**Curso Laravel 5.2 com ACL**

**EspecializaTi (Carlos Ferreira)**

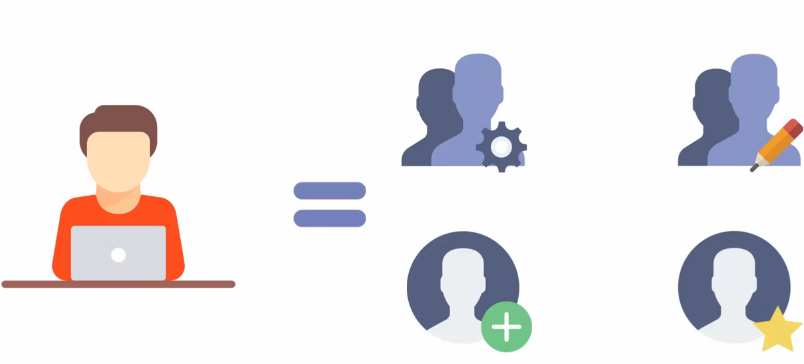
<https://www.youtube.com/watch?v=-AVa6GNpEL8&list=PLVSNL1PHDWvTch1r8uTSluw9SkzSA9cDJ>

Resumo do curso feito por Roberto Pinheiro

# Aula 01 - Apresentação do curso



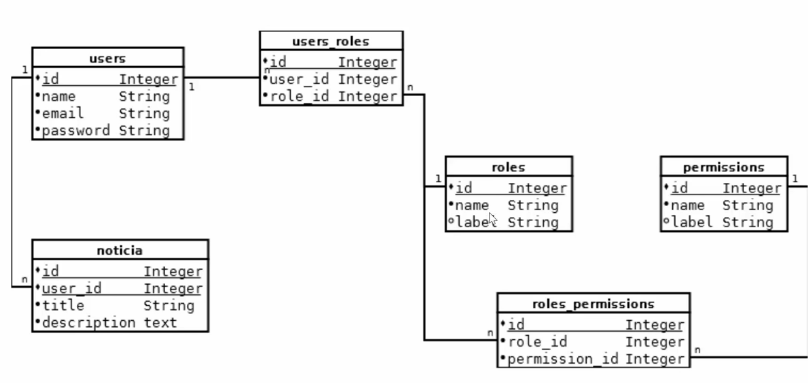
### Possui nível de acesso a usuários.



### Funções permitidas para cada nível de usuário

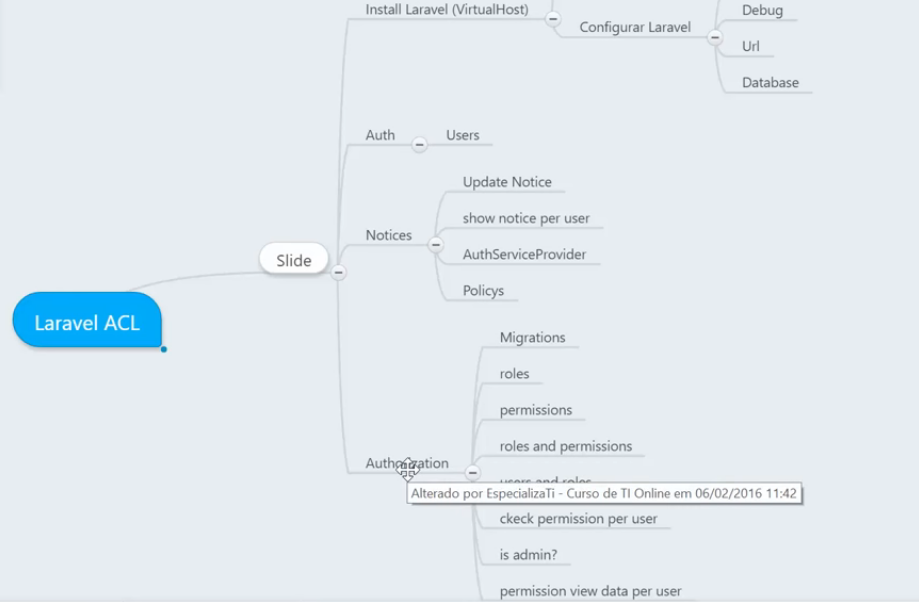


### Relacionamento entre as tabelas da aplicação

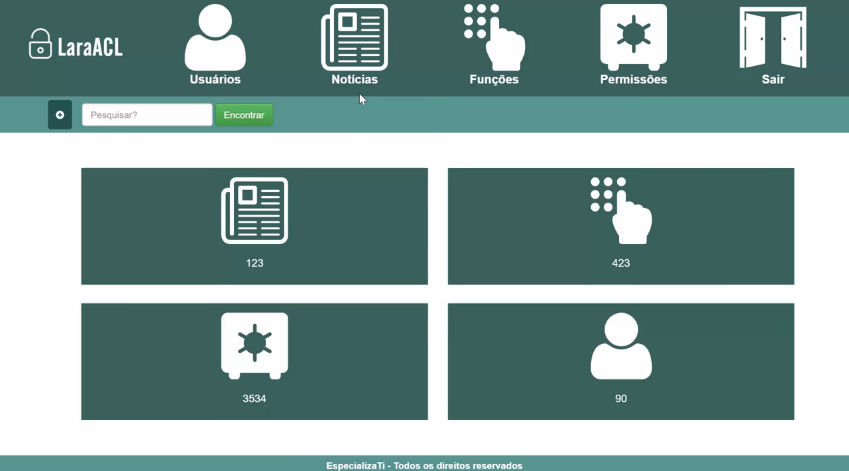


# Aula 02 - Organização do Curso Laravel com ACL

## Mapa mental







# Aula 03 - Instalando Laravel e configurando Virtual Host

## Instalando o Laravel e criando o projeto

composer create-project --prefer-dist laravel/laravel laravel-acl

## Configurando Virtual Host

**c:/Windows/System32/drivers/etc/hosts**

# Copyright (c) 1993-2009 Microsoft Corp.

#

# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.

#

# This file contains the mappings of IP addresses to host names. Each

# entry should be kept on an individual line. The IP address should

# be placed in the first column followed by the corresponding host name.

# The IP address and the host name should be separated by at least one

# space.

#

# Additionally, comments (such as these) may be inserted on individual

# lines or following the machine name denoted by a '#' symbol.

#

# For example:

#

# 102.54.94.97 rhino.acme.com # source server

# 38.25.63.10 x.acme.com # x client host

# localhost name resolution is handled within DNS itself.

# 127.0.0.1 localhost

# ::1 localhost

127.0.0.1 localhost

::1 localhost

127.0.0.1 appLogin.test #laragon magic!

127.0.0.1 blog.test #laragon magic!

127.0.0.1 laravel-acl.test #laragon magic!

127.0.0.1 laravel53-basico.test #laragon magic!

127.0.0.1 NewsApp.test #laragon magic!

127.0.0.1 primeiro.test #laragon magic!

127.0.0.1 rest-laravel.test #laragon magic!

127.0.0.1 rest\_api.test #laragon magic!

127.0.0.1 SistemaUsuarios.test #laragon magic!

**config/app.php**

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

**.env**

APP\_NAME=Laravel

APP\_ENV=local

APP\_KEY=base64:M29FqBdy1ir30TerdGYjv3YPRsMN+3k8vDvIgM+stTw=

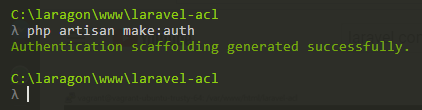
APP\_DEBUG=true

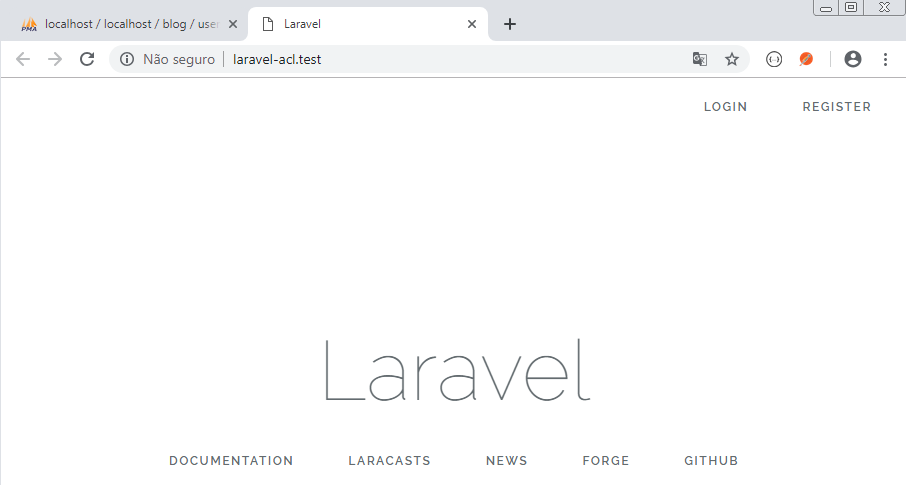
APP\_URL=http://laravel-acl.test

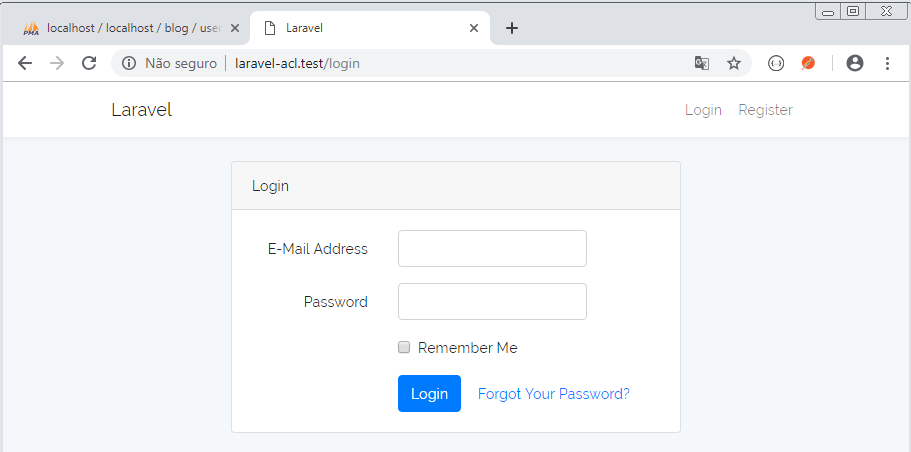
# Aula 04 - Auth Laravel

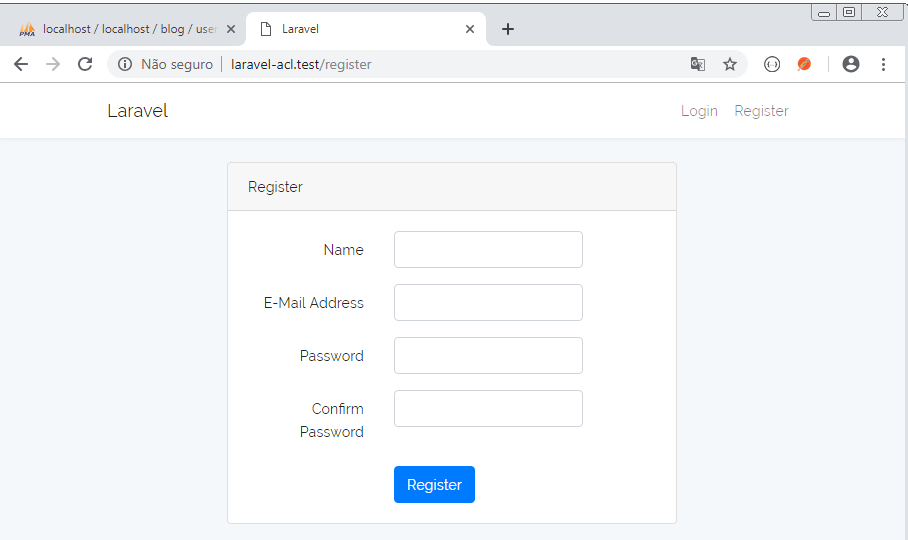
## Criando sistema de autenticação com o Laravel

php artisan make:auth



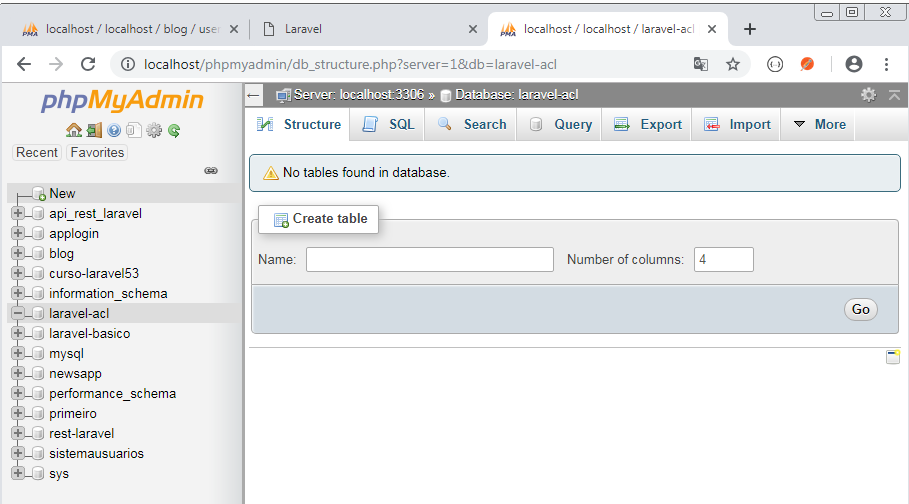






## Criando o banco de dados

Crie o banco de dados chamado laravel-acl



## Configurando o acesso ao banco de dados

Abrir o arquivo .env e fazer a seguinte configuração:

DB\_CONNECTION=mysql

DB\_HOST=localhost

DB\_PORT=3306

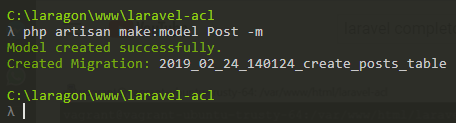
DB\_DATABASE=laravel-acl

DB\_USERNAME=root

DB\_PASSWORD=

## Criar model e migration

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



**database/migrations/2019\_02\_24\_140124\_create\_posts\_table.php**

<?php

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

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

$table->string('title', 200);

$table->text('description');

$table->timestamps();

});

}

public function down()

{

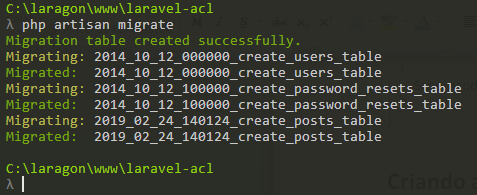
Schema::drop('posts');

}

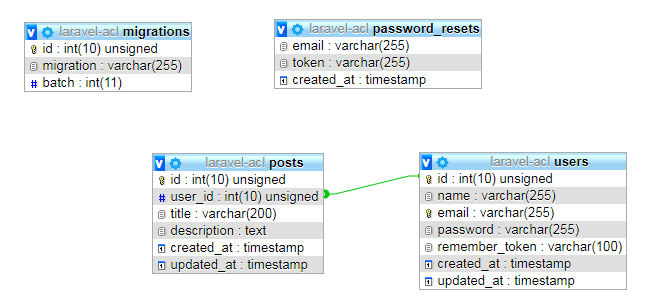
}

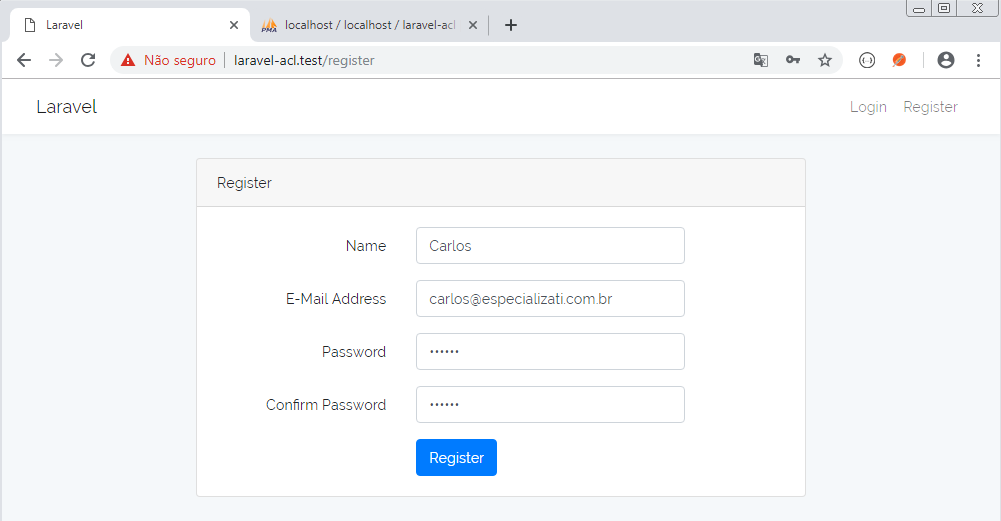
## Criando as tabelas

php artisan migrate

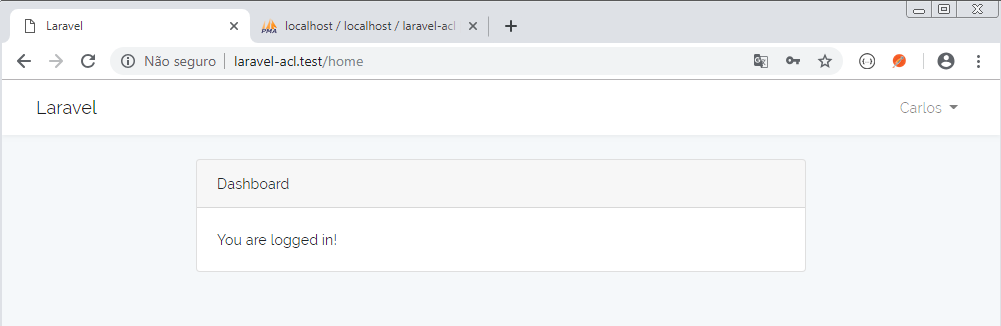


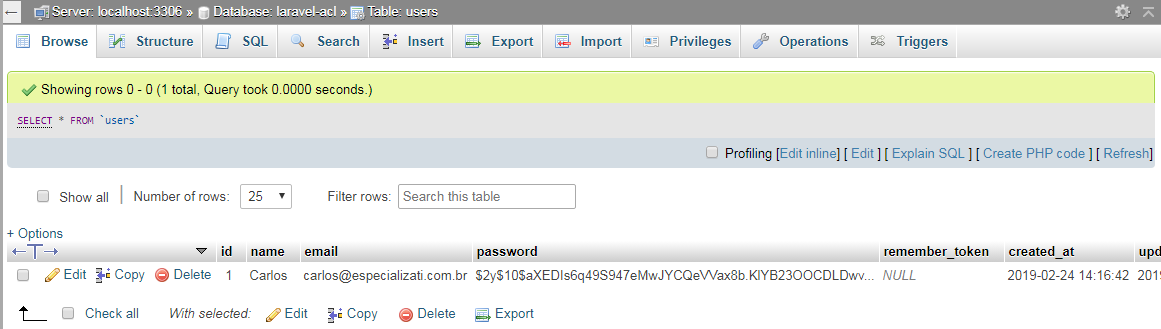
No phpmyadmin clique em Design



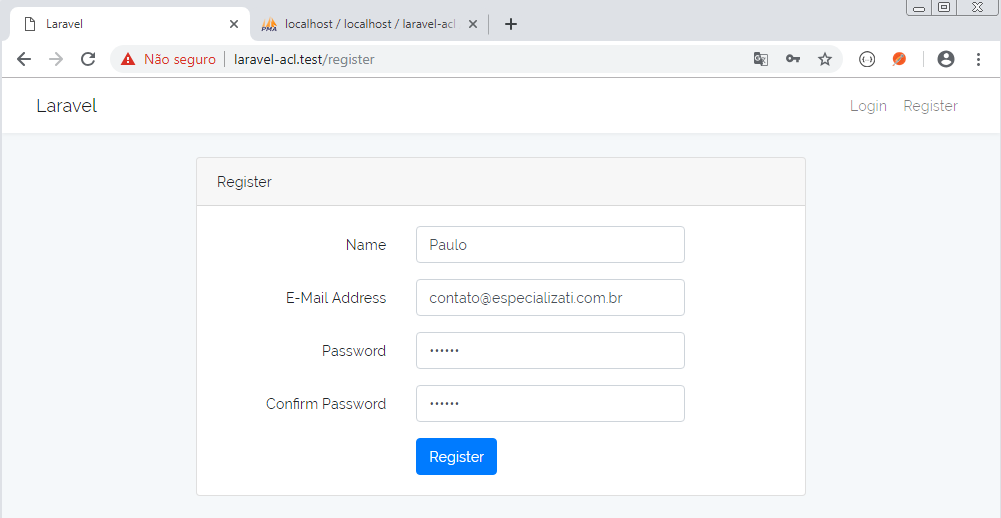


Senha: 123456

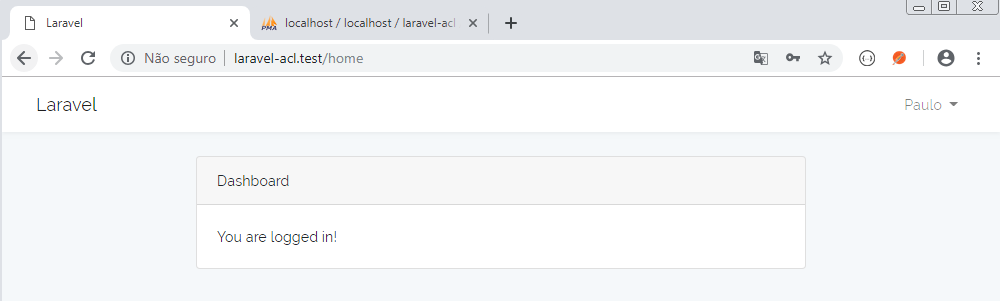




Faça logout e cadastre um novo usuário



Senha: 123456



# Aula 05 - Cadastrando Posts e listando

**app/Post.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Post extends Model

{

public function user()

{

// Relacionamento de muitos para um

return $this->belongsTo(User::class);

}

}

**app/http/Controllers/HomeController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Post;

class HomeController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function index(Post $post)

{

$posts = $post->all();

return view('home', compact('posts'));

}

}

**resources/views/home.blade.php**

@extends('layouts.app')

@section('content')

<div class="container">

@forelse($posts as $post)

<h1>{{$post->title}}</h1>

<p>{{$post->description}}</p>

<p><b>Autor: {{$post->user->name}}</b></p>

<hr>

@empty

<p>Nenhum post cadastrado</p>

@endforelse

</div>

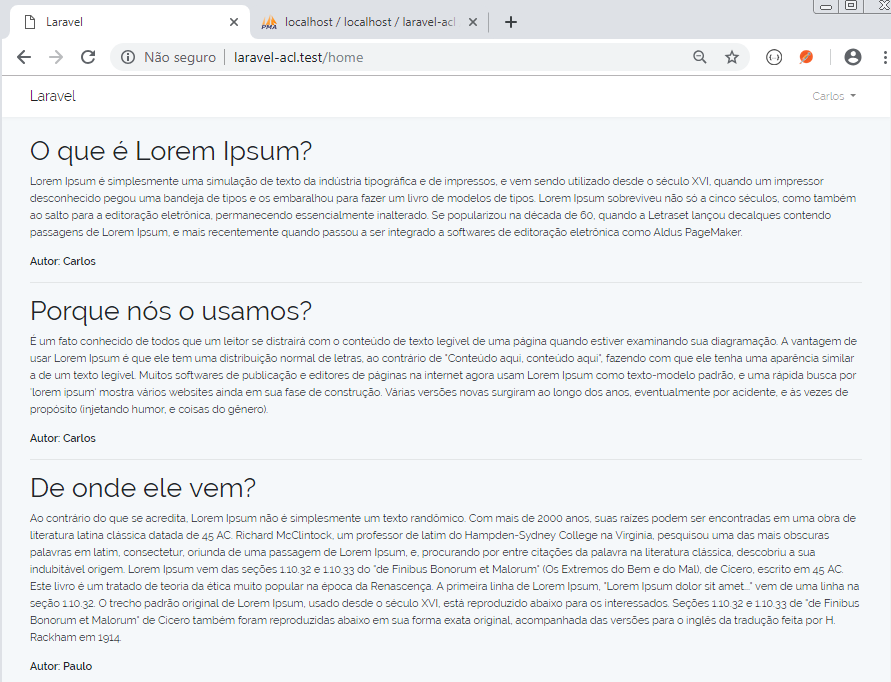
@endsection

**routes/web.php**

// Foi inserido automaticamente ao criar auth

Auth::routes();

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



# Aula 06 - Iniciando ACL com Laravel

## Exibindo apenas os posts do usuário logado

**app/http/Controllers/HomeController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Post;

class HomeController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function index(Post $post)

{

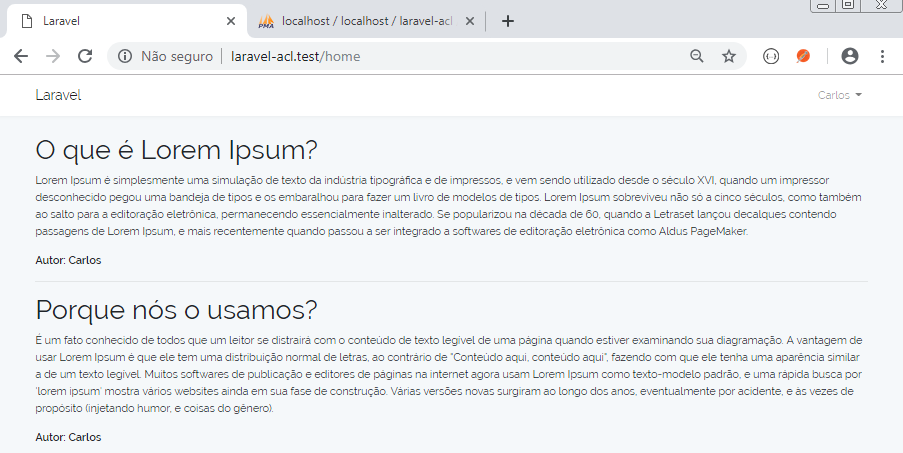
// Seleciona todos os posts do usuário logado

$posts = $post->where('user\_id', auth()->user()->id)->get();

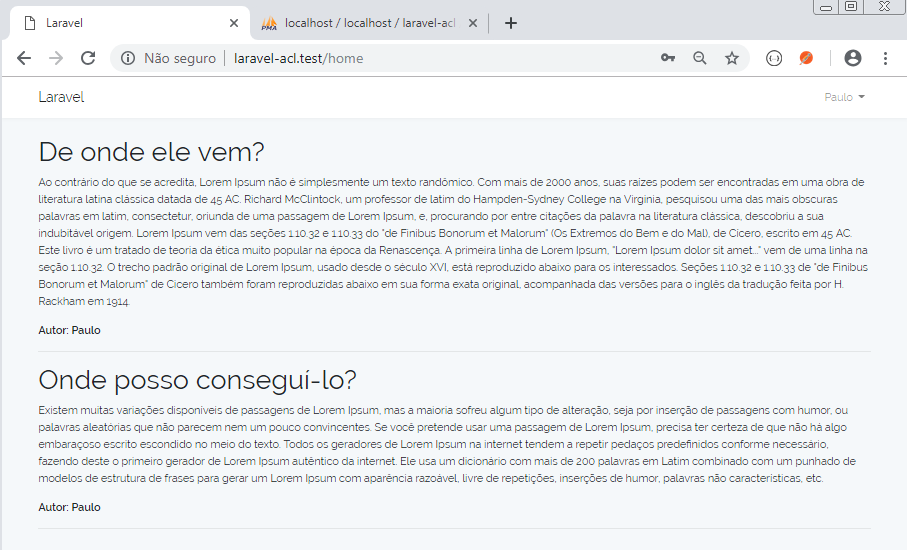
return view('home', compact('posts'));

}

}



Usuário logado como Paulo



## Preparando edição do post

**routes/web.php**

Auth::routes();

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

Route::get('/post/{id}/update', 'HomeController@update');

**resources/views/home.blade.php**

@extends('layouts.app')

@section('content')

<div class="container">

@forelse($posts as $post)

<h1>{{$post->title}}</h1>

<p>{{$post->description}}</p>

<p><b>Autor: {{$post->user->name}}</b></p>

<p><a href="{{url("/post/$post->id/update")}}">Editar</a></p>

<hr>

@empty

<p>Nenhum post cadastrado</p>

@endforelse

</div>

@endsection

**app/http/Controllers/HomeController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Post;

class HomeController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function index(Post $post)

{

// Seleciona todos os posts

// $posts = $post->all();

// Seleciona todos os posts do usuário logado

$posts = $post->where('user\_id', auth()->user()->id)->get();

return view('home', compact('posts'));

}

public function update($idpost){

$post = Post::find($idpost);

return view('post-update', compact('post'));

}

}

**resources/views/post-update.blade.php**

@extends('layouts.app')

@section('content')

<div class="container">

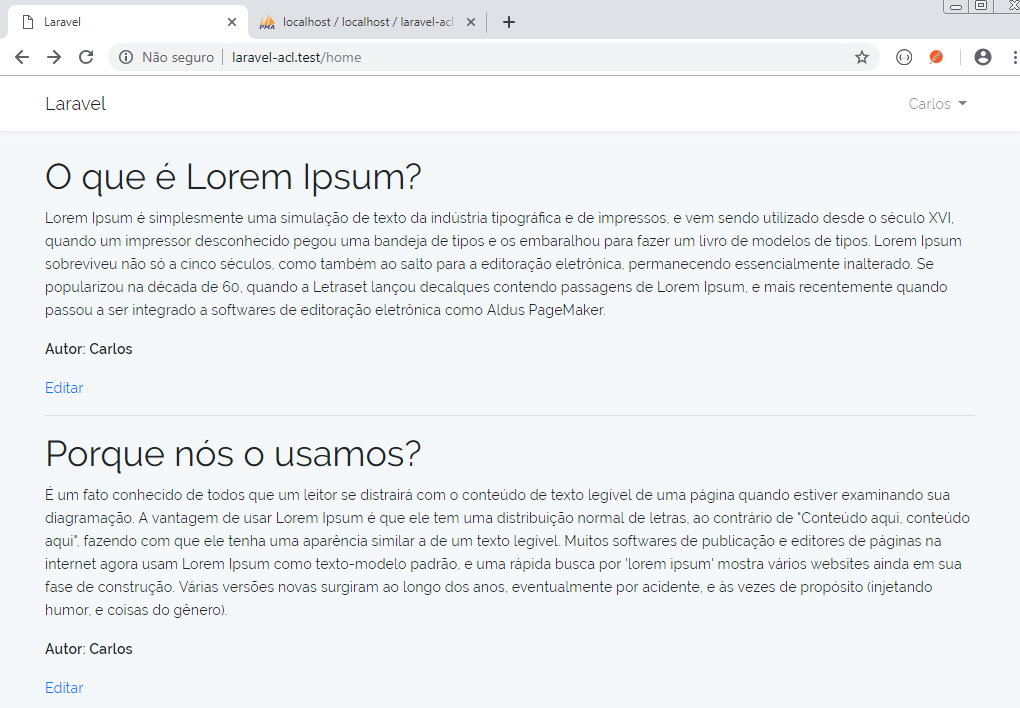
<h1>{{$post->title}}</h1>

<p>{{$post->description}}</p>

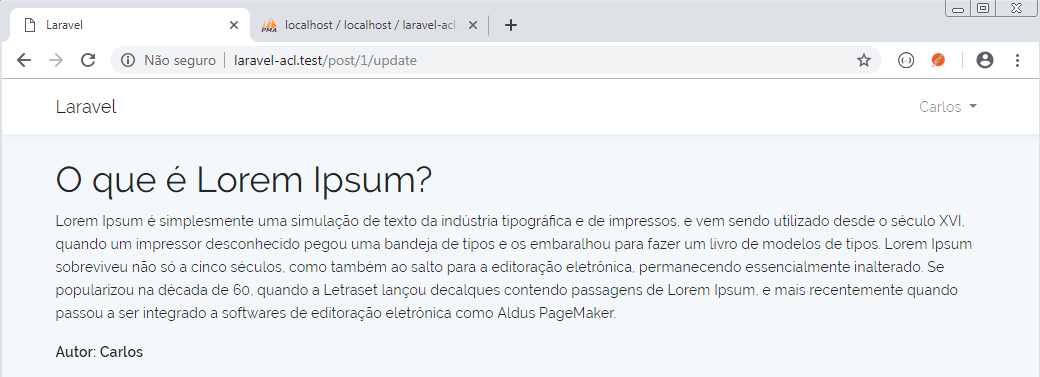
<p><b>Autor: {{$post->user->name}}</b></p>

</div>

@endsection



Clique no link "Editar" do primeiro post:



## Bloqueando edição de posts que não são do usuário

### Definindo a primeira ACL

**app/Providers/AuthServiceProvider.php**

<?php

namespace App\Providers;

// use Illuminate\Support\Facades\Gate;

use Illuminate\Contracts\Auth\Access\Gate as GateContract;

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

use App\Post;

use App\User;

class AuthServiceProvider extends ServiceProvider

{

protected $policies = [

'App\Model' => 'App\Policies\ModelPolicy',

];

public function boot(GateContract $gate)

{

$this->registerPolicies();

$gate->define('update-post', function(User $user, Post $post){

return $user->id == $post->user\_id;

});

}

}

**app/http/Controllers/HomeController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Post;

use Gate;

class HomeController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function index(Post $post)

{

// Seleciona todos os posts

// $posts = $post->all();

// Seleciona todos os posts do usuário logado

$posts = $post->where('user\_id', auth()->user()->id)->get();

return view('home', compact('posts'));

}

public function update($idpost){

$post = Post::find($idpost);

// $this->authorize('update-post', $post);

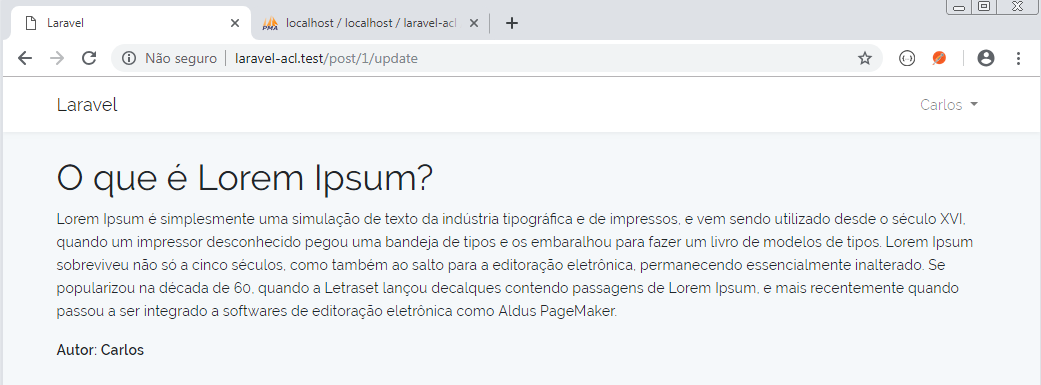
if( Gate::denies('update-post', $post) )

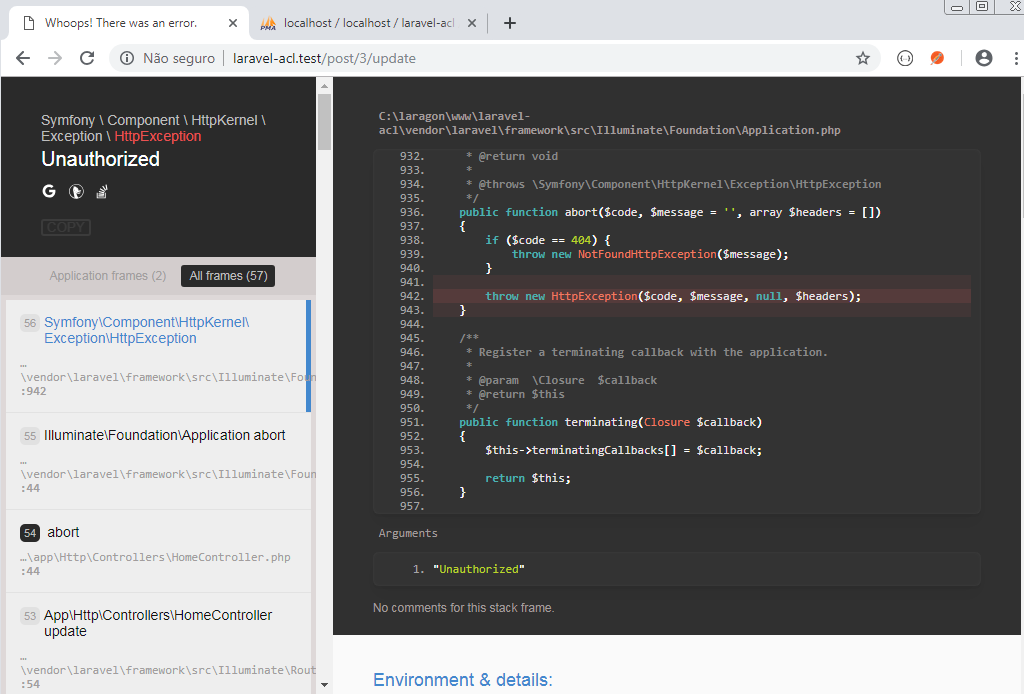
abort(403, 'Unauthorized');

return view('post-update', compact('post'));

}

}





# Aula 07 - Laravel ACL na View

### Exibindo todos os posts e permitindo editar apenas os do usuário

**app/http/Controllers/HomeController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Post;

use Gate;

class HomeController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function index(Post $post)

{

// Seleciona todos os posts

$posts = $post->all();

// Seleciona todos os posts do usuário logado

// $posts = $post->where('user\_id', auth()->user()->id)->get();

return view('home', compact('posts'));

}

public function update($idpost){

$post = Post::find($idpost);

// $this->authorize('update-post', $post);

if( Gate::denies('update-post', $post) )

abort(403, 'Unauthorized');

return view('post-update', compact('post'));

}

}

**resources/views/home.blade.php**

@extends('layouts.app')

@section('content')

<div class="container">

@forelse($posts as $post)

<h1>{{$post->title}}</h1>

<p>{{$post->description}}</p>

<p><b>Autor: {{$post->user->name}}</b></p>

@can('update-post', $post)

<p><a href="{{url("/post/$post->id/update")}}">Editar</a></p>

@endcan

<hr>

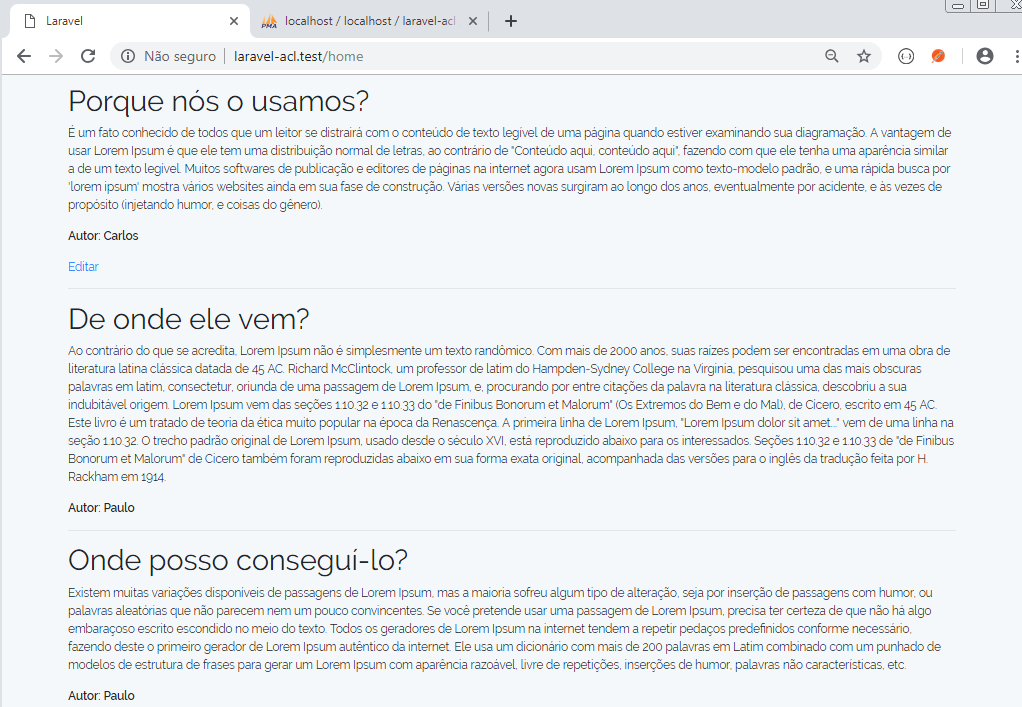
@empty

<p>Nenhum post cadastrado</p>

@endforelse

</div>

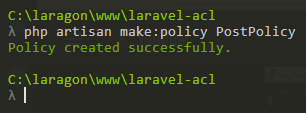
@endsection



# Aula 08 - Definindo políticas de acesso ao sistema

## Criando políticas de acesso ao sistema

php artisan make:policy PostPolicy



## Organizando as políticas de acesso

**app/Policies/PostPolicy.php**

<?php

namespace App\Policies;

use Illuminate\Auth\Access\HandlesAuthorization;

use App\User;

use App\Post;

class PostPolicy

{

use HandlesAuthorization;

public function \_\_construct()

{

//

}

public function updatePost(User $user, Post $post)

{

return $user->id == $post->user\_id;

}

}

**app/Providers/AuthServiceProvider.php**

<?php

namespace App\Providers;

// use Illuminate\Support\Facades\Gate;

use Illuminate\Contracts\Auth\Access\Gate as GateContract;

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

use App\Post;

use App\User;

class AuthServiceProvider extends ServiceProvider

{

protected $policies = [

// 'App\Model' => 'App\Policies\ModelPolicy',

\App\Post::class => \App\Policies\PostPolicy::class

];

public function boot(GateContract $gate)

{

$this->registerPolicies();

/\*

$gate->define('update-post', function(User $user, Post $post){

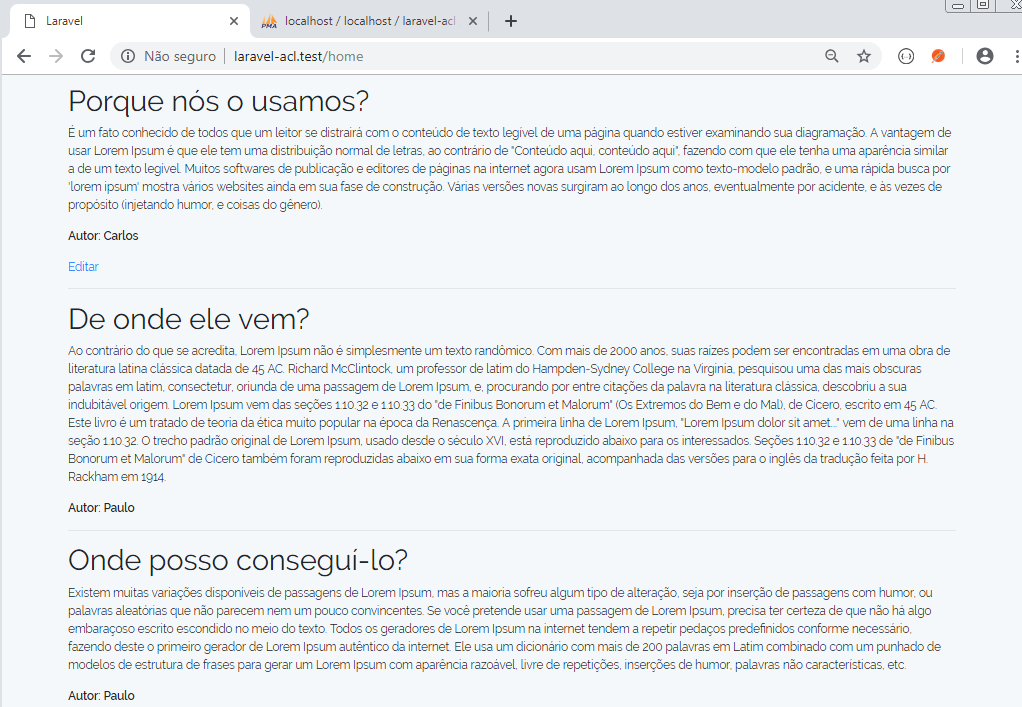
return $user->id == $post->user\_id;

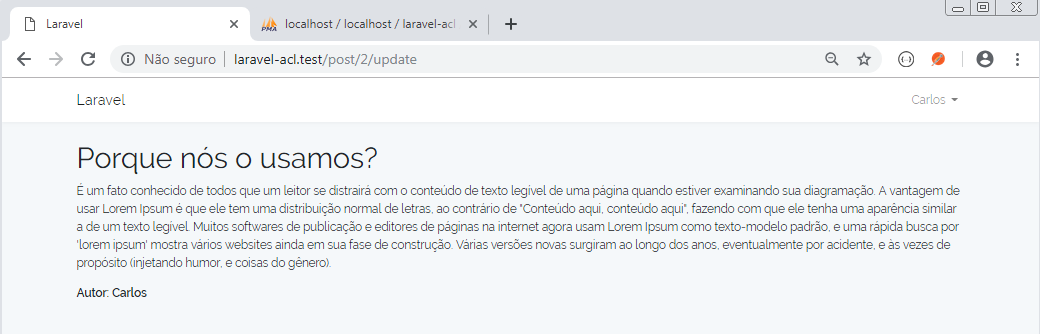
});

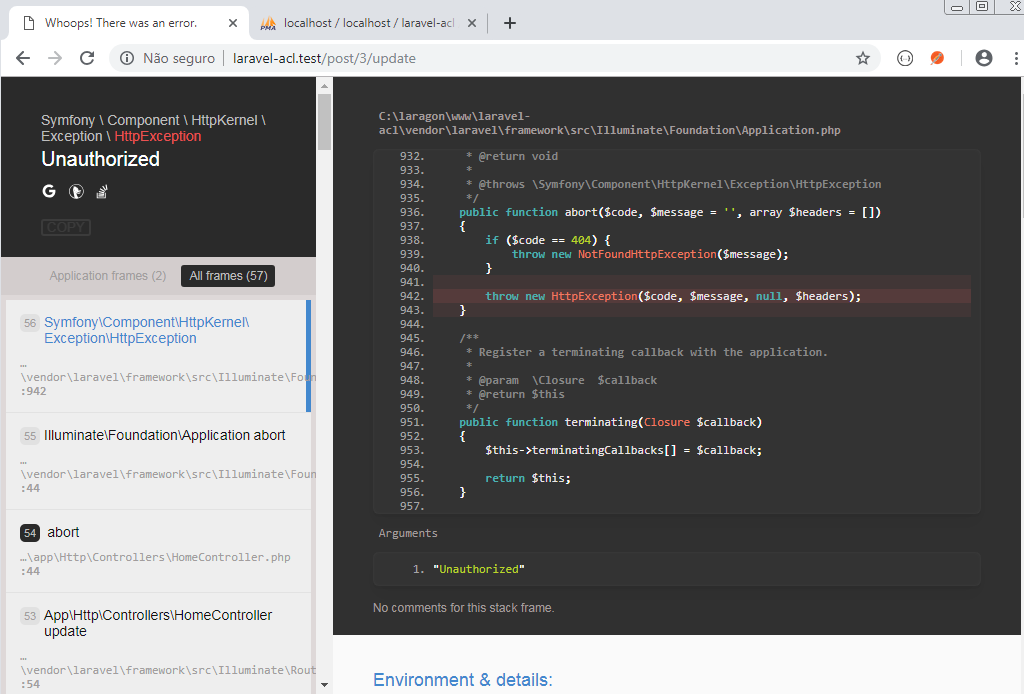
\*/

}

}



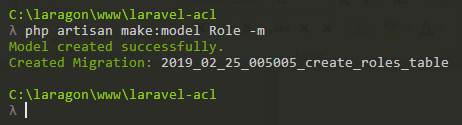




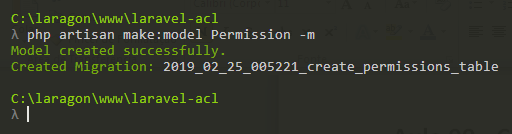
# Aula 09 - Criando migrations ACL FOR LARGE SYSTEMS

## Criando model e migration para as funções

php artisan make:model Role -m



php artisan make:model Permission -m



**database/migrations/2019\_02\_25\_005005\_create\_roles\_table.php**

<?php

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreateRolesTable extends Migration

{

public function up()

{

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

$table->increments('id');

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

$table->string('label', 200);

$table->timestamps();

});

Schema::create('role\_user', function (Blueprint $table) {

$table->increments('id');

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

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

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

->references('id')

->on('roles')

->onDelete('cascade');

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

->references('id')

->on('users')

->onDelete('cascade');

});

}

public function down()

{

Schema::drop('role\_user');

Schema::drop('roles');

}

}

**database/migrations/2019\_02\_25\_005221\_create\_permissions\_table.php**

<?php

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Database\Migrations\Migration;

class CreatePermissionsTable extends Migration

{

public function up()

{

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

$table->increments('id');

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

$table->string('label', 200);

$table->timestamps();

});

Schema::create('permission\_role', function (Blueprint $table) {

$table->increments('id');

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

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

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

->references('id')

->on('permissions')

->onDelete('cascade');

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

->references('id')

->on('roles')

->onDelete('cascade');

});

}

public function down()

{

Schema::drop('permission\_role');

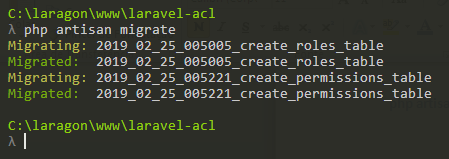
Schema::drop('permissions');

}

}

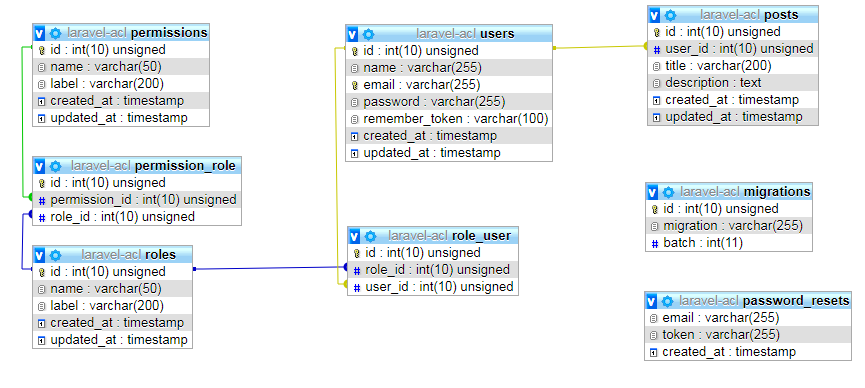
## Criando as tabelas

php artisan migrate



## Design

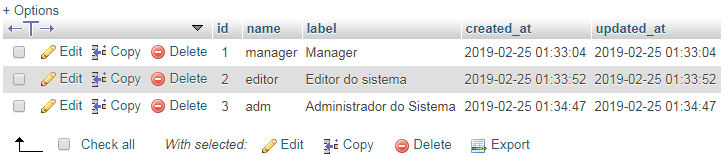
No phpmyadmin clique em Design



# Aula 10 - Definindo permissão dinamicamente

Preencha manualmente as tabelas a seguir:

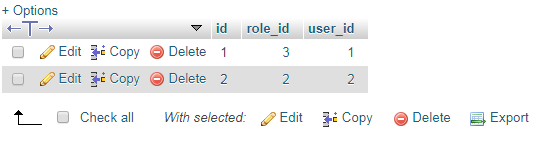
### Tabela roles



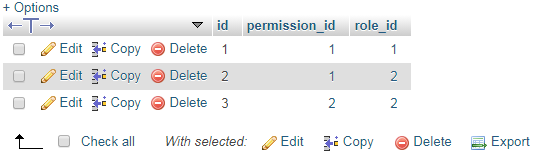
### Tabela permissions



### Tabela role\_user



### Tabela permission\_role



**app/Permission.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Permission extends Model

{

public function roles()

{

return $this->belongsToMany(\App\Role::class);

}

}

**app/Providers/AuthServiceProvider.php**

<?php

namespace App\Providers;

use Illuminate\Contracts\Auth\Access\Gate as GateContract;

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

use App\Post;

use App\User;

use App\Permission;

class AuthServiceProvider extends ServiceProvider

{

protected $policies = [

/\*

\App\Post::class => \App\Policies\PostPolicy::class,

\*/

];

public function boot(GateContract $gate)

{

$this->registerPolicies($gate);

$permissions = Permission::with('roles')->get();

foreach( $permissions as $permission )

{

$gate->define($permission->name, function(User $user) use ($permission){

return $user->hasPermission($permission);

});

}

}

}

# Aula 11 - Criando lógica de ACL

**#users**

Carlos

Paulo

**#roles**

Manager

Adm

Editor

Publish

**#users <-> roles**

Paulo -> Manager

**#permissions**

edit\_post

edit

delete\_post

edit\_fin

view\_rel\_fin

**#roles <-> permissions**

Manager -> edit\_post, edit, delete\_post

@can('edit\_post', $post)

@endcan

**app/User.php**

<?php

namespace App;

use Illuminate\Foundation\Auth\User as Authenticatable;

use App\Permission;

class User extends Authenticatable

{

protected $fillable = [

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

];

protected $hidden = [

'password', 'remember\_token',

];

public function roles()

{

// Relacionamento de muitos para muitos

return $this->belongsToMany(\App\Role::class);

}

public function hasPermission(Permission $permission)

{

return $this->hasAnyRoles($permission->roles);

}

public function hasAnyRoles($roles)

{

if(is\_array($roles) || is\_object($roles) ) {

foreach($roles as $role){

return $this->roles->contains('name', $role->name);

}

}

return $this->roles->contains('name', $roles);

}

}

# Aula 12 - Mostrar Posts por Perfil

**resources/views/home.blade.php**

@extends('layouts.app')

@section('content')

<div class="container">

@forelse($posts as $post)

@can('view\_post', $post)

<h1>{{$post->title}}</h1>

<p>{{$post->description}}</p>

<p><b>Autor: {{$post->user->name}}</b></p>

<p><a href="{{url("/post/$post->id/update")}}">Editar</a></p>

@endcan

<hr>

@empty

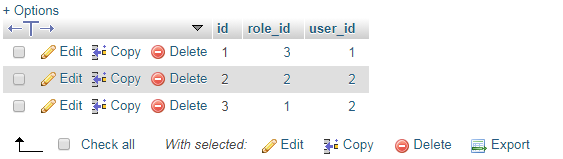
<p>Nenhum post cadastrado</p>

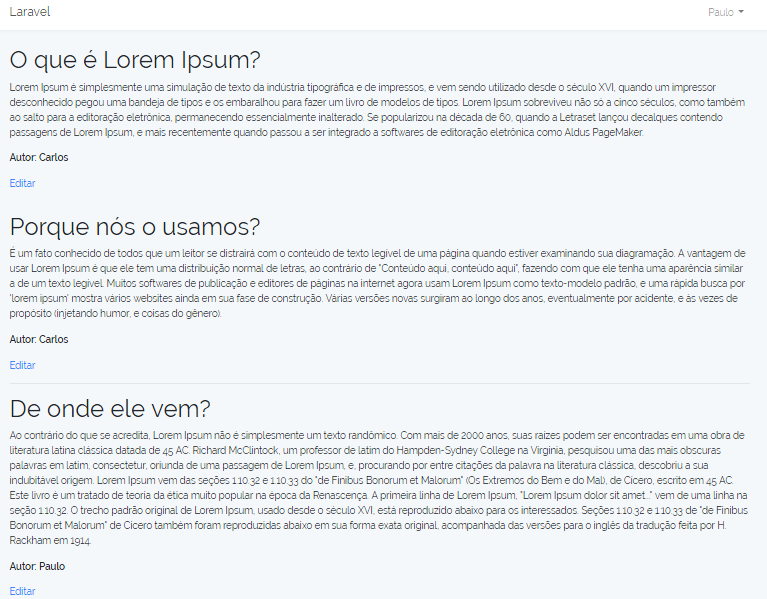
@endforelse

</div>

@endsection

Na tabela role\_user inserir o registro 3:





# Aula 13 - Debug perfis e melhorar código

**routes/web.php**

Auth::routes();

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

Route::get('/post/{id}/update', 'HomeController@update');

Route::get('/roles-permissions', 'HomeController@rolesPermissions');

**app/http/Controllers/HomeController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Post;

use Gate;

class HomeController extends Controller

{

public function \_\_construct()

{

$this->middleware('auth');

}

public function index(Post $post)

{

// Seleciona todos os posts

$posts = $post->all();

// Seleciona todos os posts do usuário logado

// $posts = $post->where('user\_id', auth()->user()->id)->get();

return view('home', compact('posts'));

}

public function update($idpost){

$post = Post::find($idpost);

// $this->authorize('update-post', $post);

if( Gate::denies('update-post', $post) )

abort(403, 'Unauthorized');

return view('post-update', compact('post'));

}

public function rolesPermissions(){

// return 'Roles e Permissions to users';

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

echo "<h1>{$nameUser}</h1>";

foreach(auth()->user()->roles as $role){

echo "<b>$role->name</b> -> ";

$permissions = $role->permissions;

foreach($permissions as $permission){

echo " $permission->name , ";

}

echo "<hr>";

}

}

}

**app/Role.php**

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Role extends Model

{

public function permissions()

{

return $this->belongsToMany(\App\Permission::class);

}

}

**app/User.php**

<?php

namespace App;

use Illuminate\Foundation\Auth\User as Authenticatable;

use App\Permission;

class User extends Authenticatable

{

protected $fillable = [

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

];

protected $hidden = [

'password', 'remember\_token',

];

public function roles()

{

// Relacionamento de muitos para muitos

return $this->belongsToMany(\App\Role::class);

}

public function hasPermission(Permission $permission)

{

return $this->hasAnyRoles($permission->roles);

}

public function hasAnyRoles($roles)

{

if(is\_array($roles) || is\_object($roles) ) {

// foreach($roles as $role){

// return $this->roles->contains('name', $role->name);

// }

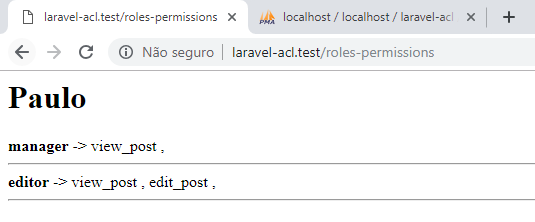
return !! $roles->intersect($this->roles)->count();

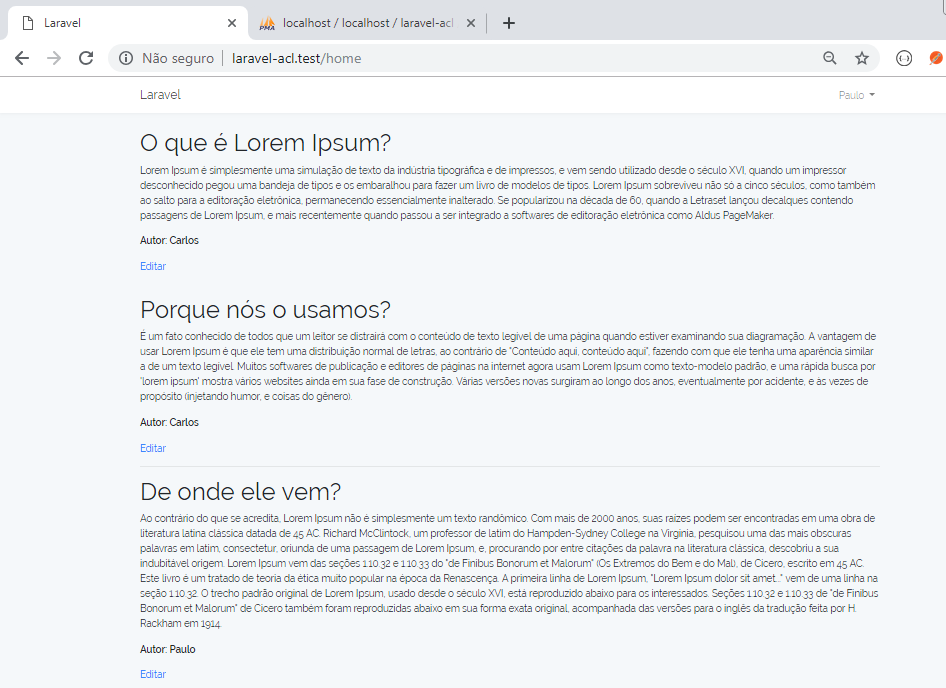
}

return $this->roles->contains('name', $roles);

}

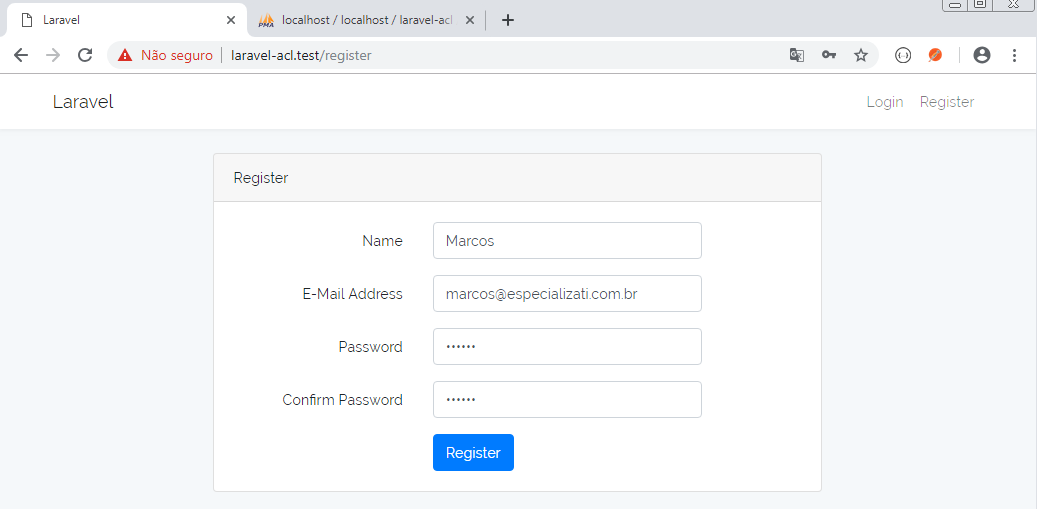
}



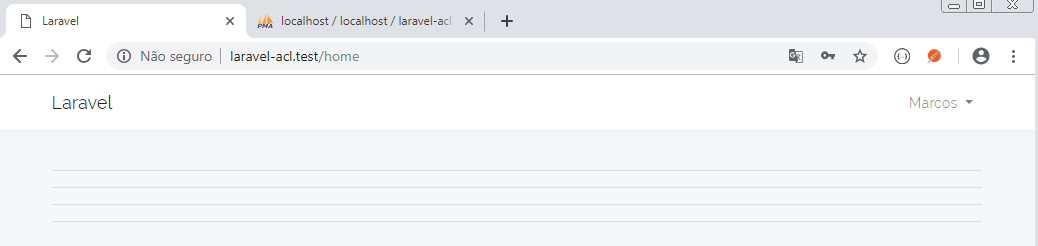


# Aula 14 - Definindo Super Admin Sistema

Insira um novo usuário:

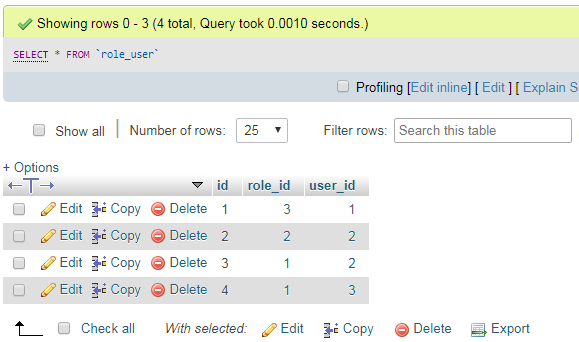


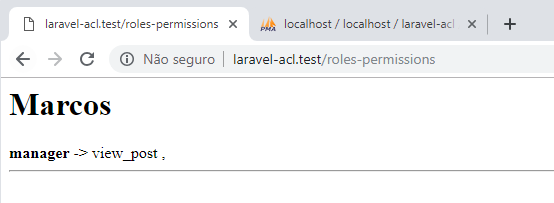
Senha: 123456

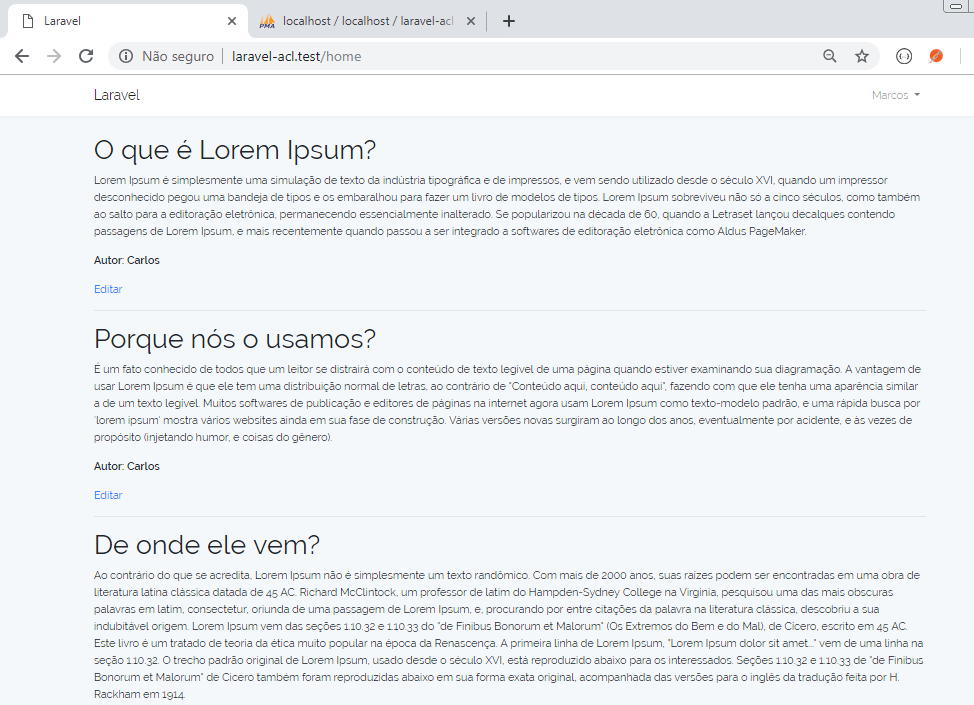


**tabela role\_user**

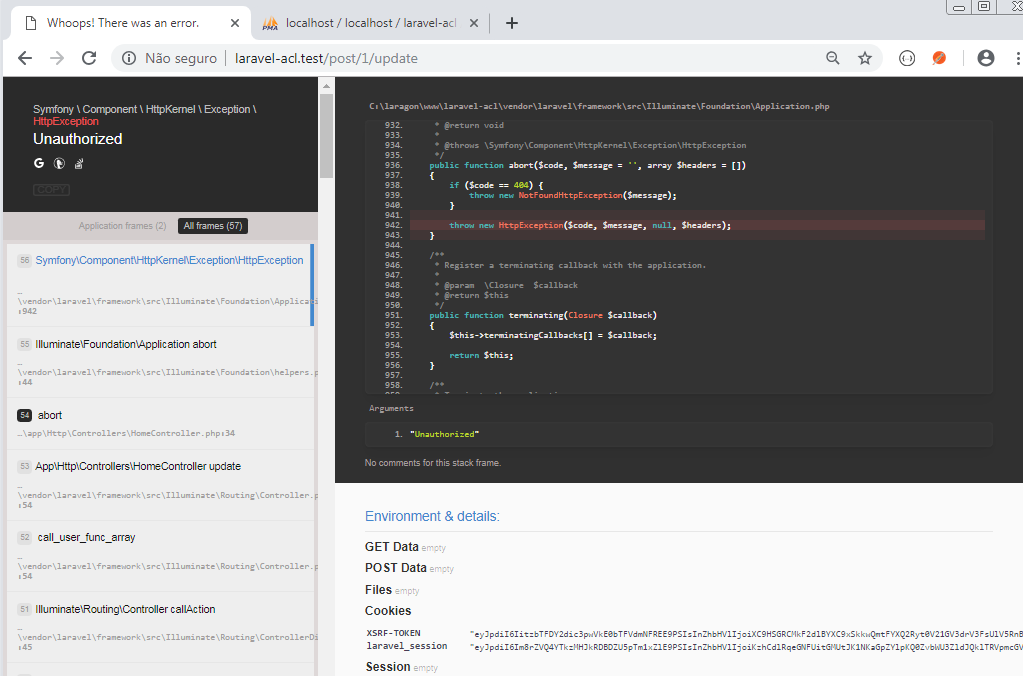
Insira o registro 4:







Ao tentar editar o post 1



**app/Providers/AuthServiceProvider.php**

<?php

namespace App\Providers;

use Illuminate\Contracts\Auth\Access\Gate as GateContract;

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

use App\Post;

use App\User;

use App\Permission;

class AuthServiceProvider extends ServiceProvider

{

protected $policies = [

/\*

\App\Post::class => \App\Policies\PostPolicy::class,

\*/

];

public function boot(GateContract $gate)

{

$this->registerPolicies($gate);

/\*

$gate->define('update-post', function(User $user, Post $post){

return $user->id == $post->user\_id;

});

\*/

$permissions = Permission::with('roles')->get();

foreach( $permissions as $permission )

{

$gate->define($permission->name, function(User $user) use ($permission){

return $user->hasPermission($permission);

});

}

$gate->before(function(User $user, $ability){

if ( $user->hasAnyRoles('adm') )

return true;

});

}

}

**app/Policies/PostPolicy.php**

<?php

namespace App\Policies;

use Illuminate\Auth\Access\HandlesAuthorization;

use App\User;

use App\Post;

class PostPolicy

{

use HandlesAuthorization;

public function \_\_construct()

{

//

}

public function updatePost(User $user, Post $post)

{

return $user->id == $post->user\_id;

}

public function before(User $user)

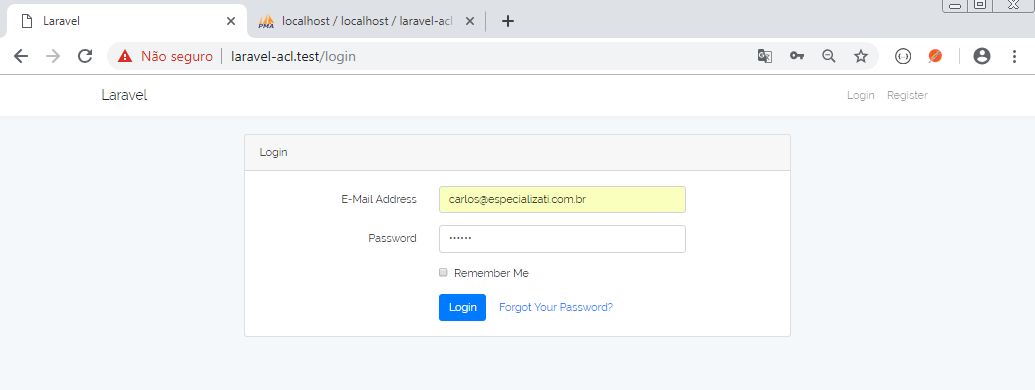
{

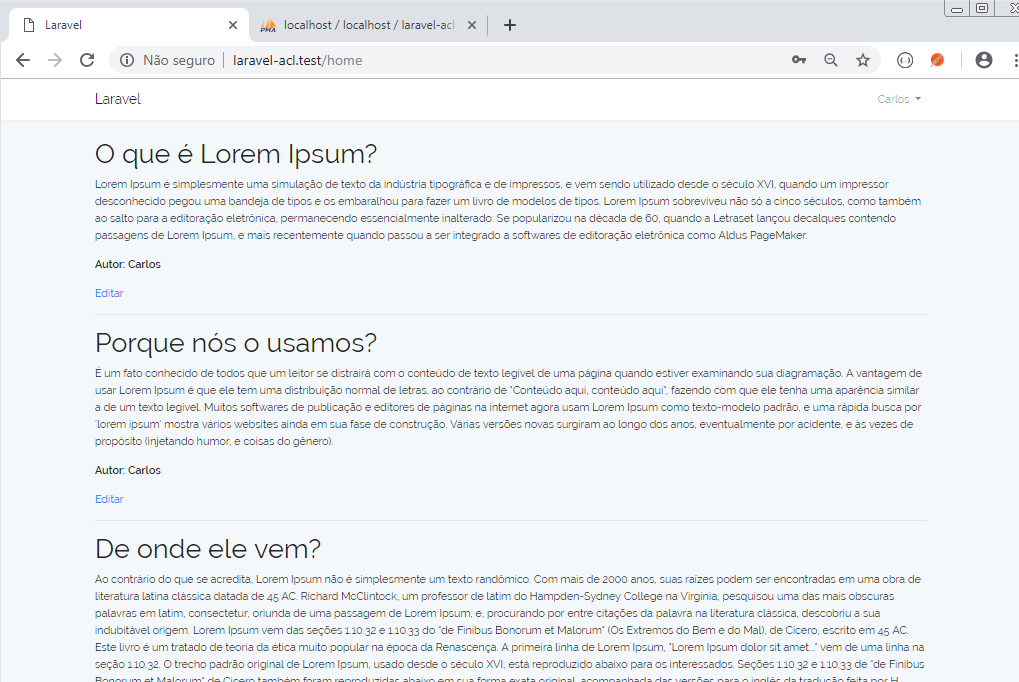
return $user->name == 'Carlos';

}

}

## Logando como um administrador do sistema





Clique no link Editar do terceiro post

