: secret

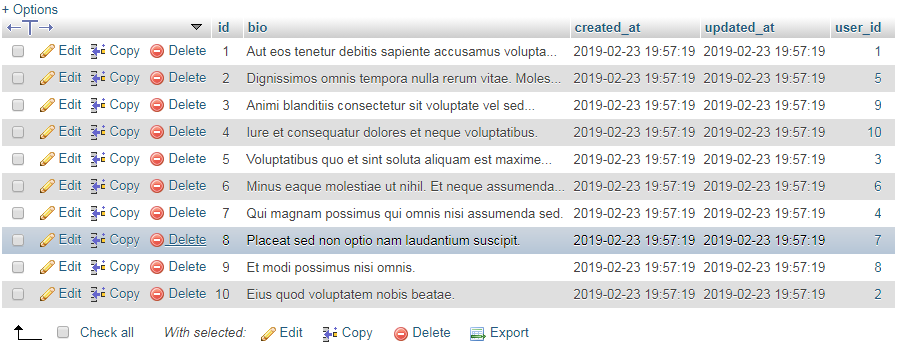




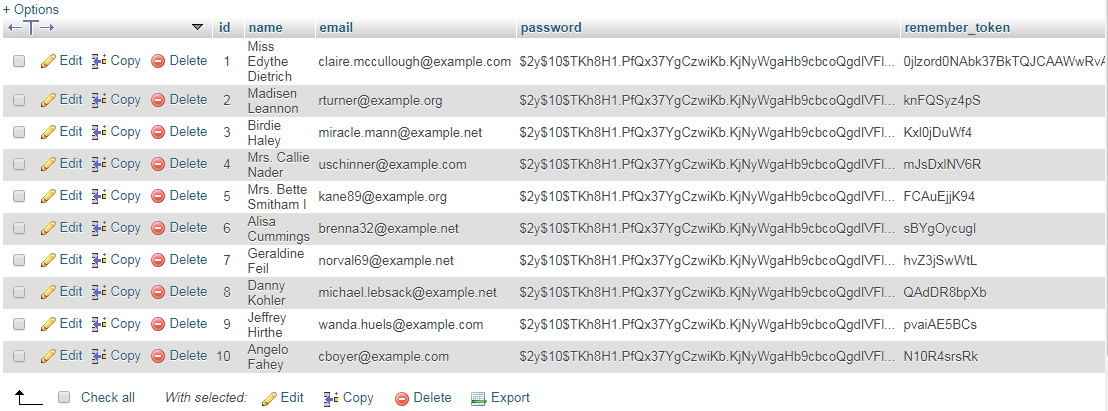


Perceba que o autor 3 é o usuário de id 9

**Tabela authors**



**Tabela users**



OBS.: As rotinas para apagar e editar os posts ainda não estão implementadas.

# Aula 10 - Apagando posts do autor

## Implementar método destroy no PostController

**app/Http/Controllers/PostController.php**

public function destroy(Request $request, Post $post)

{

$post->delete();

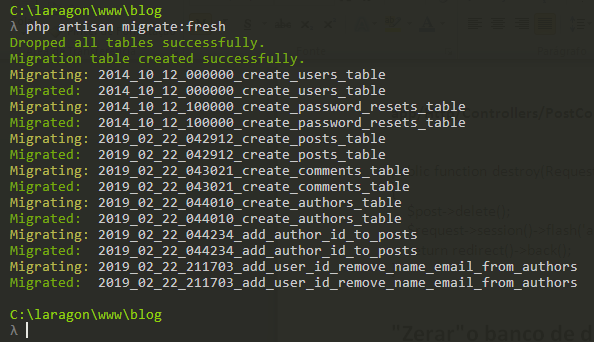
$request->session()->flash('alert-success', 'Post apagado com sucesso!');

return redirect()->back();

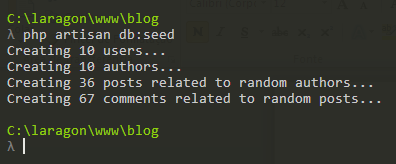
}

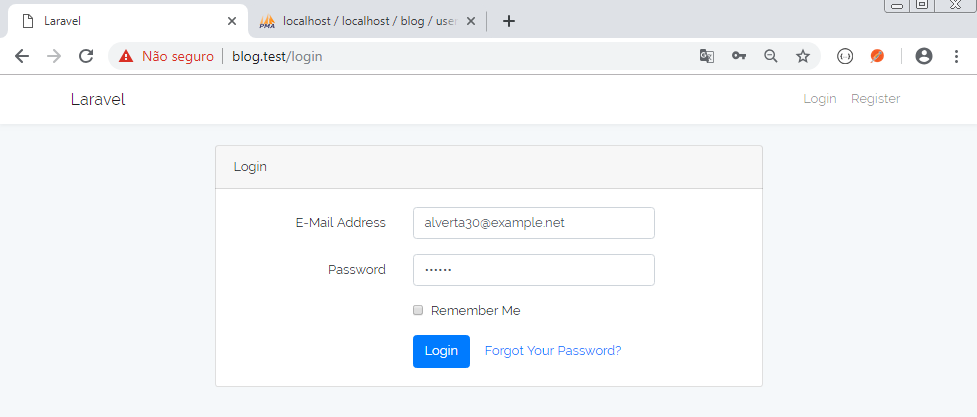
## "Zerar"o banco de dados e popular com o novo seeder

php artisan migrate:fresh

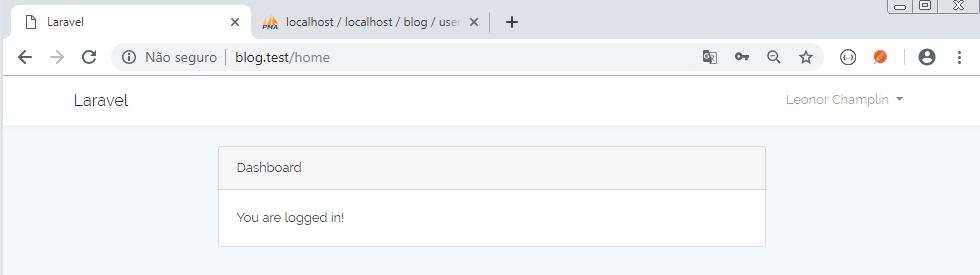


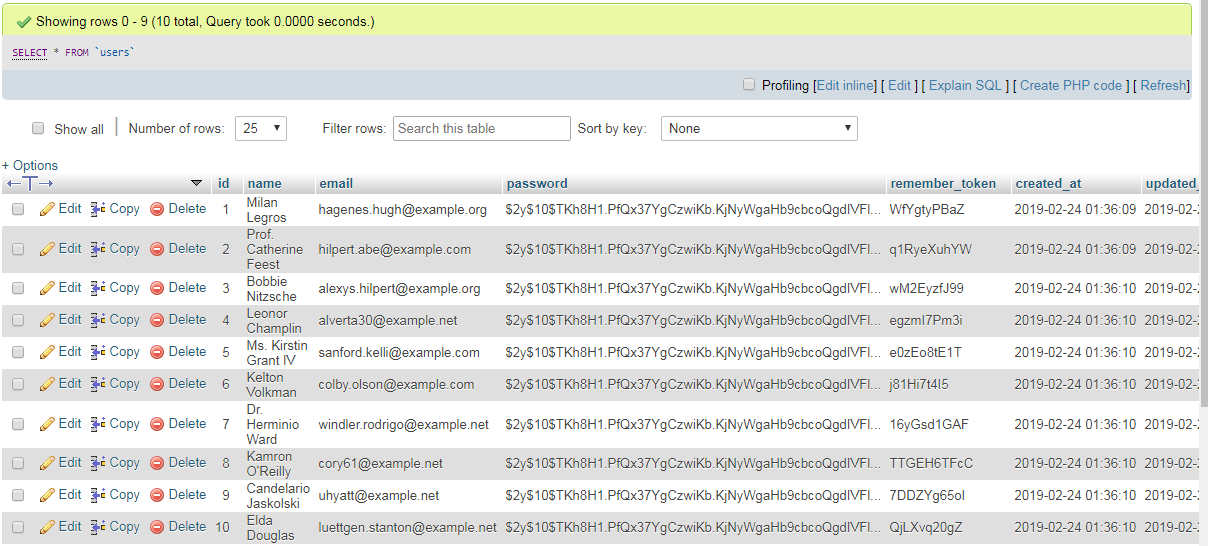
php artisan db:seed

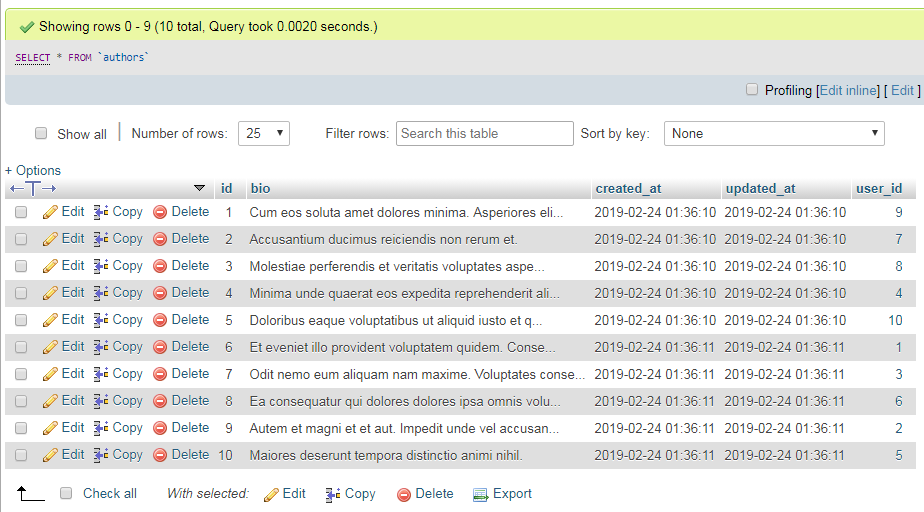




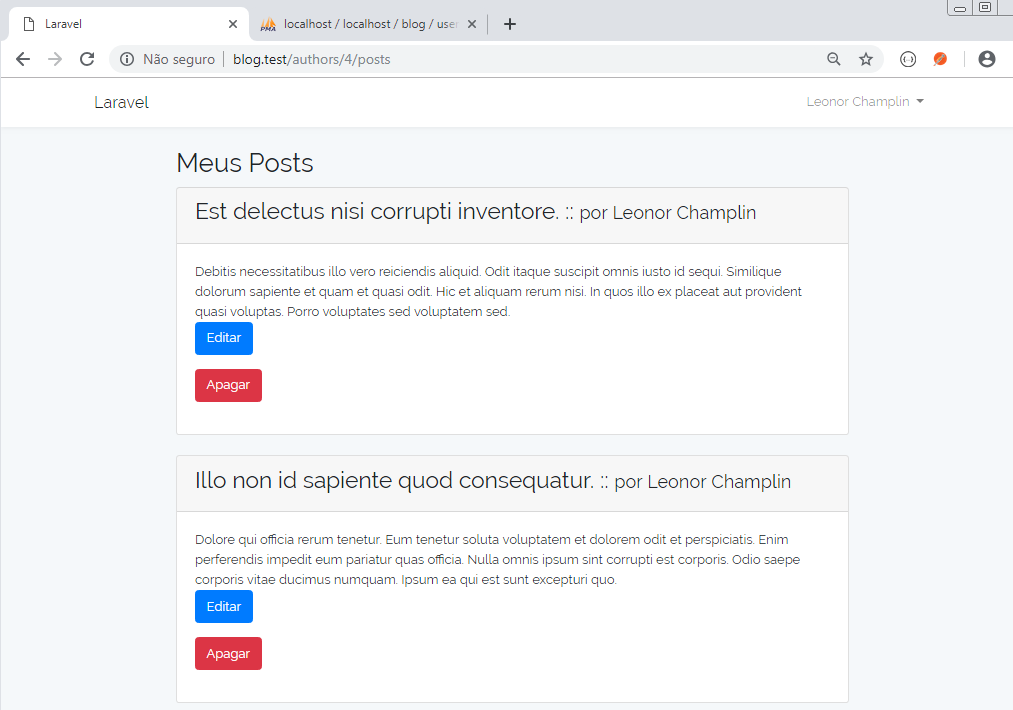
A senha é: secret







O id do autor é 4



## Ajustar a migration para que possamos apagar o post e seus comentários

Temos que incluir o "onDelete('cascade')", para que os comentários sejam apagados junto com o post.

**database/migrations/2019\_02\_22\_043021\_create\_comments\_table.php**

<?php

use Illuminate\Support\Facades\Schema;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateCommentsTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->increments('id');

$table->string('author\_email');

$table->text('content');

$table->integer('post\_id')->unsigned();

$table->timestamps();

$table->foreign('post\_id')->references('id')->on('posts')->onDelete('cascade');

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

{

Schema::dropIfExists('comments');

}

}

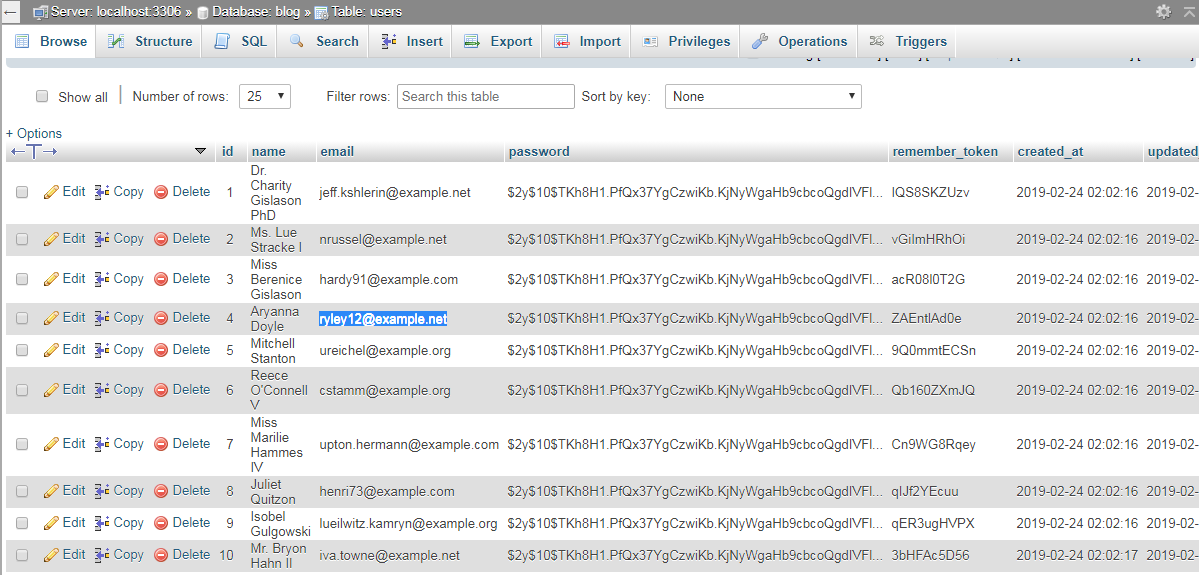
Zerar novamente o banco de dados e em seguida popular novamente.

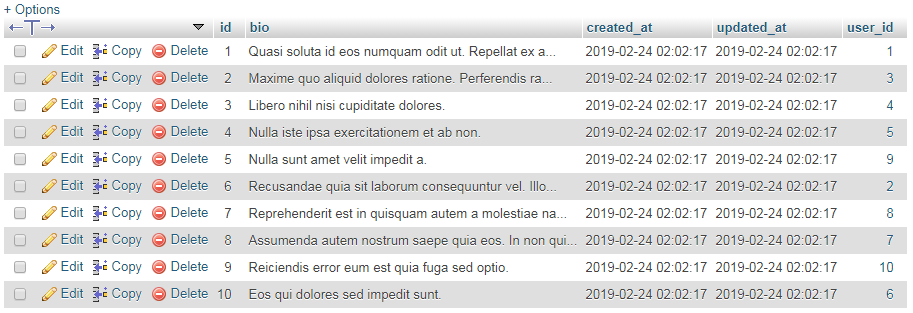
php artisan migrate:fresh

php artisan db:seed



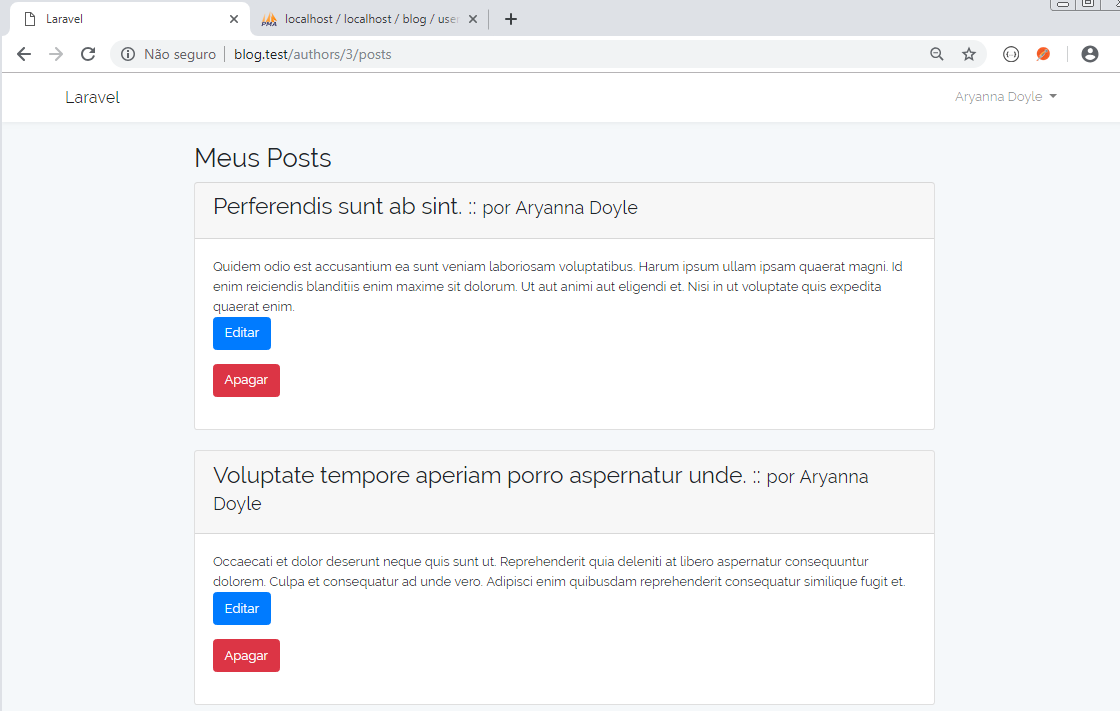
A senha é: secret



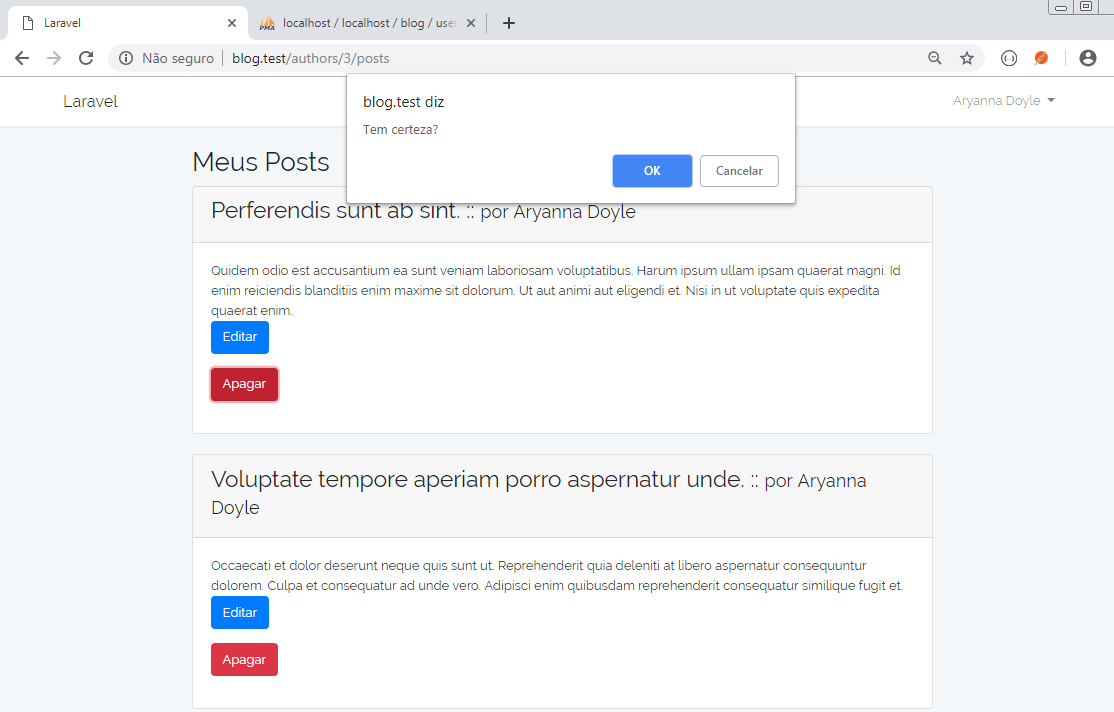


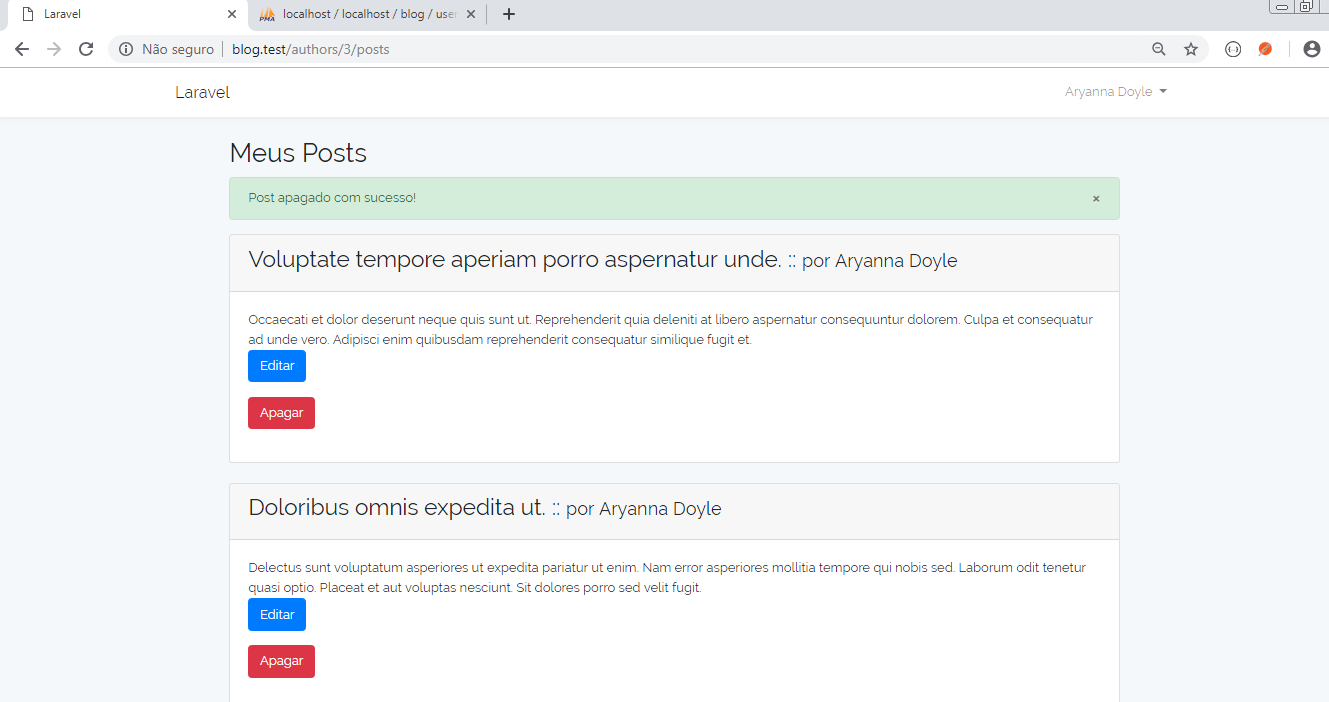
O id do autor é 3

**http://blog.test/authors/3/posts**



Apagando o primeiro post





**resources/views/authors/post.blade.php**

@extends('layouts.app')

@section('content')

<div class="row justify-content-center">

<div class="col-md-8">

<h2>Meus Posts</h2>

<div class="flash-message">

@foreach (['danger', 'warning', 'success', 'info'] as $msg)

@if(Session::has('alert-' . $msg))

<p class="alert alert-{{ $msg }}">{{ Session::get('alert-' . $msg) }} <a href="#" class="close" data-dismiss="alert" aria-label="fechar">&times;</a></p>

@endif

@endforeach

</div>

@foreach ($posts as $post)

<div class="card card-default">

<div class="card-header">

<h3>{{ $post->title }} :: <small>por {{ $post->author->user->name }}</small></h3>

</div>

<div class="card-body">

<p>

{{ $post->content }} <br>

@if(Auth::check() && Auth::user()->author->id==$post->author\_id)

<a class="btn btn-primary" href="{{ action('PostController@edit', $post->id) }}" title="Editar o post">Editar</a><br>

<form method="post" action="{{ action('PostController@destroy', $post->id) }}">

{{ csrf\_field() }}

{{ method\_field('delete') }}

<button type="submit" class="btn btn-danger delete-button" onclick="return confirm('Tem certeza?');")>Apagar</button>

</form>

@endif

</p>

</div>

</div>

<br>

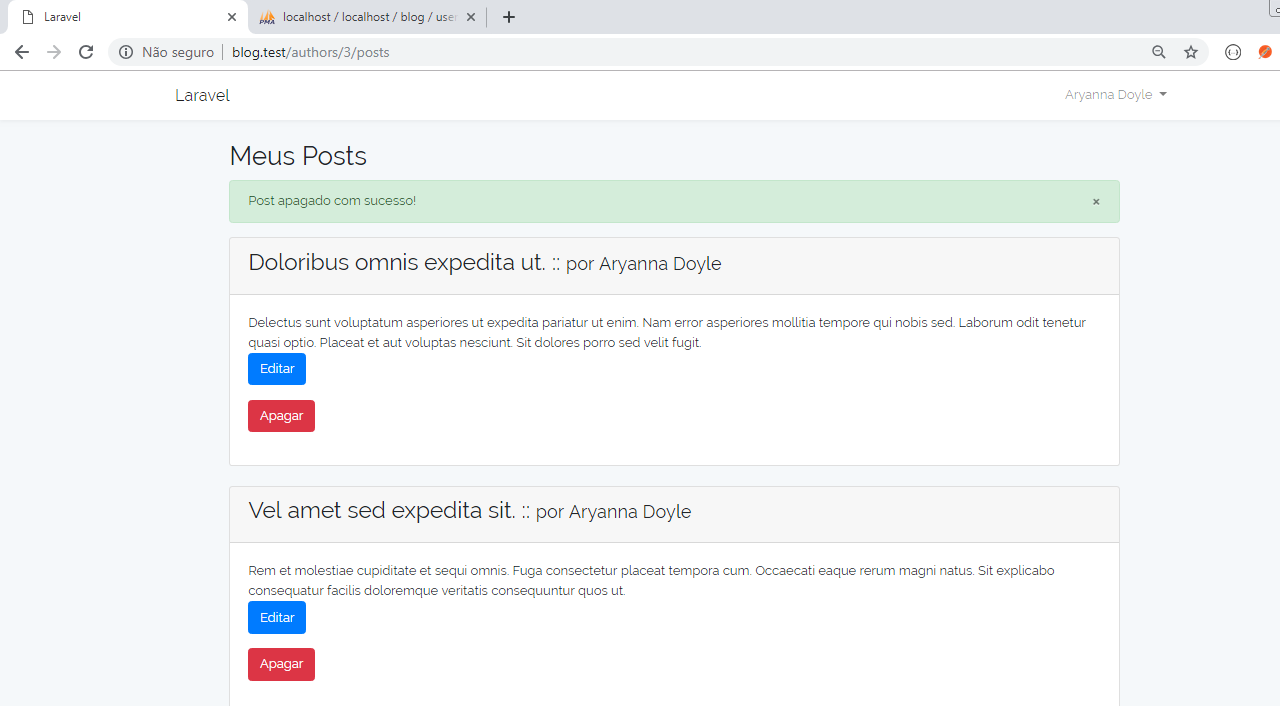
@endforeach

</div>

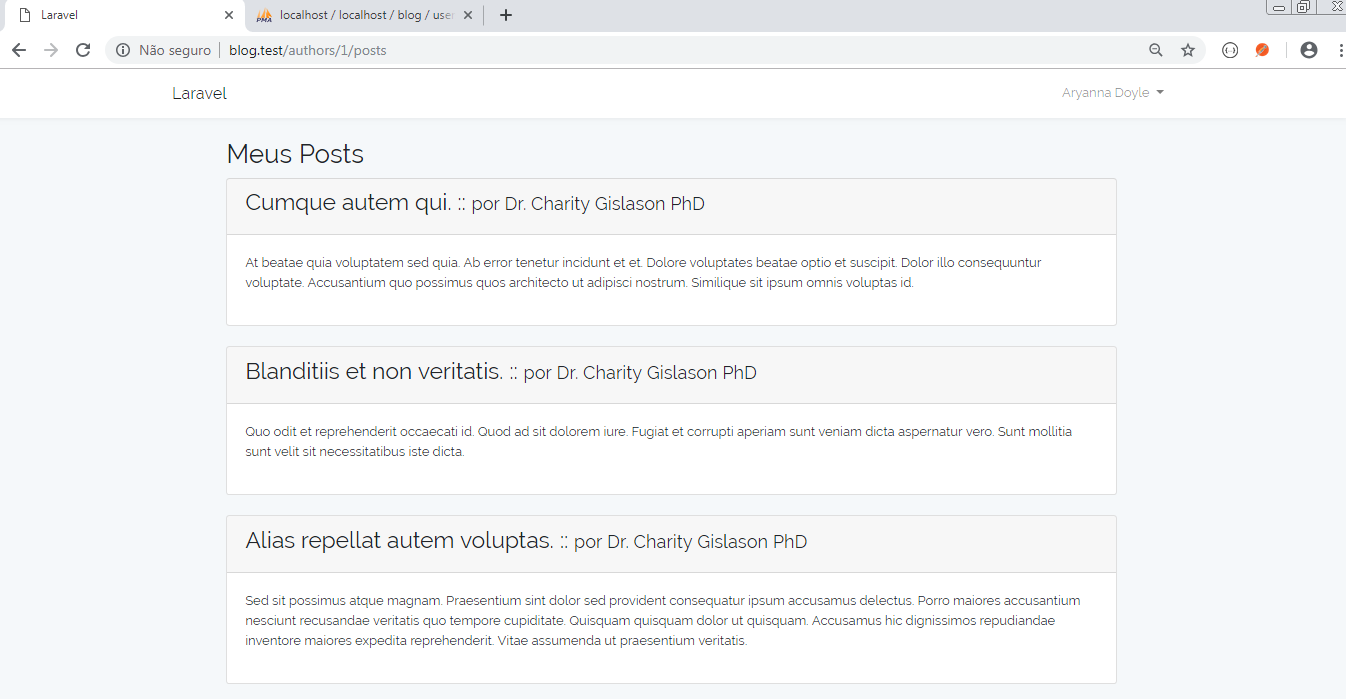
</div>

@endsection

Apague o primeiro post exibido



Entre em um outro id (exemplo: 1) para apagar e os botões não devem ser exibidos.



## Alterar a view dos posts do autor

**resources/views/authors/posts**

@extends('layouts.app')

@section('content')

<div class="row justify-content-center">

<div class="col-md-8">

<h2>Meus Posts</h2>

<div class="flash-message">

@foreach (['danger', 'warning', 'success', 'info'] as $msg)

@if(Session::has('alert-' . $msg))

<p class="alert alert-{{ $msg }}">{{ Session::get('alert-' . $msg) }} <a href="#" class="close" data-dismiss="alert" aria-label="fechar">&times;</a></p>

@endif

@endforeach

</div>

@foreach ($posts as $post)

<div class="card card-default">

<div class="card-header">

<h3>{{ $post->title }} :: <small>por {{ $post->author->user->name }}</small></h3>

</div>

<div class="card-body">

<p>

{{ $post->content }} <br>

<a class="btn btn-primary" href="{{ action('PostController@edit', $post->id) }}" title="Editar o post">Editar</a><br>

<form method="post" action="{{ action('PostController@destroy', $post->id) }}">

{{ csrf\_field() }}

{{ method\_field('delete') }}

<button type="submit" class="btn btn-danger delete-button" onclick="return confirm('Tem certeza?');")>Apagar</button>

</form>

</p>

</div>

</div>

<br>

@endforeach

</div>

</div>

@endsection

# Aula 11 - Editando posts do autor

## Implementar método update no PostController

**app/Http/Controllers/PostController.php**

public function update(Request $request, Post $post)

{

$post->title = $request->title;

$post->content = $request->content;

$post->save();

$request->session()->flash('alert-success', 'Post alterado com sucesso!');

return redirect("authors/$post->author\_id/posts");

}

## Alterar a rota para a home do autor

### Adicionar o método login ao AuthorController

/\*\*

\* Redirect logged author to posts

\*

\* @param \App\Author $author

\* @return \Illuminate\Http\Response

\*/

public function login()

{

$author = \Auth::user()->author;

$posts = $author->posts;

return view('authors.posts', compact('posts'));

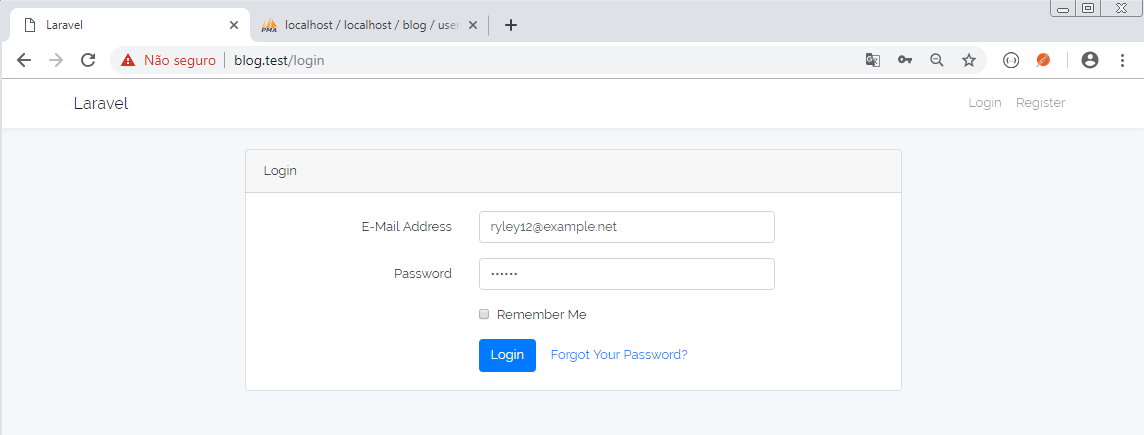
}

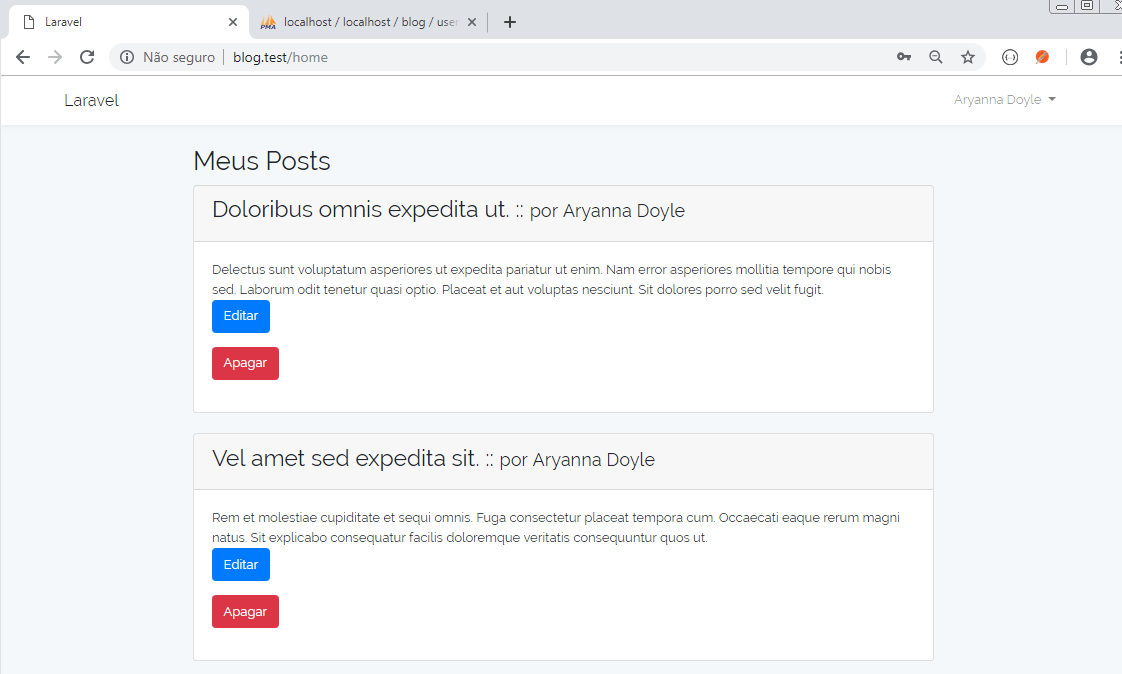
## Alterar a rota /home para o autor logado

Route::get('/home', 'AuthorController@login');

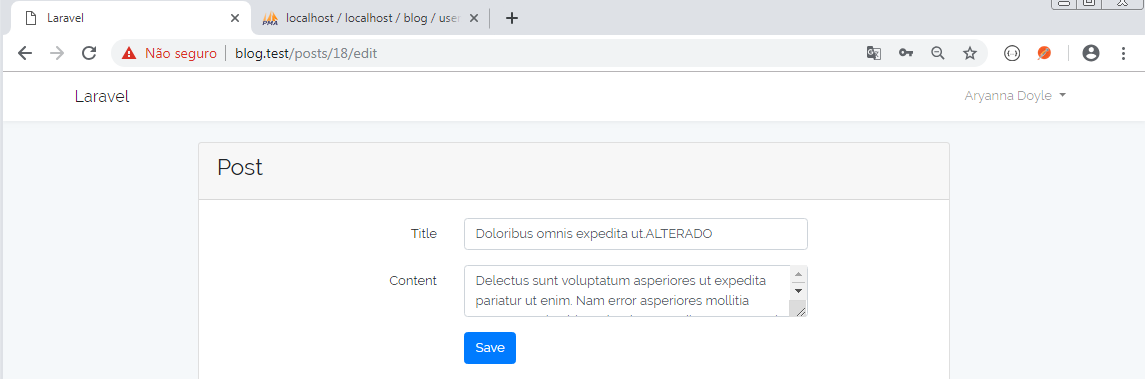
## Alterar a rota / para a lista de todos os posts (sem login)

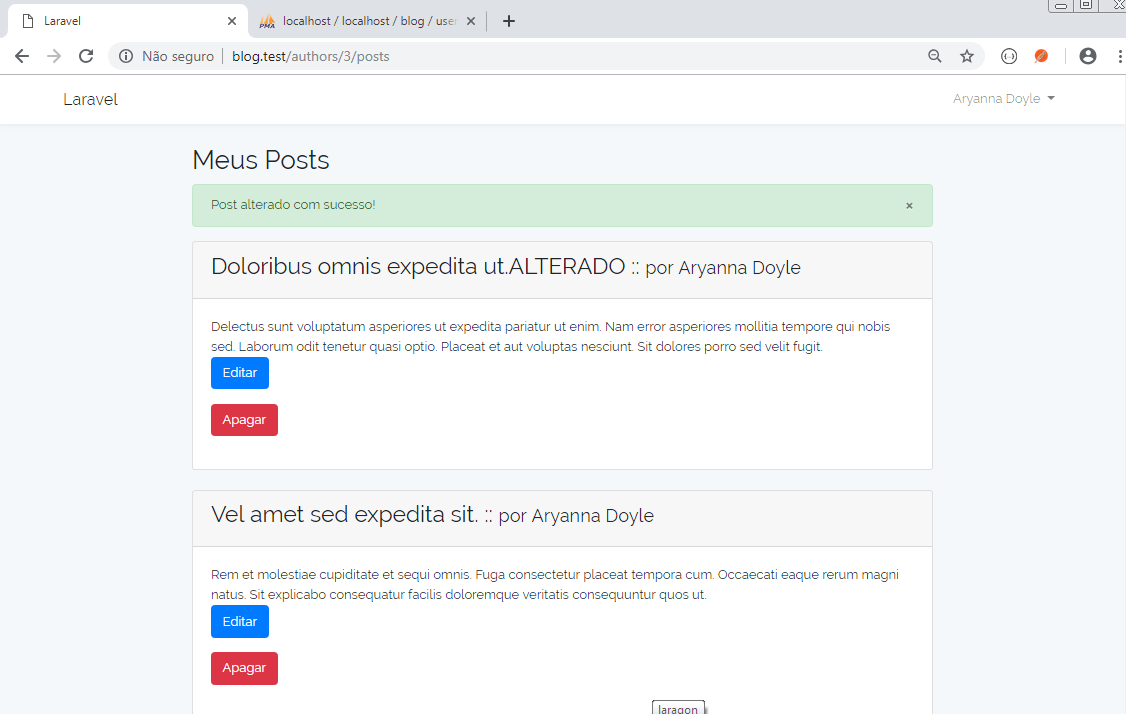
Route::get('/', 'PostController@index');



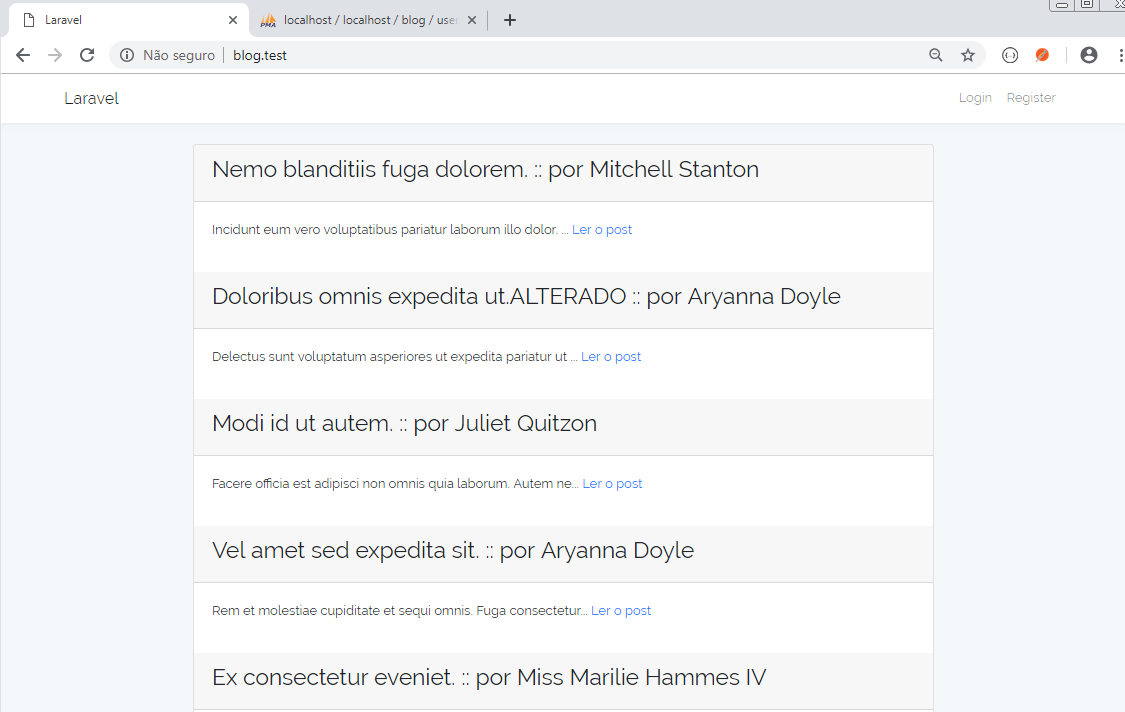


Editando o primeiro post:





Ao fazer logout:



# Aula 12 - Deletando comentários

## Implementar método destroy no CommentController

**app/Http/Controllers/CommentController.php**

public function destroy(Request $request, Comment $comment)

{

$comment->delete();

$request->session()->flash('alert-success', 'Comentário apagado com sucesso!');

return redirect()->back();

}

## Adicionar a rota para o método destroy do CommentController

**routes/web.php**

Route::delete('comments/{comment}', 'CommentController@destroy');

## Adicionar link para mostrar o post e seus comentários na view de posts do autor

**resources/views/authors/posts.blade.php**

@extends('layouts.app')

@section('content')

<div class="row justify-content-center">

<div class="col-md-8">

<h2>Meus Posts</h2>

<div class="flash-message">

@foreach (['danger', 'warning', 'success', 'info'] as $msg)

@if(Session::has('alert-' . $msg))

<p class="alert alert-{{ $msg }}">{{ Session::get('alert-' . $msg) }} <a href="#" class="close" data-dismiss="alert" aria-label="fechar">&times;</a></p>

@endif

@endforeach

</div>

@foreach ($posts as $post)

<div class="card card-default">

<div class="card-header">

<h3>{{ $post->title }} :: <small>por {{ $post->author->user->name }}</small></h3>

</div>

<div class="card-body">

<p>

{{ $post->content }} <br>

<a href="{{ action('PostController@show', $post->id) }}" title="Ver post e comentários">Ver post e comentários</a><br>

<a class="btn btn-primary" href="{{ action('PostController@edit', $post->id) }}" title="Editar o post">Editar</a><br>

<form method="post" action="{{ action('PostController@destroy', $post->id) }}">

{{ csrf\_field() }}

{{ method\_field('delete') }}

<button type="submit" class="btn btn-danger delete-button" onclick="return confirm('Tem certeza?');")>Apagar</button>

</form>

</p>

</div>

</div>

<br>

@endforeach

</div>

</div>

@endsection

## Alterar a view show do post para mostrar o botão deletar só para o autor do post

**resources/views/posts/show.blade.php**

@extends('layouts.app')

@section('content')

<div class="row justify-content-center">

<div class="col-md-8">

<div class="card card-default">

<div class="flash-message">

@foreach (['danger', 'warning', 'success', 'info'] as $msg)

@if(Session::has('alert-' . $msg))

<p class="alert alert-{{ $msg }}">{{ Session::get('alert-' . $msg) }} <a href="#" class="close" data-dismiss="alert" aria-label="fechar">&times;</a></p>

@endif

@endforeach

</div>

<div class="card-header">

<h3>{{ $post->title }} :: <small>por {{ $post->author->user->name }}</small></h3>

</div>

<div class="card-body">

<p>

{{ $post->content }}

</p>

<h4>Comentários:</h4>

<!-- Comentários do post -->

@foreach($post->comments as $comment)

<div class="card card-default">

<div class="card-header">

{{ $comment->author\_email }}

</div>

<div class="card-body">

{{ $comment->content }}

@if (Auth::check() && Auth::user()->author->id == $post->author\_id)

<form method="post" action="{{ action('CommentController@destroy', $comment->id) }}">

{{ csrf\_field() }}

{{ method\_field('delete') }}

<button type="submit" class="btn btn-danger delete-button" onclick="return confirm('Tem certeza?');")>Apagar Comentário</button>

</form>

@endif

</div>

</div>

<br>

@endforeach

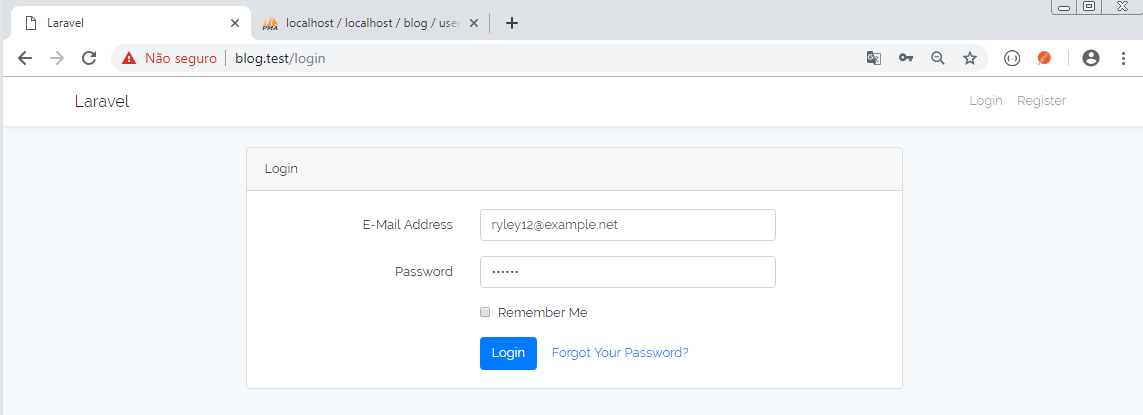
</div>

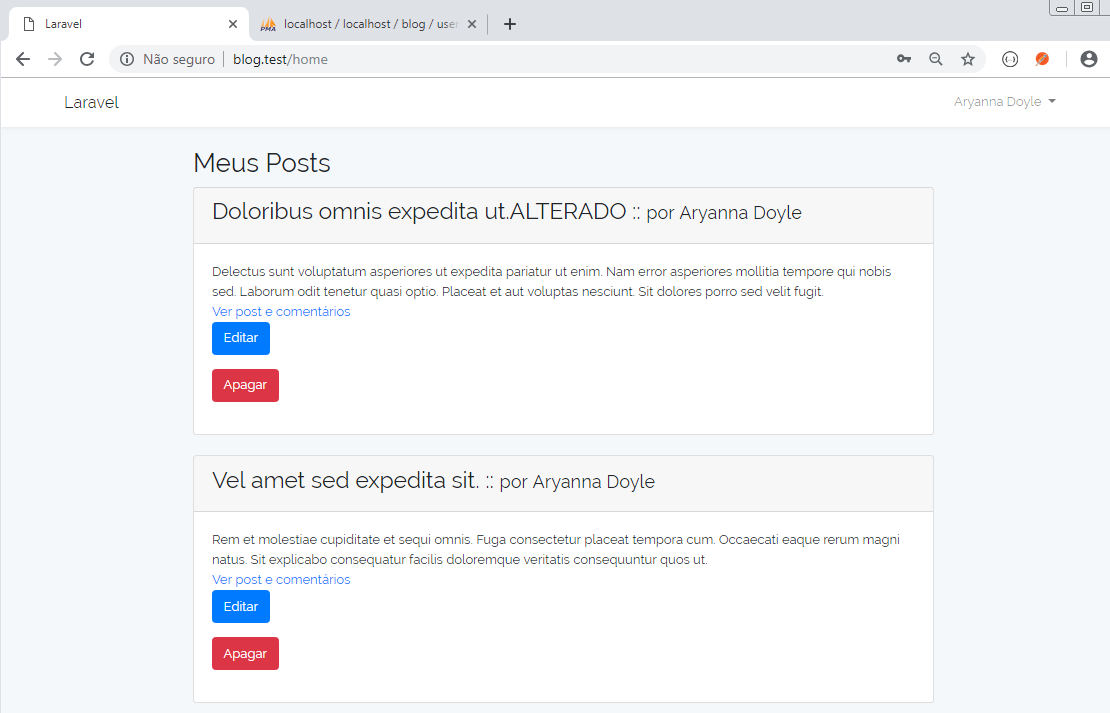
</div>

</div>

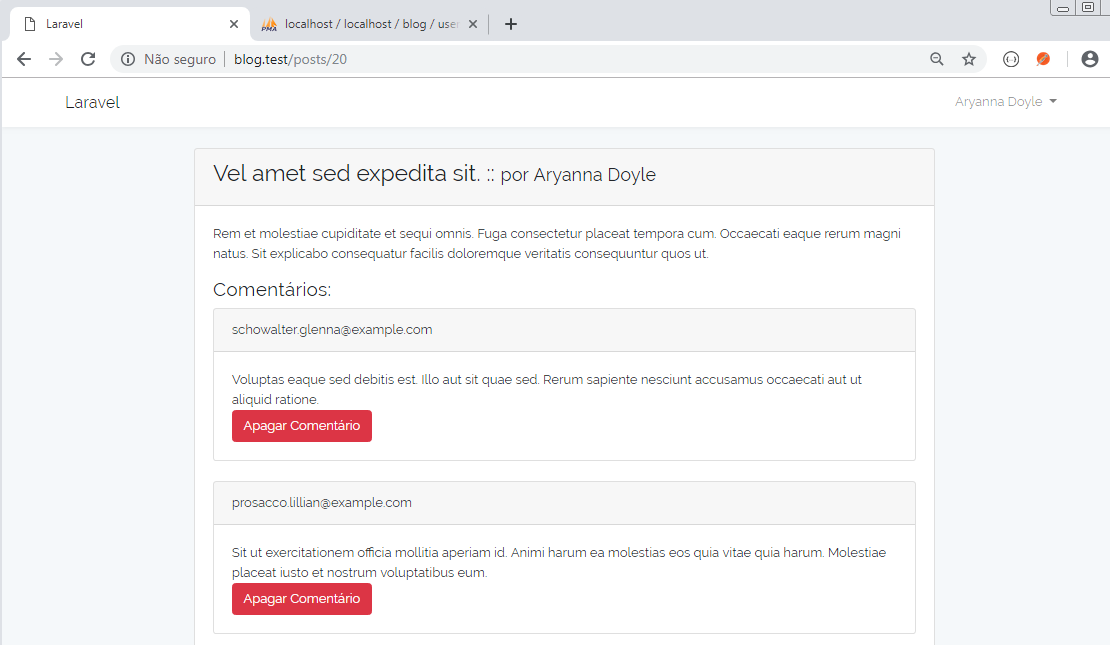
</div>

@endsection

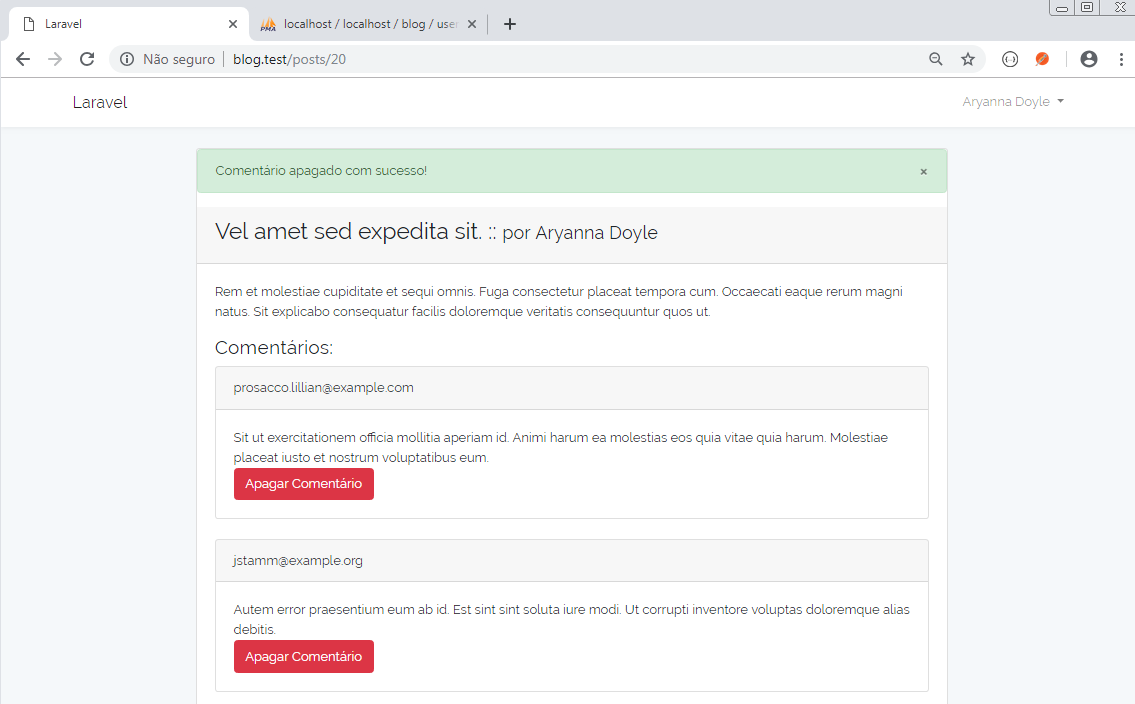




Clique no link "Ver post e comentário" do segundo post:



Apague o primeiro comentário:



# Aula 13 - Menu da aplicação

## Colocar os links no menu da aplicação

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

<!DOCTYPE html>

<html lang="{{ app()->getLocale() }}">

<head>

<meta charset="utf-8">

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

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

<!-- CSRF Token -->

<meta name="csrf-token" content="{{ csrf\_token() }}">

<title>{{ config('app.name', 'Laravel') }}</title>

<!-- Styles -->

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

</head>

<body>

<div id="app">

<nav class="navbar navbar-expand-md navbar-light navbar-laravel">

<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">

<li>

<a class="nav-link" href="/">Todos os posts</a>

</li>

@auth

<li>

<a class="nav-link"href="/home">Meus posts</a>

</li>

@endauth

<!-- Authentication Links -->

@guest

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

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

@else

<li class="nav-item dropdown">

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

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

</a>

<div class="dropdown-menu" 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>

<main class="py-4">

@yield('content')

</main>

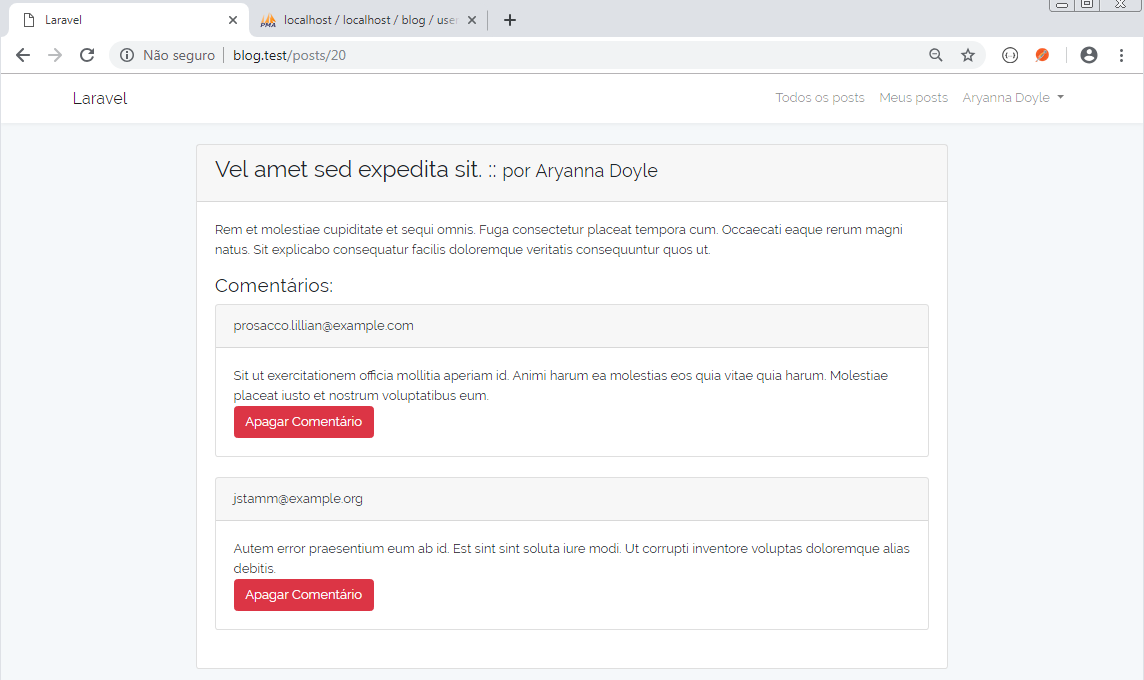
</div>

<!-- Scripts -->

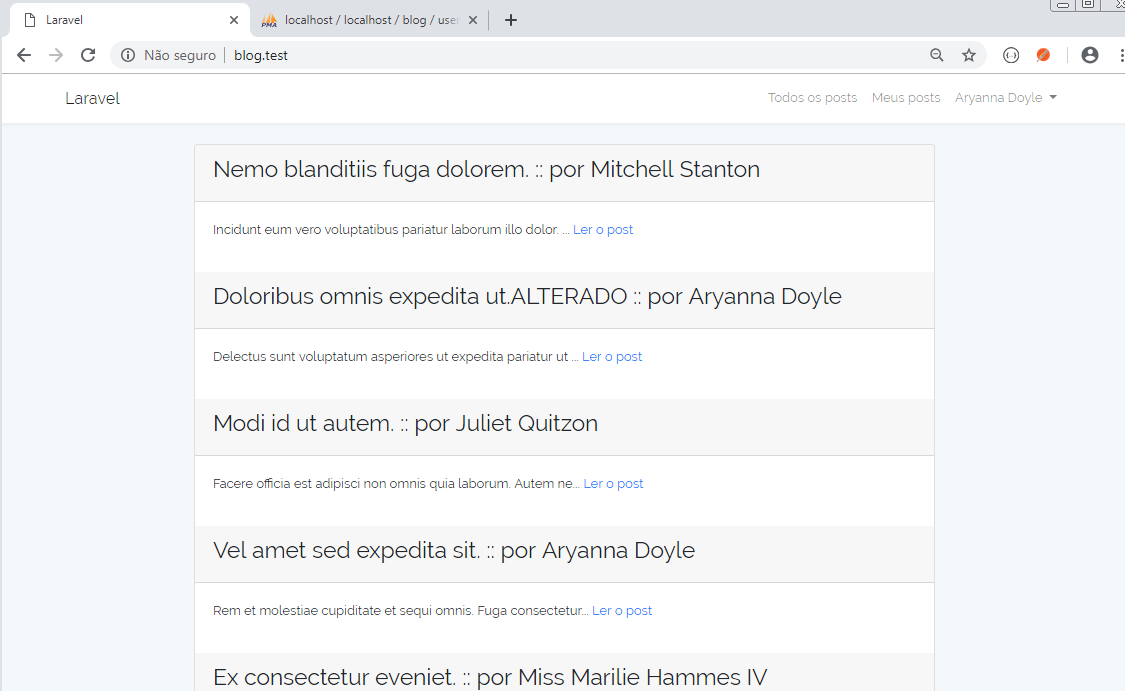
<script src="{{ asset('js/app.js') }}"></script>

</body>

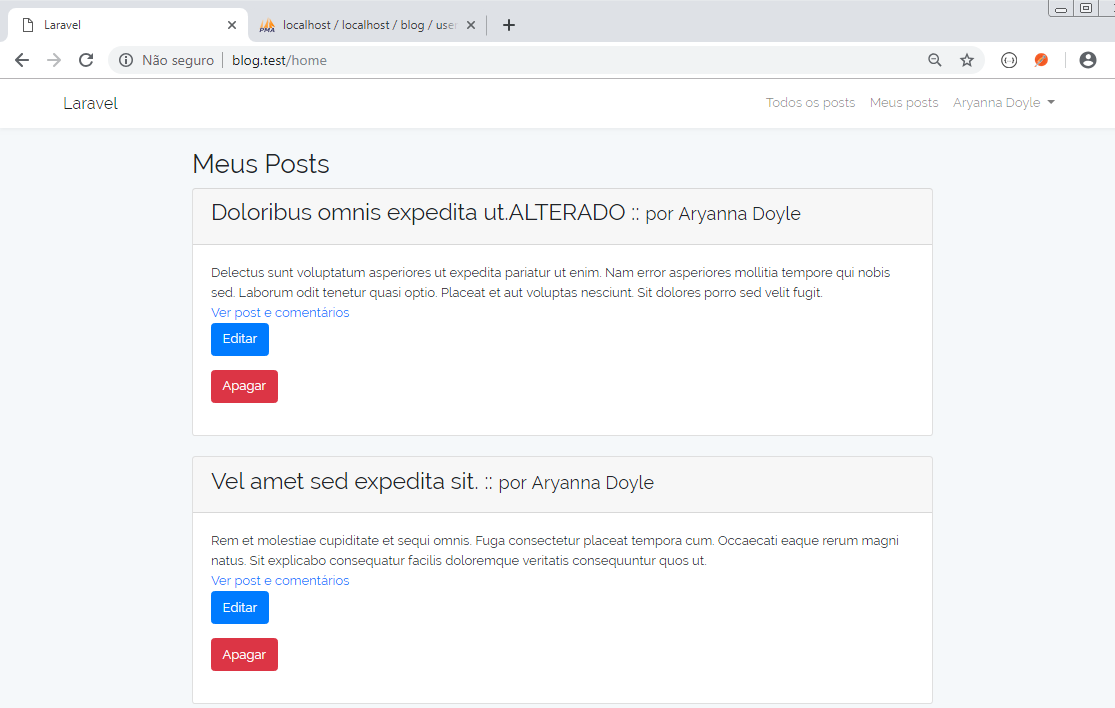
</html>



Clicando no link "Todos os posts":



Clicando no link "Meus posts":



Clicando em "Logout":

