**Curso de Laravel 8**

**cursodesarrolloweb**

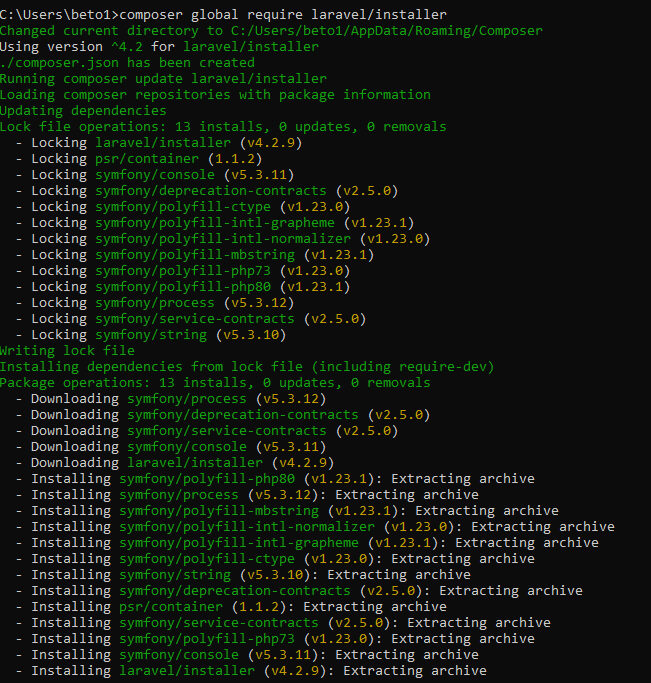
<https://www.youtube.com/watch?v=75FYScxTBiY&list=PLd3a4dr8oUsDAjQa8T0eKSyOxUCoiMVxO>

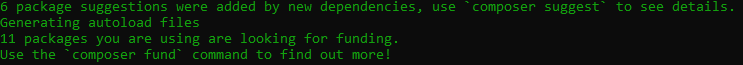
Resumo feito por Roberto Pinheiro

# Aula 03 - Instalação

## Instalação global

composer global require laravel/installer





laravel -V



laravel new blog

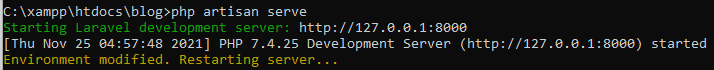
cd blog

## Instalação usando composer

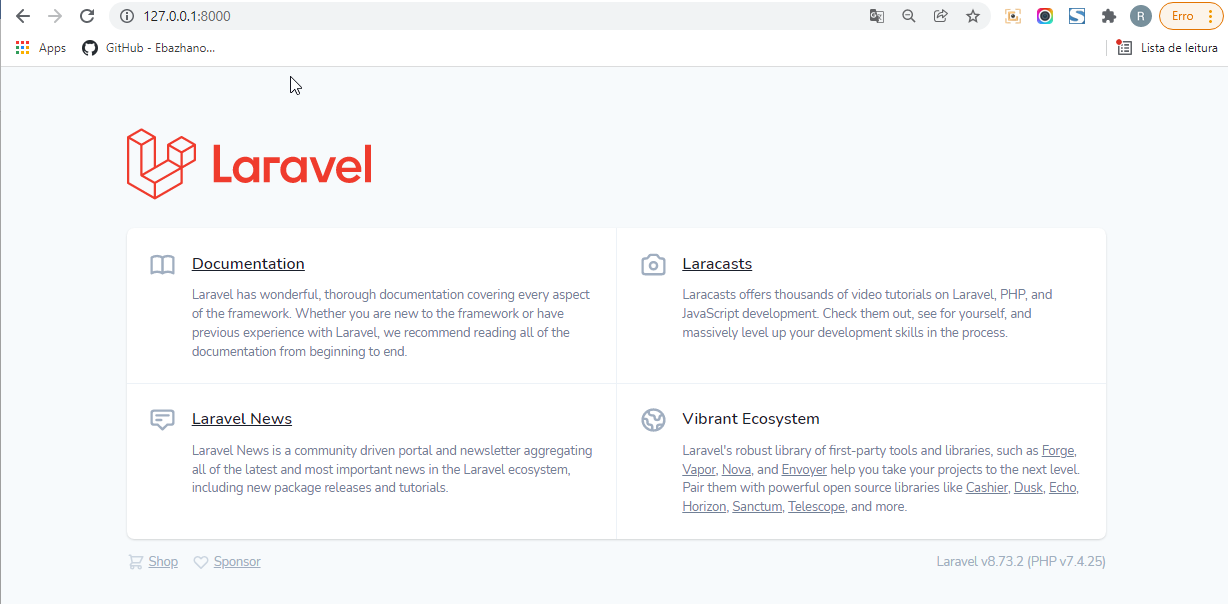
composer create-project laravel/laravel blog2

# Aula 04 - Rodando o servidor

php artisan serve

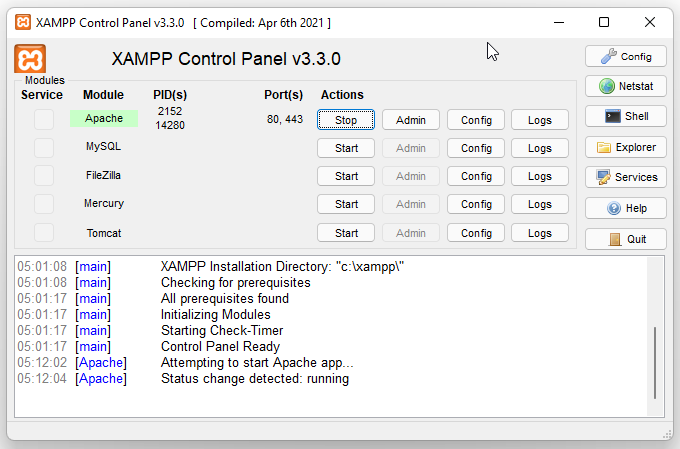


http://127.0.0.1:8000

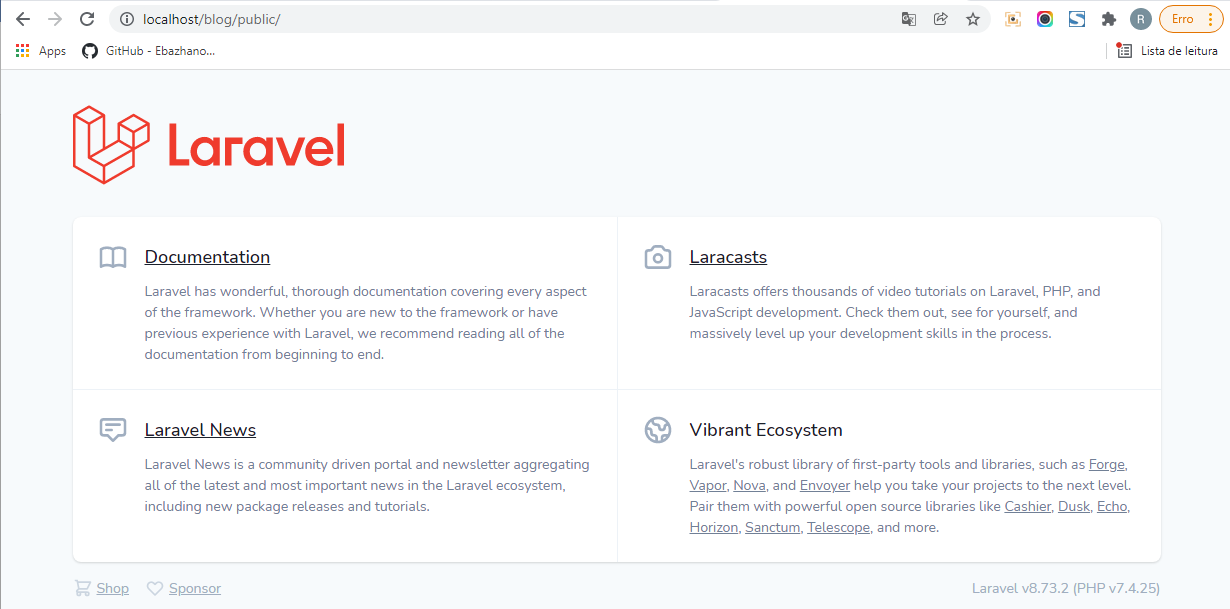


## Usando Xampp

- Inicie o servidor Apache do Xampp



http://localhost/blog/public/



# Aula 08 - Introdução ao sistema de rotas

**routes\web.php**

<?php

use Illuminate\Support\Facades\Route;

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

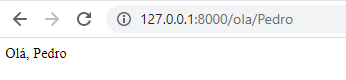
return view('welcome');

});

Route::get('/ola/{name}', function ($name) {

return ("Olá, {$name}");

});



# Aula 09 - Introdução aos controladores

## Criando um controller de forma manual

**app\Http\Controllers\OlaController.php**

<?php

namespace App\Http\Controllers;

class OlaController extends Controller {

public function \_\_invoke(string $name){

return ("Olá, {$name}");

}

}

**routes\web.php**

<?php

use Illuminate\Support\Facades\Route;

use \App\Http\Controllers\OlaController;

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

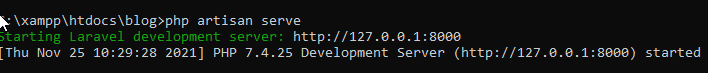
return view('welcome');

});

Route::get('/ola/{name}', OlaController::class);

# Aula 10 - Artisan, a linha de comandos do Laravel

php artisan serve

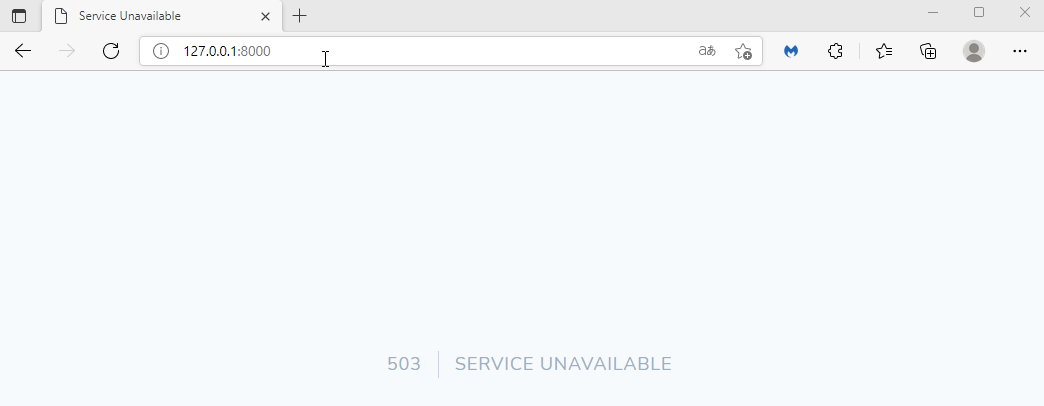


php artisan inspire



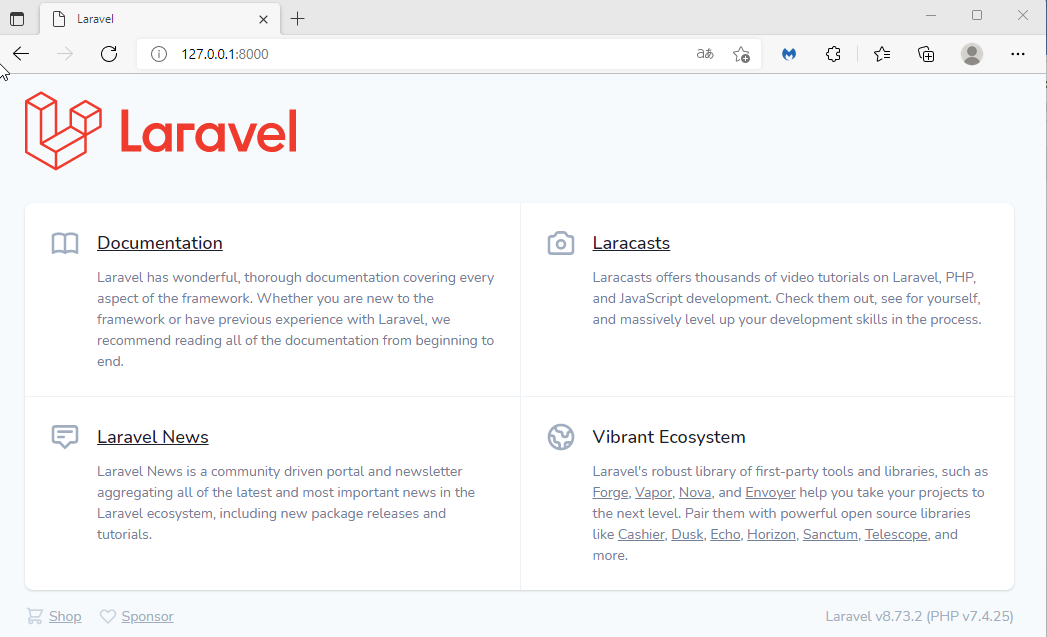
php artisan down





php artisan up

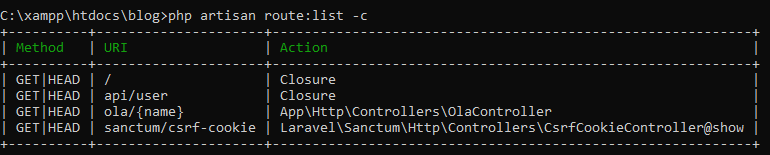




php artisan route:list



php artisan route:list -c



## Criando controller com o Artisan

php artisan make:controller OlaController



**app\Http\Controllers\OlaController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class OlaController extends Controller

{

public function \_\_invoke(string $name)

{

return ("Olá, {$name}");

}

}

**routes\web.php**

<?php

use Illuminate\Support\Facades\Route;

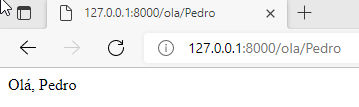
use \App\Http\Controllers\OlaController;

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

return view('welcome');

});

Route::get('/ola/{name}', OlaController::class);



# Aula 11 - Uso de métodos em Controladores

**routes\web.php**

<?php

use Illuminate\Support\Facades\Route;

use \App\Http\Controllers\OlaController;

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

return view('welcome');

});

Route::get('/ola/{name}', [OlaController::class, "index"]);

**app\Http\Controllers\OlaController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class OlaController extends Controller

{

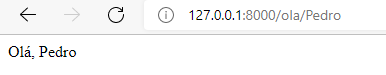
public function index(string $name)

{

return ("Olá, {$name}");

}

}



**app\Http\Controllers\OlaController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class OlaController extends Controller

{

public function index(string $name)

{

return view("saudar", compact("name"));

}

}

**resources\views\saudar.blade.php**

<h1>Olá, {{ $name }}</h1>



# Aula 12 - Variáveis de ambiente

**.env**

APP\_NAME="Curso de Laravel 8 - Cursosdesarrolloweb"

APP\_ENV=local

APP\_KEY=base64:ejpjIGU95fkNemcwzBHpU9YMaDLBmGSWo5ebkN22F/Q=

APP\_DEBUG=true

APP\_URL=http://blog.test

LOG\_CHANNEL=stack

LOG\_DEPRECATIONS\_CHANNEL=null

LOG\_LEVEL=debug

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=blog

DB\_USERNAME=root

DB\_PASSWORD=

BROADCAST\_DRIVER=log

CACHE\_DRIVER=file

FILESYSTEM\_DRIVER=local

QUEUE\_CONNECTION=sync

SESSION\_DRIVER=file

SESSION\_LIFETIME=120

MEMCACHED\_HOST=127.0.0.1

REDIS\_HOST=127.0.0.1

REDIS\_PASSWORD=null

REDIS\_PORT=6379

MAIL\_MAILER=smtp

MAIL\_HOST=mailhog

MAIL\_PORT=1025

MAIL\_USERNAME=null

MAIL\_PASSWORD=null

MAIL\_ENCRYPTION=null

MAIL\_FROM\_ADDRESS=null

MAIL\_FROM\_NAME="${APP\_NAME}"

AWS\_ACCESS\_KEY\_ID=

AWS\_SECRET\_ACCESS\_KEY=

AWS\_DEFAULT\_REGION=us-east-1

AWS\_BUCKET=

AWS\_USE\_PATH\_STYLE\_ENDPOINT=false

PUSHER\_APP\_ID=

PUSHER\_APP\_KEY=

PUSHER\_APP\_SECRET=

PUSHER\_APP\_CLUSTER=mt1

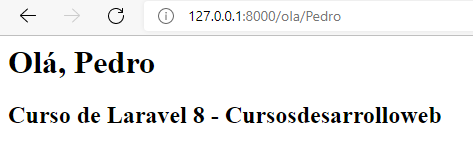
MIX\_PUSHER\_APP\_KEY="${PUSHER\_APP\_KEY}"

MIX\_PUSHER\_APP\_CLUSTER="${PUSHER\_APP\_CLUSTER}"

**resources\views\saudar.blade.php**

<h1>Olá, {{ $name }}</h1>

<h2>{{env("APP\_NAME")}}</h2>

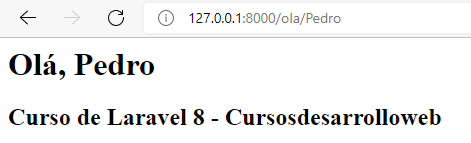


# Aula 13 - Uso correto de variáveis de ambiente

**resources\views\saudar.blade.php**

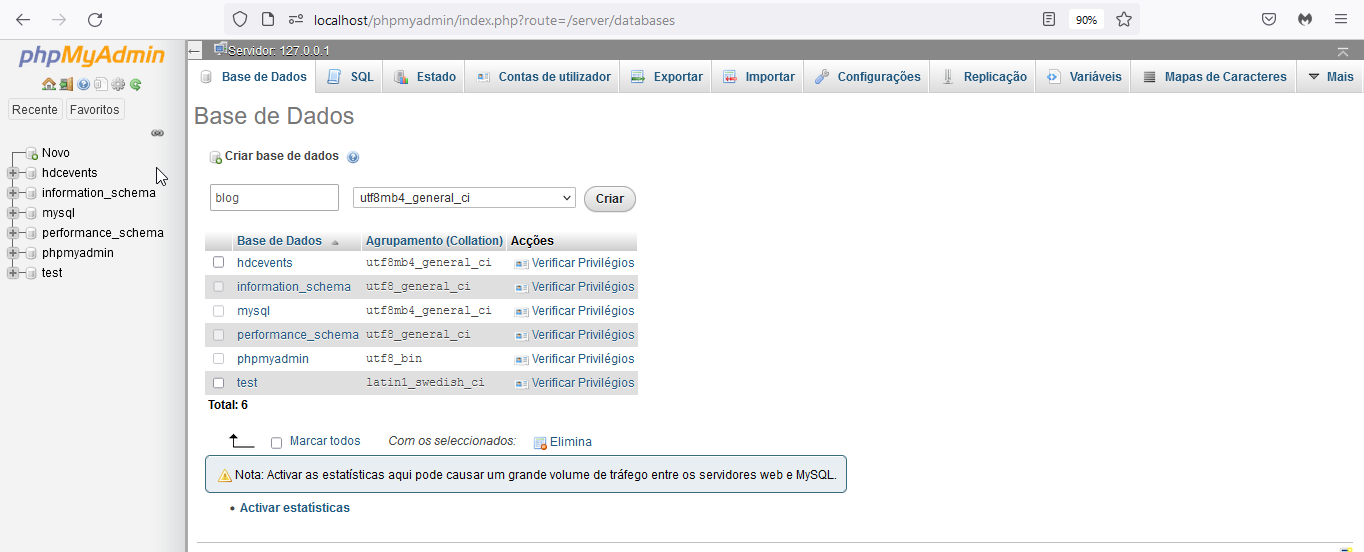
<h1>Olá, {{ $name }}</h1>

<h2>{{ config("app.name") }}</h2>



# Aula 16 - Introdução às migrations

- No PHPMyAdmin crie a base de dados chamada blog.



**database\migrations\2014\_10\_12\_000000\_create\_users\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreateUsersTable extends Migration

{

/\*\*

\* Run the migrations.

\*

\* @return void

\*/

public function up()

{

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

$table->id();

$table->string('name');

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

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

$table->string('password');

$table->rememberToken();

$table->timestamps();

});

}

/\*\*

\* Reverse the migrations.

\*

\* @return void

\*/

public function down()

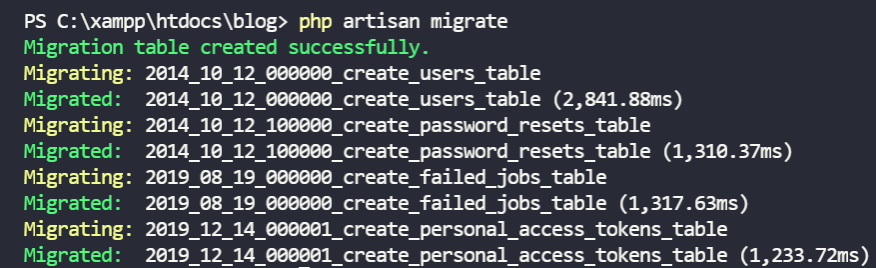
{

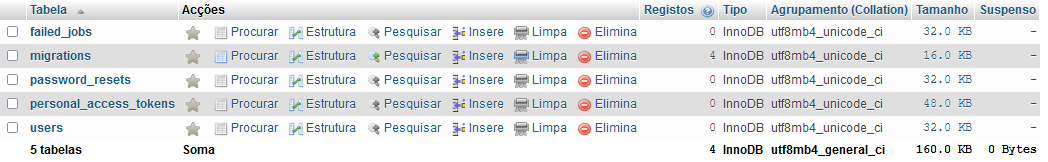
Schema::dropIfExists('users');

}

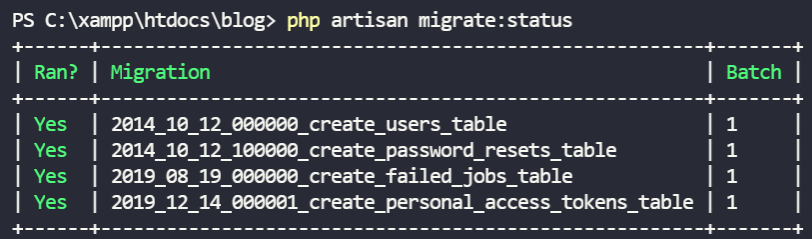
}

php artisan migrate



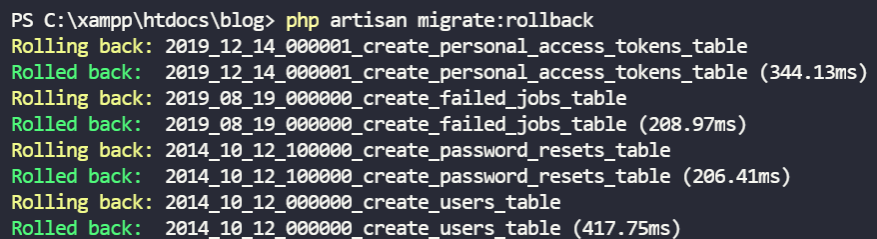


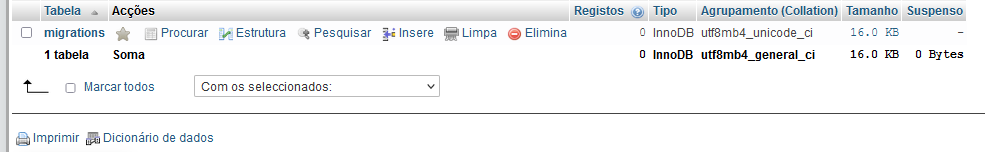
php artisan migrate:status



## Voltando atrás na última migration

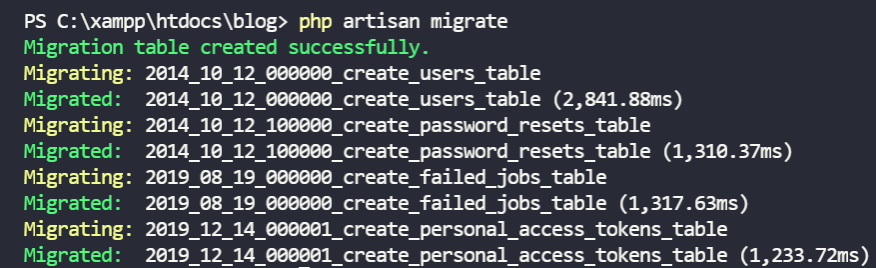
php artisan migrate:rollback





## Recriando as tabelas no banco de dados

php artisan migrate



## Squashing migrations

Em variáveis de ambiente, adicionar no path:

C:\xampp\mysql\bin

php artisan schema:dump



**database\schema\mysql-schema.dump**

/\*!40103 SET @OLD\_TIME\_ZONE=@@TIME\_ZONE \*/;

/\*!40103 SET TIME\_ZONE='+00:00' \*/;

/\*!40014 SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0 \*/;

/\*!40014 SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0 \*/;

/\*!40101 SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='NO\_AUTO\_VALUE\_ON\_ZERO' \*/;

/\*!40111 SET @OLD\_SQL\_NOTES=@@SQL\_NOTES, SQL\_NOTES=0 \*/;

DROP TABLE IF EXISTS `failed\_jobs`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `failed\_jobs` (

`id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT,

`uuid` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`connection` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

`queue` text COLLATE utf8mb4\_unicode\_ci NOT NULL,

`payload` longtext COLLATE utf8mb4\_unicode\_ci NOT NULL,

`exception` longtext COLLATE utf8mb4\_unicode\_ci NOT NULL,

`failed\_at` timestamp NOT NULL DEFAULT current\_timestamp(),

PRIMARY KEY (`id`),

UNIQUE KEY `failed\_jobs\_uuid\_unique` (`uuid`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

DROP TABLE IF EXISTS `migrations`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `migrations` (

`id` int(10) unsigned NOT NULL AUTO\_INCREMENT,

`migration` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`batch` int(11) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

DROP TABLE IF EXISTS `password\_resets`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `password\_resets` (

`email` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`token` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

KEY `password\_resets\_email\_index` (`email`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

DROP TABLE IF EXISTS `personal\_access\_tokens`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `personal\_access\_tokens` (

`id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT,

`tokenable\_type` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`tokenable\_id` bigint(20) unsigned NOT NULL,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`token` varchar(64) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`abilities` text COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`last\_used\_at` timestamp NULL DEFAULT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `personal\_access\_tokens\_token\_unique` (`token`),

KEY `personal\_access\_tokens\_tokenable\_type\_tokenable\_id\_index` (`tokenable\_type`,`tokenable\_id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

DROP TABLE IF EXISTS `users`;

/\*!40101 SET @saved\_cs\_client = @@character\_set\_client \*/;

/\*!40101 SET character\_set\_client = utf8 \*/;

CREATE TABLE `users` (

`id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT,

`name` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`email` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`email\_verified\_at` timestamp NULL DEFAULT NULL,

`password` varchar(255) COLLATE utf8mb4\_unicode\_ci NOT NULL,

`remember\_token` varchar(100) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`created\_at` timestamp NULL DEFAULT NULL,

`updated\_at` timestamp NULL DEFAULT NULL,

PRIMARY KEY (`id`),

UNIQUE KEY `users\_email\_unique` (`email`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

/\*!40101 SET character\_set\_client = @saved\_cs\_client \*/;

/\*!40103 SET TIME\_ZONE=@OLD\_TIME\_ZONE \*/;

/\*!40101 SET SQL\_MODE=@OLD\_SQL\_MODE \*/;

/\*!40014 SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS \*/;

/\*!40014 SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS \*/;

/\*!40111 SET SQL\_NOTES=@OLD\_SQL\_NOTES \*/;

INSERT INTO `migrations` VALUES (5,'2014\_10\_12\_000000\_create\_users\_table',1);

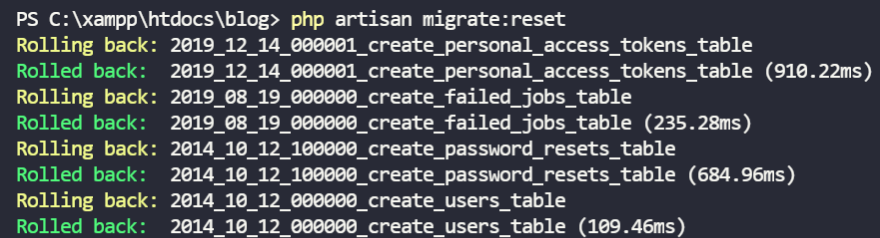
INSERT INTO `migrations` VALUES (6,'2014\_10\_12\_100000\_create\_password\_resets\_table',1);

INSERT INTO `migrations` VALUES (7,'2019\_08\_19\_000000\_create\_failed\_jobs\_table',1);

INSERT INTO `migrations` VALUES (8,'2019\_12\_14\_000001\_create\_personal\_access\_tokens\_table',1);

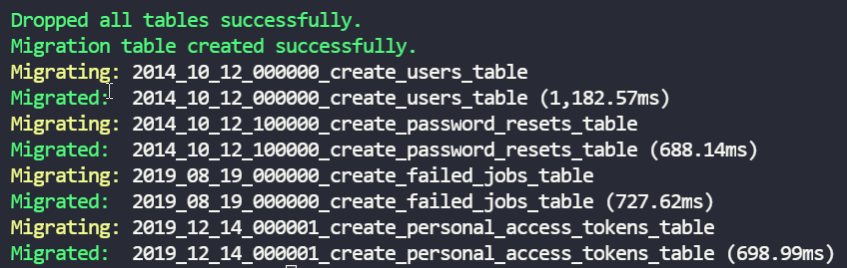
## Voltando atrás todas as migrations

php artisan migrate:reset



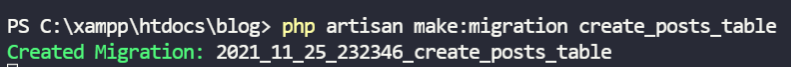
## Resetando e voltando a criar as migrations

php artisan migrate:fresh



# Aula 17 - Criar uma migration simples para a tabela de Posts

php artisan make:migration create\_posts\_table



**database\migrations\2021\_11\_25\_232346\_create\_posts\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreatePostsTable extends Migration

{

public function up()

{

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

$table->id();

$table->string("title", 100)->unique();

$table->text("content");

$table->timestamps();

});

}

public function down()

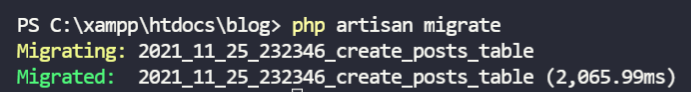
{

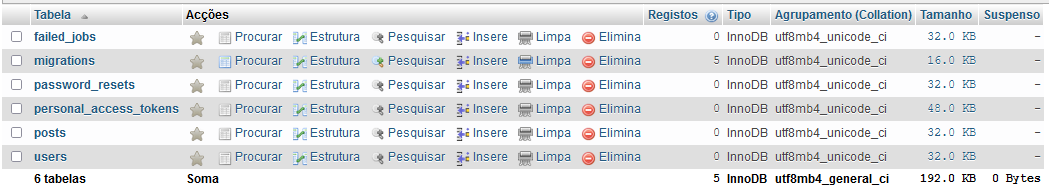
Schema::dropIfExists('posts');

}

}

php artisan migrate





# Aula 18 - Adicionar foreign key user\_id a tabela de posts

php artisan make:migration add\_user\_id\_to\_posts\_table



**database\migrations\2021\_11\_25\_234851\_add\_user\_id\_to\_posts\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class AddUserIdToPostsTable extends Migration

{

public function up()

{

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

$table->unsignedBigInteger('user\_id')->after("id");

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

});

}

public function down()

{

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

$table->dropForeign(['user\_id']);

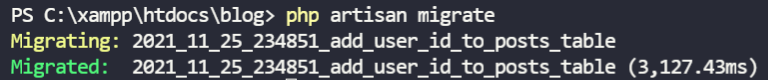
$table->dropColumn("user\_id");

});

}

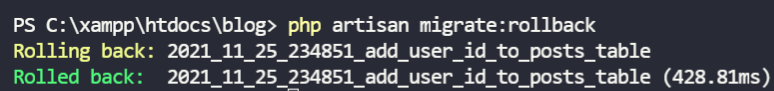
}

php artisan migrate





php artisan migrate:rollback





## Outra forma de criar a migration

**database\migrations\2021\_11\_25\_234851\_add\_user\_id\_to\_posts\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class AddUserIdToPostsTable extends Migration

{

public function up()

{

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

$table->foreignId('user\_id')->constrained()->after("id");

});

}

public function down()

{

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

$table->dropForeign(['user\_id']);

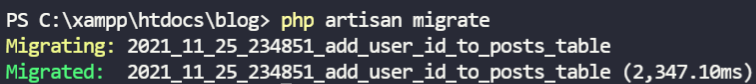
$table->dropColumn("user\_id");

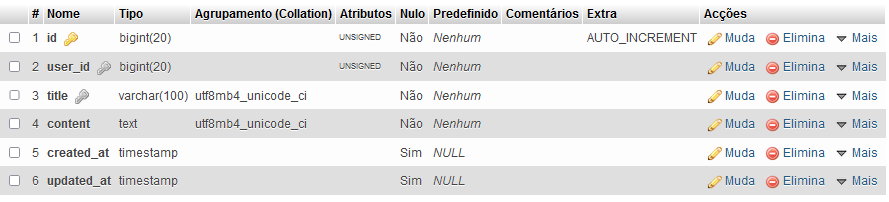
});

}

}

php artisan migrate





# Aula 19 - Modificar colunas utilizando migrations e DoctrineDBAL

## Instalando doctrine/dbal

composer require doctrine/dbal



php artisan make:migration update\_title\_to\_posts\_table



**database\migrations\2021\_11\_26\_003127\_update\_title\_to\_posts\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class UpdateTitleToPostsTable extends Migration

{

public function up()

{

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

$table->string('title', 180)->change();

});

}

public function down()

{

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

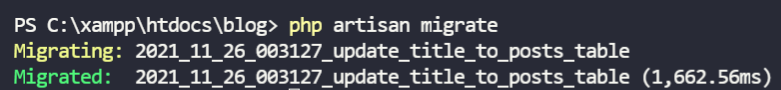
$table->string('title', 100)->change();

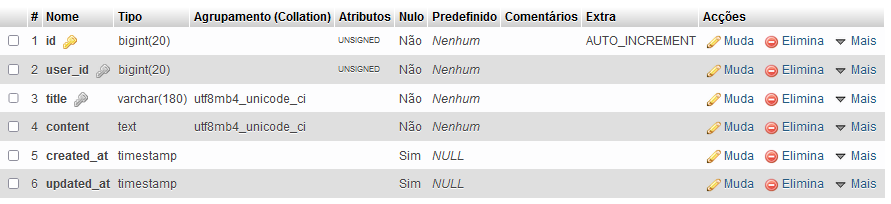
});

}

}

php artisan migrate





# Aula 20 - Introdução aos modelos em Laravel e criar modelo Post

## Criando o modelo Post

php artisan make:model Post



**app\Models\Post.php**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Post extends Model

{

use HasFactory;

}

# Aula 21 - Trabalhando com Factories e Seeds

**database\factories\UserFactory.php**

<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

use Illuminate\Support\Str;

class UserFactory extends Factory

{

public function definition()

{

return [

'name' => $this->faker->name(),

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

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

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

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

];

}

public function unverified()

{

return $this->state(function (array $attributes) {

return [

'email\_verified\_at' => null,

];

});

}

}

**database\seeders\DatabaseSeeder.php**

<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

use \App\Models\User;

class DatabaseSeeder extends Seeder

{

public function run()

{

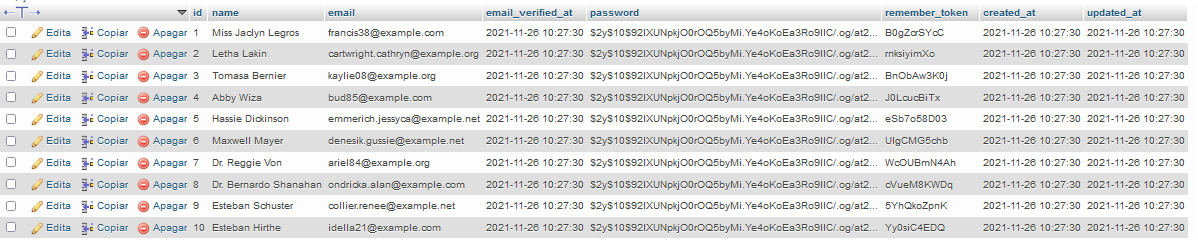
User::factory(10)->create();

}

}

php artisan db:seed





## Criando um factory para posts

php artisan make:factory PostFactory



**database\factories\PostFactory.php**

<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

use \App\Models\User;

class PostFactory extends Factory

{

public function definition()

{

return [

"user\_id" => User::factory(),

"title" => $this->faker->text(50),

"content" => $this->faker->text(500),

"created\_at" => now()

];

}

}

php artisan migrate:fresh

**database\seeders\DatabaseSeeder.php**

<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

// use \App\Models\User;

use \App\Models\Post;

class DatabaseSeeder extends Seeder

{

public function run()

{

// User::factory(10)->create();

Post::factory(20)->create();

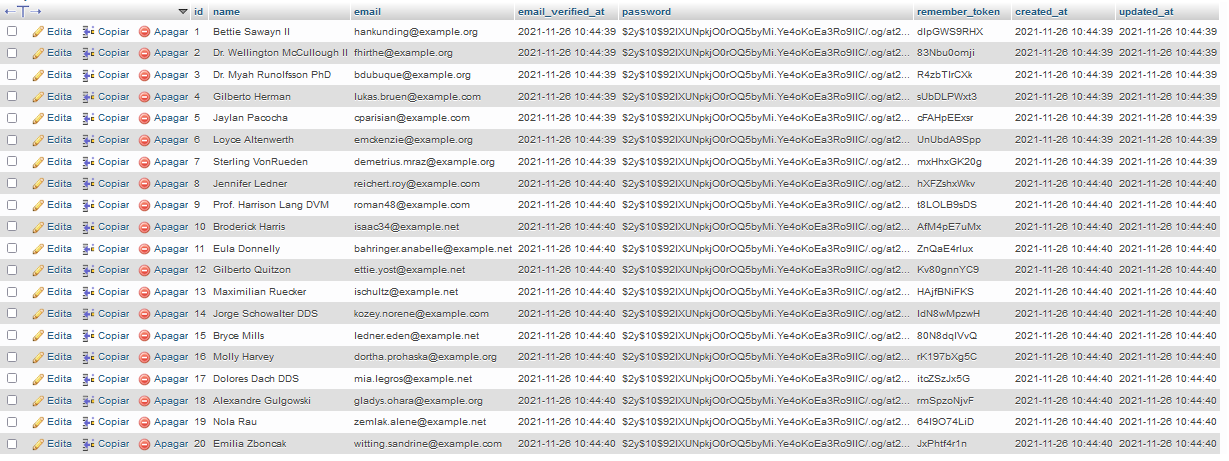
}

}

php artisan db:seed







php artisan make:seeder PostSeeder



**database\seeders\PostSeeder.php**

<?php

namespace Database\Seeders;

use \App\Models\Post;

use Illuminate\Database\Seeder;

class PostSeeder extends Seeder

{

public function run()

{

Post::factory(20)->create();

}

}

**database\seeders\DatabaseSeeder.php**

<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;

use \App\Models\Post;

class DatabaseSeeder extends Seeder

{

public function run()

{

$this->call([

PostSeeder::class,

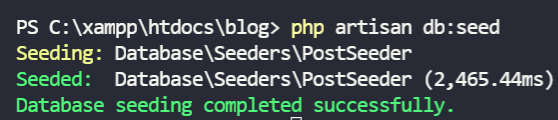
]);

}

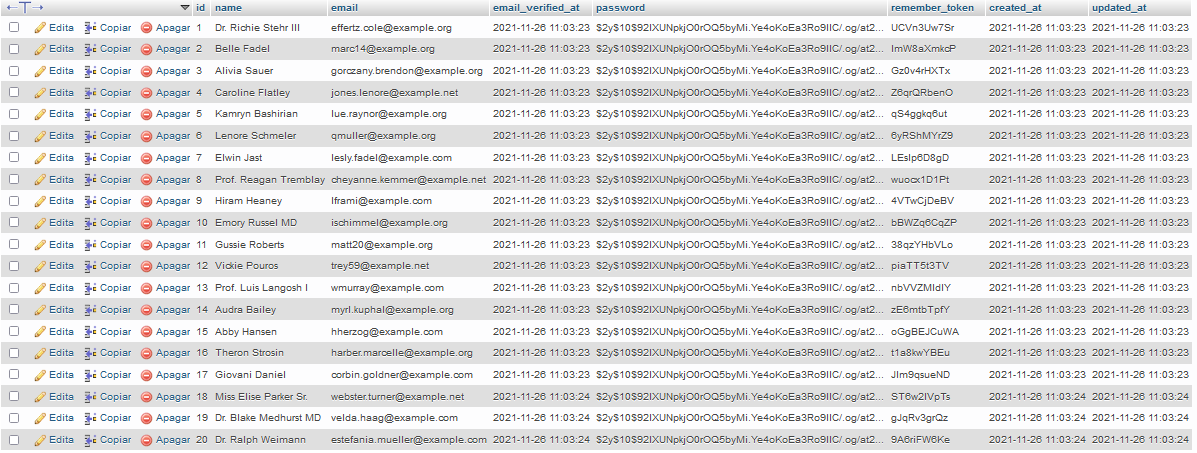
}

php artisan migrate:fresh

php artisan db:seed

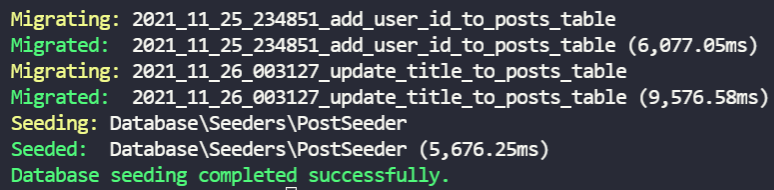






# Aula 22 - Eloquent, relações e novo modelo Phone

php artisan migrate:fresh --seed



**app\Models\User.php**

<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable

{

use HasApiTokens, HasFactory, Notifiable;

protected $fillable = [

'name',

'email',

'password',

];

protected $hidden = [

'password',

'remember\_token',

];

protected $casts = [

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

];

public function posts(): HasMany {

return $this->hasMany(Post::class);

}

}

**app\Models\Post.php**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Post extends Model

{

use HasFactory;

protected $fillable = [

"user\_id",

"title",

"content",

];

public function user(): BelongsTo {

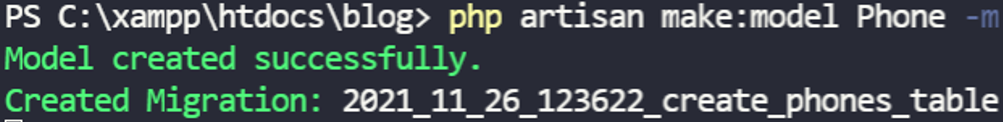
return $this->belongsTo(User::class);

}

}

## Criando o model Phone e sua migration

php artisan make:model Phone -m



**database\migrations\2021\_11\_26\_123622\_create\_phones\_table.php**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

class CreatePhonesTable extends Migration

{

public function up()

{

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

$table->id();

$table->foreignId("user\_id")->constrained()->onDelete("cascade");

$table->string("phone\_number", 15)->unique();

$table->timestamps();

});

}

public function down()

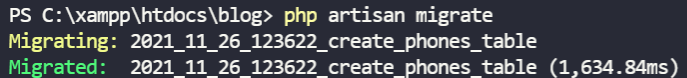
{

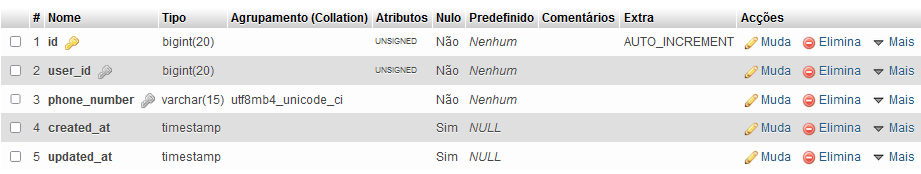
Schema::dropIfExists('phones');

}

}

php artisan migrate





**app\Models\Phone.php**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Phone extends Model

{

use HasFactory;

protected $fillable = [

"user\_id", "phone\_number"

];

public function user(): BelongsTo {

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

}

}

**app\Models\User.php**

<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Relations\HasMany;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable

{

use HasApiTokens, HasFactory, Notifiable;

protected $fillable = [

'name',

'email',

'password',

];

protected $hidden = [

'password',

'remember\_token',

];

protected $casts = [

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

];

public function phone(): HasOne {

return $this->hasOne(Phone::class);

}

public function posts(): HasMany{

return $this->hasMany(Post::class);

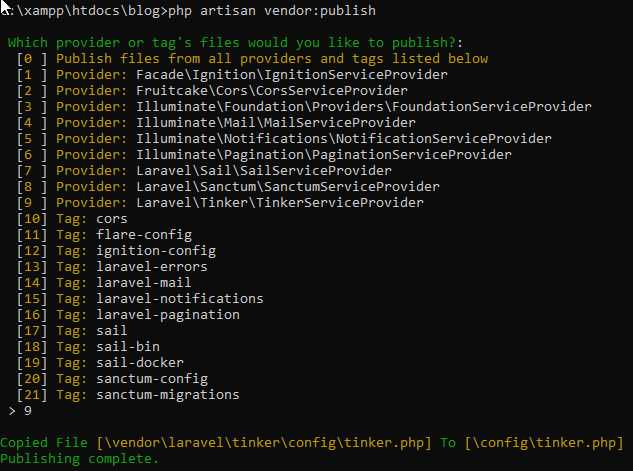
}

}

# Aula 23 - Tinker, REPL para Laravel

php artisan vendor:publish

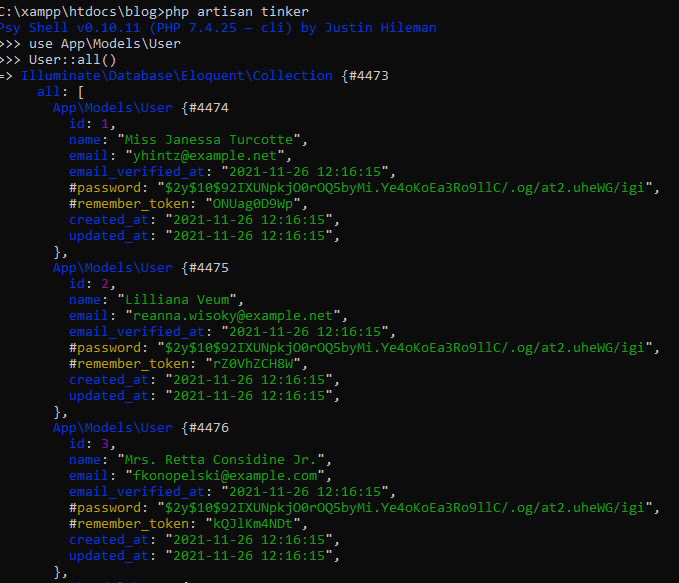
- Selecione a opção 9: Tinker



php artisan tinker

use App\Models\User

User::all()

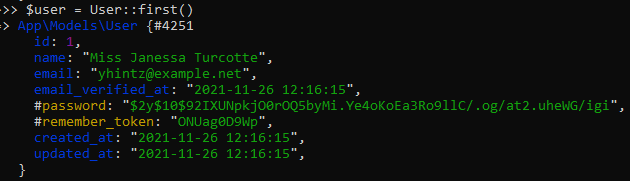


.

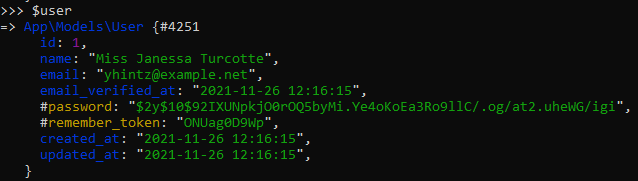
.

.

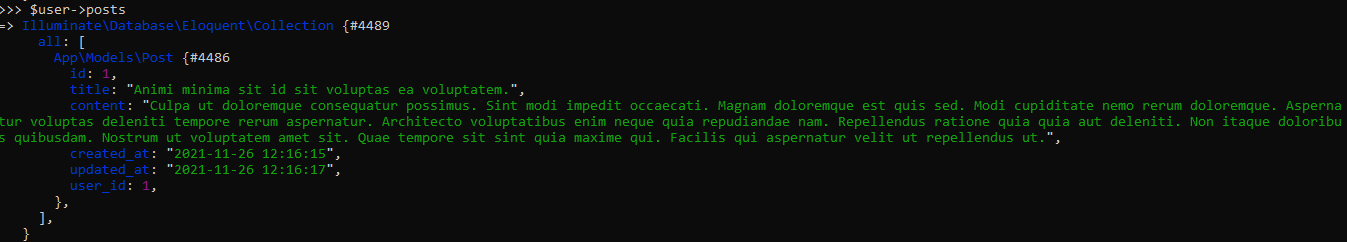
$user = User::first()



$user



$user->posts



use App\Models\Phone

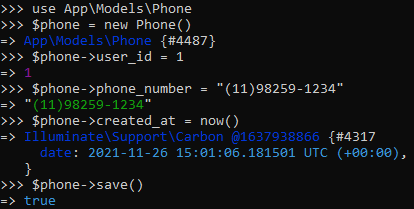
$phone = new Phone()

$phone->user\_id = 1

$phone->phone\_number = "(11)98259-1234"

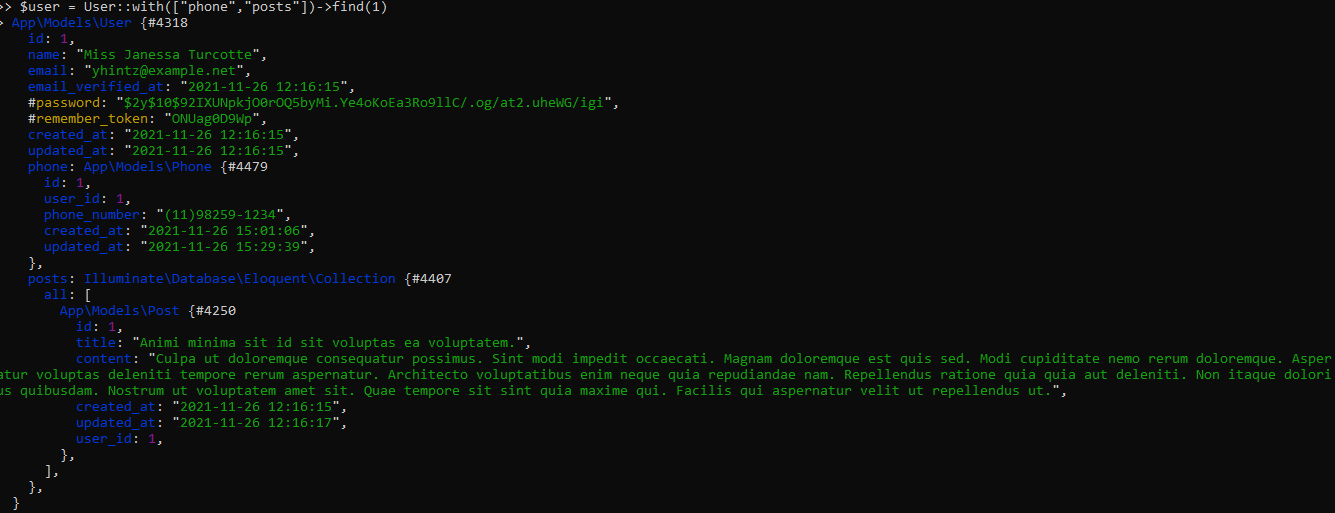
$phone->created\_at = now()

$phone->save()

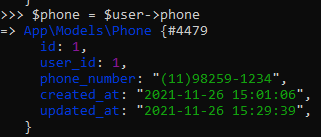




$user = User::with(["phone","posts"])->find(1)



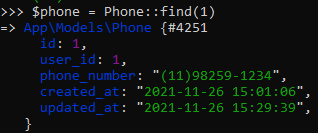
$phone = $user->phone



$phone->phone\_number



$phone = Phone::find(1)



$phone->phone\_number = "(11)99921-5638"

$phone->save()





$phone->fill(["phone\_number" => "(21)98835-0127"])->update()





## Eliminando um usuário

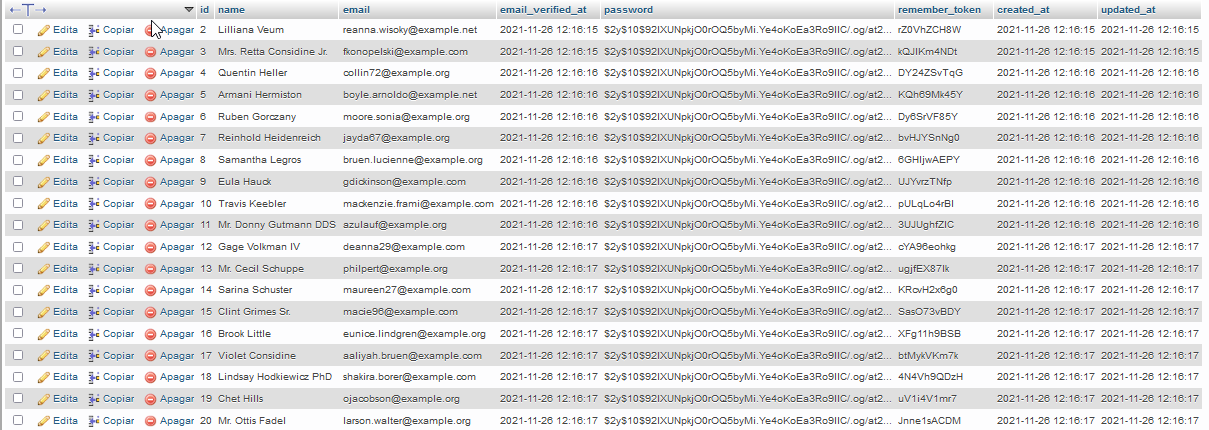
- Para eliminar um usuário, é necessário, antes de tudo, eliminar seus posts:

$user->posts()->delete()

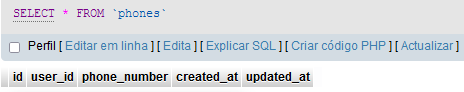


$user->delete()



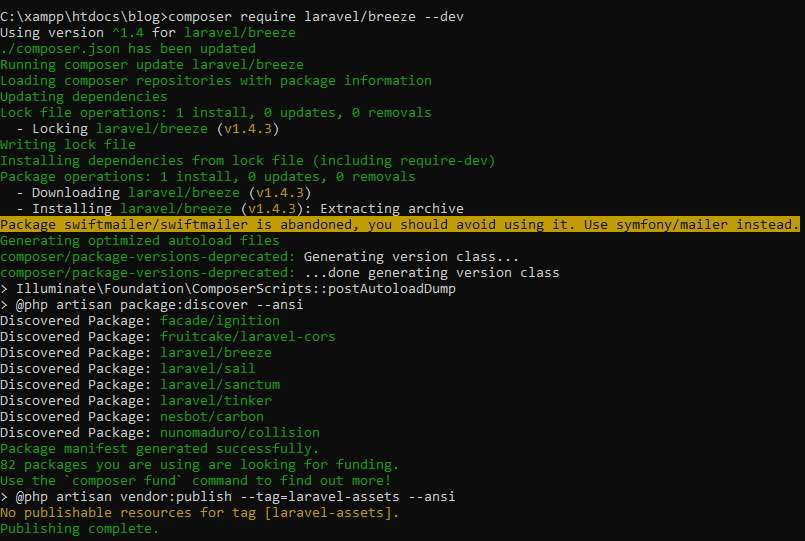


O telefone do usuário é eliminado em cascata:



# Aula 24 - Adicionar um sistema de autenticação completo ao projeto

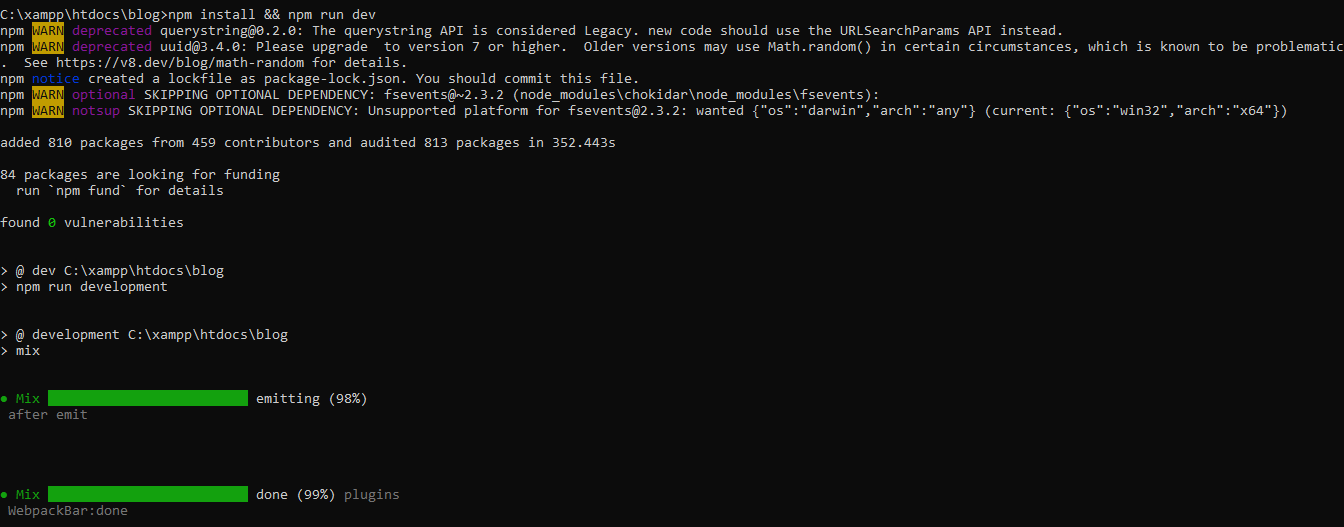
composer require laravel/breeze --dev



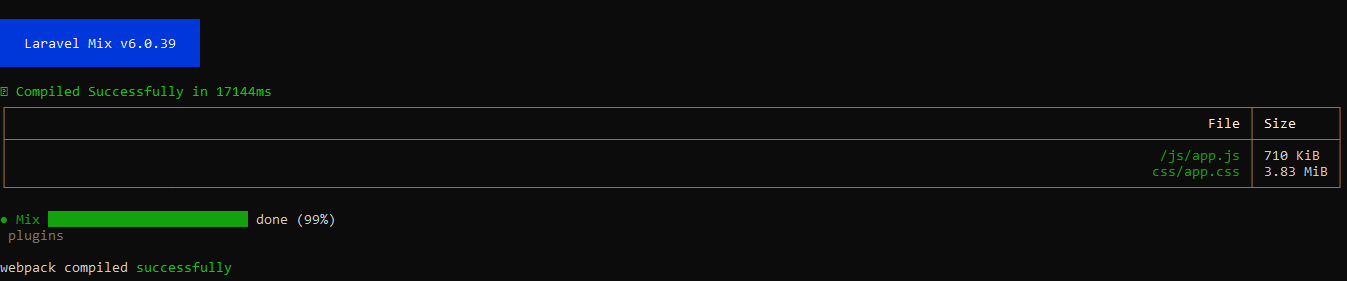
php artisan breeze:install



npm install && npm run dev

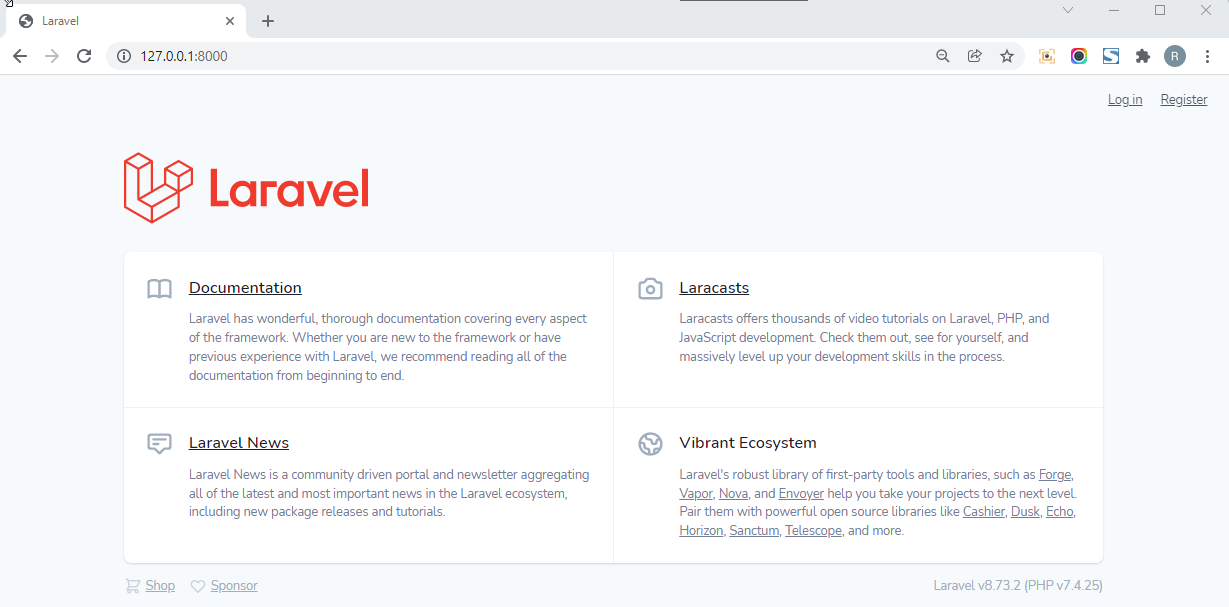






php artisan serve

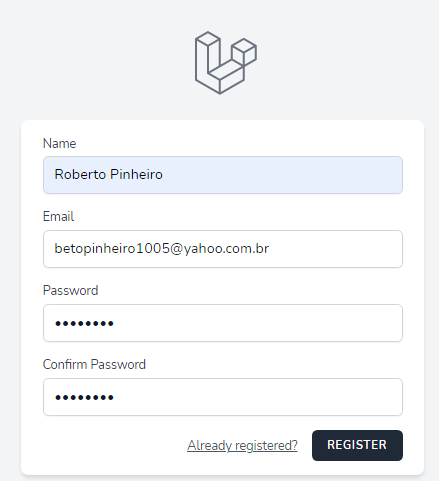
http://127.0.0.1:8000/



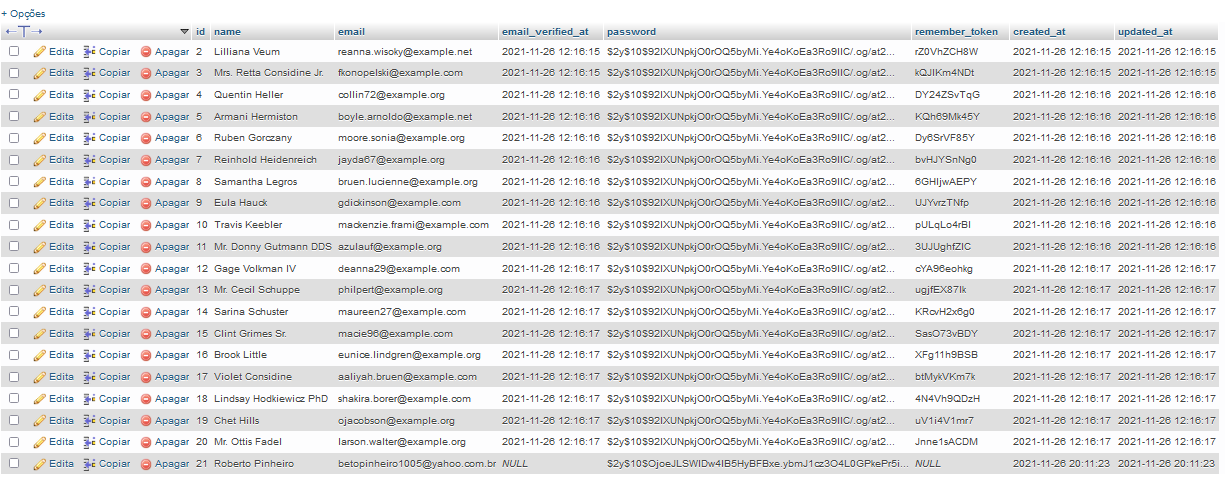
## Registrando um usuário

http://127.0.0.1:8000/register

Password: 12345678

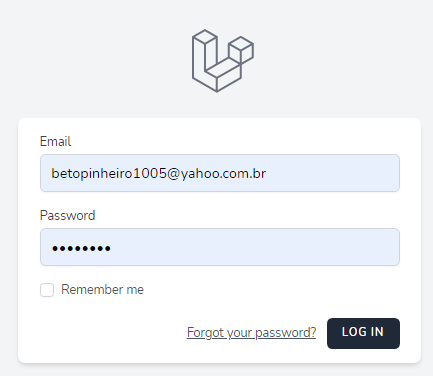


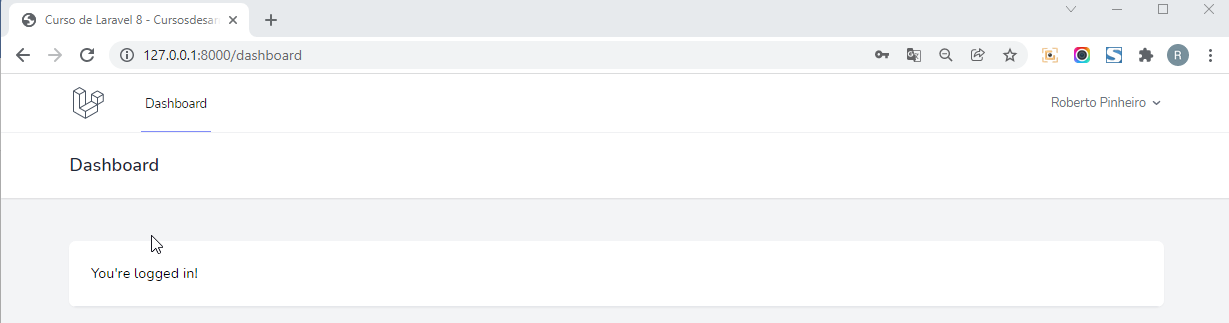




- Faça o Log out e em seguida o Log in

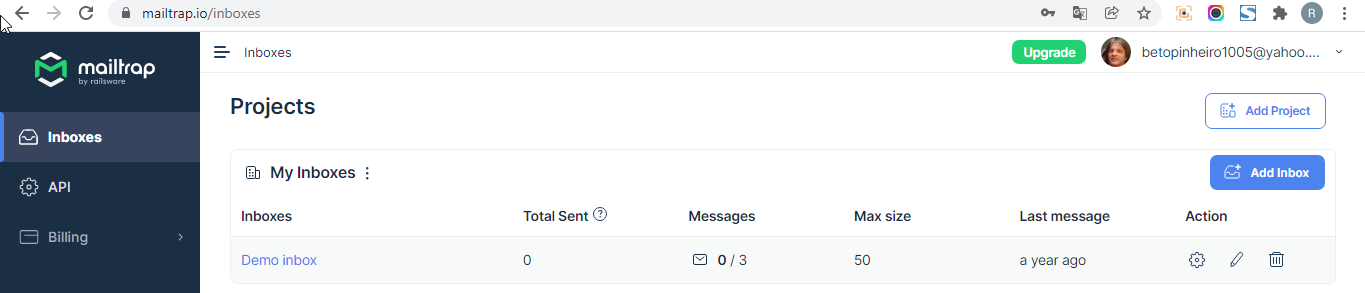
http://127.0.0.1:8000/login





# Aula 25 - Verificação de email para os novos registros

https://mailtrap.io/



- Clique no link "Demo inbox"

- Na aba Integrations, selecione "Laravel 7+"

MAIL\_MAILER=smtp

MAIL\_HOST=smtp.mailtrap.io

MAIL\_PORT=2525

MAIL\_USERNAME=b4abcd797d73bd

MAIL\_PASSWORD=22c14deab0947c

MAIL\_ENCRYPTION=tls

**.env**

APP\_NAME="Curso de Laravel 8 - Cursosdesarrolloweb"

APP\_ENV=local

APP\_KEY=base64:ejpjIGU95fkNemcwzBHpU9YMaDLBmGSWo5ebkN22F/Q=

APP\_DEBUG=true

APP\_URL=http://blog.test

LOG\_CHANNEL=stack

LOG\_DEPRECATIONS\_CHANNEL=null

LOG\_LEVEL=debug

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=blog

DB\_USERNAME=root

DB\_PASSWORD=

BROADCAST\_DRIVER=log

CACHE\_DRIVER=file

FILESYSTEM\_DRIVER=local

QUEUE\_CONNECTION=sync

SESSION\_DRIVER=file

SESSION\_LIFETIME=120

MEMCACHED\_HOST=127.0.0.1

REDIS\_HOST=127.0.0.1

REDIS\_PASSWORD=null

REDIS\_PORT=6379

MAIL\_MAILER=smtp

MAIL\_HOST=smtp.mailtrap.io

MAIL\_PORT=2525

MAIL\_USERNAME=b4abcd797d73bd

MAIL\_PASSWORD=22c14deab0947c

MAIL\_ENCRYPTION=tls

MAIL\_FROM\_ADDRESS=app@cursolaravel8.com

MAIL\_FROM\_NAME="${APP\_NAME}"

AWS\_ACCESS\_KEY\_ID=

AWS\_SECRET\_ACCESS\_KEY=

AWS\_DEFAULT\_REGION=us-east-1

AWS\_BUCKET=

AWS\_USE\_PATH\_STYLE\_ENDPOINT=false

PUSHER\_APP\_ID=

PUSHER\_APP\_KEY=

PUSHER\_APP\_SECRET=

PUSHER\_APP\_CLUSTER=mt1

MIX\_PUSHER\_APP\_KEY="${PUSHER\_APP\_KEY}"

MIX\_PUSHER\_APP\_CLUSTER="${PUSHER\_APP\_CLUSTER}"

**app\Models\User.php**

<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Relations\HasMany;

use Illuminate\Database\Eloquent\Relations\HasOne;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable implements MustVerifyEmail

{

use HasApiTokens, HasFactory, Notifiable;

/\*\*

\* The attributes that are mass assignable.

\*

\* @var string[]

\*/

protected $fillable = [

'name',

'email',

'password',

];

/\*\*

\* The attributes that should be hidden for serialization.

\*

\* @var array

\*/

protected $hidden = [

'password',

'remember\_token',

];

/\*\*

\* The attributes that should be cast.

\*

\* @var array

\*/

protected $casts = [

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

];

public function phone(): HasOne {

return $this->hasOne(Phone::class);

}

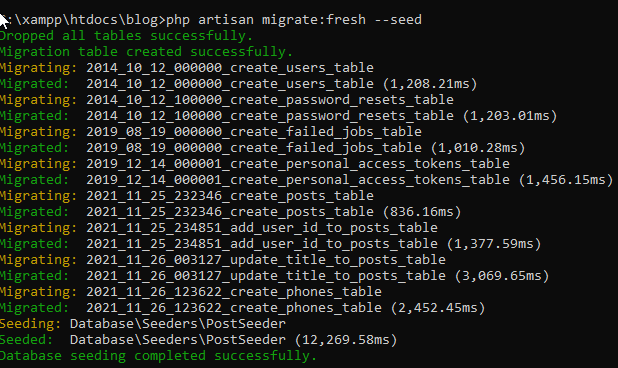
public function posts(): HasMany{

return $this->hasMany(Post::class);

}

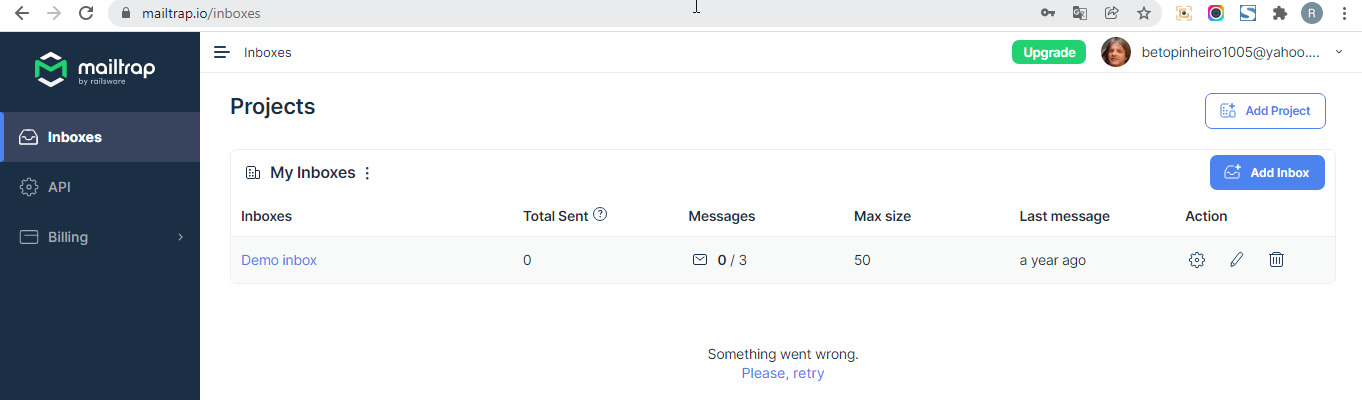
}

php artisan migrate:fresh --seed



php artisan serve





**routes\web.php**

<?php

use Illuminate\Support\Facades\Route;

/\*

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

| Web Routes

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

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| contains the "web" middleware group. Now create something great!

|

\*/

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

return view('welcome');

});

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

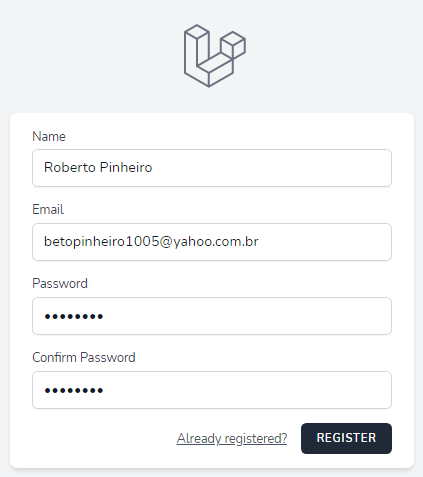
return view('dashboard');

})->middleware(['auth','verified'])->name('dashboard');

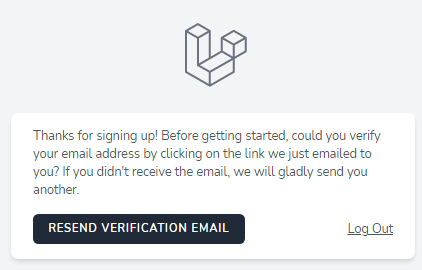
require \_\_DIR\_\_.'/auth.php';

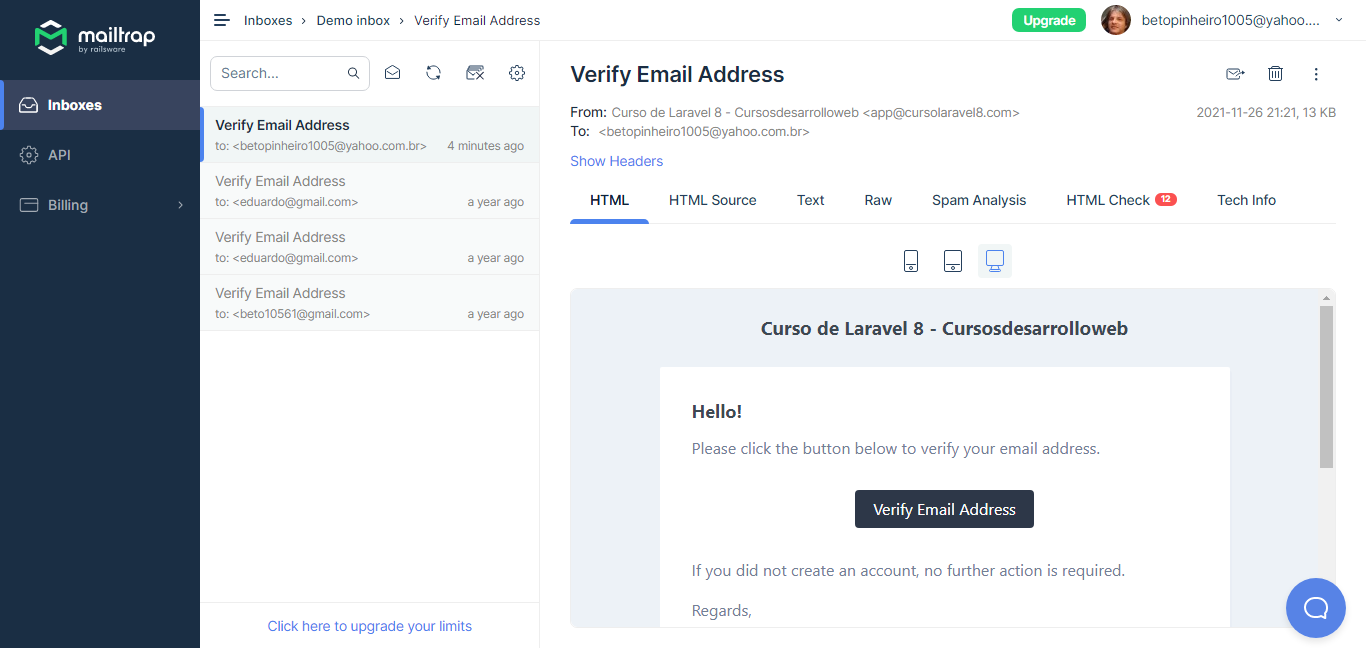
php artisan migrate:fresh

http://127.0.0.1:8000/register

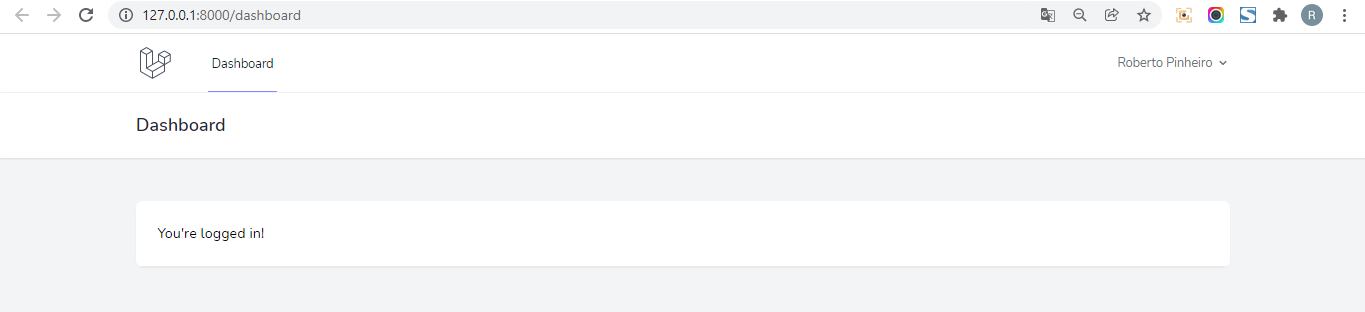


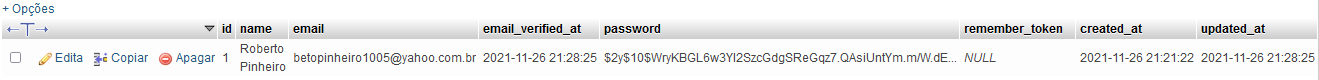
http://127.0.0.1:8000/verify-email





- Clique no botão "Verify Email Address":





# Aula 27 - Iniciar formulário de contato, rotas, controlador e navegação

php artisan make:controller ContactController



**routes\web.php**

<?php

use App\Http\Controllers\ContactController;

use Illuminate\Support\Facades\Route;

/\*

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

| Web Routes

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

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| contains the "web" middleware group. Now create something great!

|

\*/

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

return view('welcome');

});

Route::group(["middleware" => ['auth', 'verified']], function(){

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

return view('dashboard');

})->name('dashboard');

Route::get("/contact", [ContactController::class, "index"])->name("contact.index");

Route::post("/contact", [ContactController::class, "send"])->name("contact.send");

});

require \_\_DIR\_\_.'/auth.php';

**app\Http\Controllers\ContactController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ContactController extends Controller

{

public function index(){

return view("contact.form");

}

public function send(){

}

}

**resources\views\layouts\navigation.blade.php**

<nav x-data="{ open: false }" class="bg-white border-b border-gray-100">

<!-- Primary Navigation Menu -->

<div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">

<div class="flex justify-between h-16">

<div class="flex">

<!-- Logo -->

<div class="flex-shrink-0 flex items-center">

<a href="{{ route('dashboard') }}">

<x-application-logo class="block h-10 w-auto fill-current text-gray-600" />

</a>

</div>

<!-- Navigation Links -->

<div class="hidden space-x-8 sm:-my-px sm:ml-10 sm:flex">

<x-nav-link :href="route('dashboard')" :active="request()->routeIs('dashboard')">

{{ \_\_('Dashboard') }}

</x-nav-link>

<x-nav-link :href="route('contact.index')" :active="request()->routeIs('contact.index')">

{{ \_\_('Contact') }}

</x-nav-link>

</div>

</div>

<!-- Settings Dropdown -->

<div class="hidden sm:flex sm:items-center sm:ml-6">

<x-dropdown align="right" width="48">

<x-slot name="trigger">

<button class="flex items-center text-sm font-medium text-gray-500 hover:text-gray-700 hover:border-gray-300 focus:outline-none focus:text-gray-700 focus:border-gray-300 transition duration-150 ease-in-out">

<div>{{ Auth::user()->name }}</div>

<div class="ml-1">

<svg class="fill-current h-4 w-4" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 20 20">

<path fill-rule="evenodd" d="M5.293 7.293a1 1 0 011.414 0L10 10.586l3.293-3.293a1 1 0 111.414 1.414l-4 4a1 1 0 01-1.414 0l-4-4a1 1 0 010-1.414z" clip-rule="evenodd" />

</svg>

</div>

</button>

</x-slot>

<x-slot name="content">

<!-- Authentication -->

<form method="POST" action="{{ route('logout') }}">

@csrf

<x-dropdown-link :href="route('logout')"

onclick="event.preventDefault();

this.closest('form').submit();">

{{ \_\_('Log Out') }}

</x-dropdown-link>

</form>

</x-slot>

</x-dropdown>

</div>

<!-- Hamburger -->

<div class="-mr-2 flex items-center sm:hidden">

<button @click="open = ! open" class="inline-flex items-center justify-center p-2 rounded-md text-gray-400 hover:text-gray-500 hover:bg-gray-100 focus:outline-none focus:bg-gray-100 focus:text-gray-500 transition duration-150 ease-in-out">

<svg class="h-6 w-6" stroke="currentColor" fill="none" viewBox="0 0 24 24">

<path :class="{'hidden': open, 'inline-flex': ! open }" class="inline-flex" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M4 6h16M4 12h16M4 18h16" />

<path :class="{'hidden': ! open, 'inline-flex': open }" class="hidden" stroke-linecap="round" stroke-linejoin="round" stroke-width="2" d="M6 18L18 6M6 6l12 12" />

</svg>

</button>

</div>

</div>

</div>

<!-- Responsive Navigation Menu -->

<div :class="{'block': open, 'hidden': ! open}" class="hidden sm:hidden">

<div class="pt-2 pb-3 space-y-1">

<x-responsive-nav-link :href="route('dashboard')" :active="request()->routeIs('dashboard')">

{{ \_\_('Dashboard') }}

</x-responsive-nav-link>

</div>

<!-- Responsive Settings Options -->

<div class="pt-4 pb-1 border-t border-gray-200">

<div class="px-4">

<div class="font-medium text-base text-gray-800">{{ Auth::user()->name }}</div>

<div class="font-medium text-sm text-gray-500">{{ Auth::user()->email }}</div>

</div>

<div class="mt-3 space-y-1">

<!-- Authentication -->

<form method="POST" action="{{ route('logout') }}">

@csrf

<x-responsive-nav-link :href="route('logout')"

onclick="event.preventDefault();

this.closest('form').submit();">

{{ \_\_('Log Out') }}

</x-responsive-nav-link>

</form>

</div>

</div>

</div>

</nav>

**resources\views\contact\form.blade.php**

<x-app-layout>

<x-slot name="header">

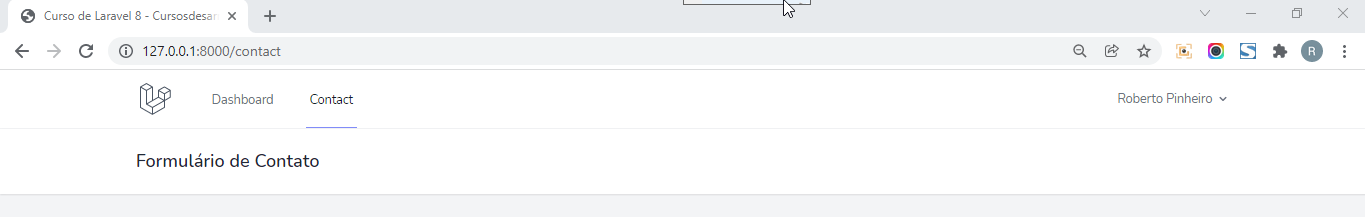
<h2 class="font-semibold text-xl text-gray-800 leading-tight">

{{ \_\_('Formulário de Contato') }}

</h2>

</x-slot>

</x-app-layout>

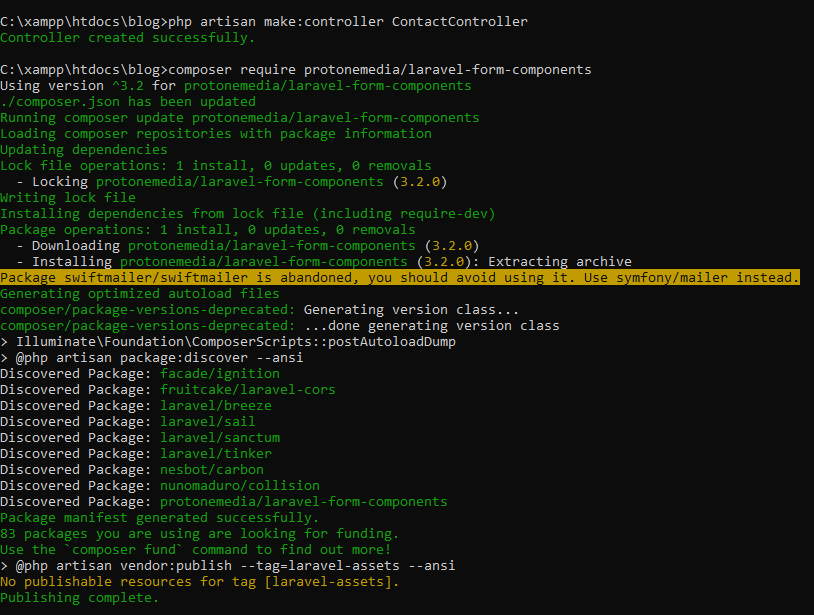


# Aula 28 - Pacote para gerar formulários com BladeX e criar formulário contato

## Instalando laravel-form-components

https://github.com/protonemedia/laravel-form-components

composer require protonemedia/laravel-form-components



**resources\views\contact\form.blade.php**

<x-app-layout>

<x-slot name="header">

<h2 class="font-semibold text-xl text-gray-800 leading-tight">

{{ \_\_('Formulário de Contato') }}

</h2>

</x-slot>

<div class="py-12">

<div class="max-w-2xl mx-auto sm:px-6 lg:px-8">

<div class="p-6 bg-white border-b border-gray-200">

<form action="{{ route("contact.send") }}" method="post">

@csrf

<div class="block mt-4">

<x-form-input name="subject" label="Escreva o motivo de sua mensagem" />

</div>

<div class="block mt-4">

<x-form-textarea name="message" label="Escreva sua mensagem aqui" />

</div>

<div class="flex items-center justify-end mt-4">

<x-form-submit>Enviar mensagem</x-form-submit>

</div>

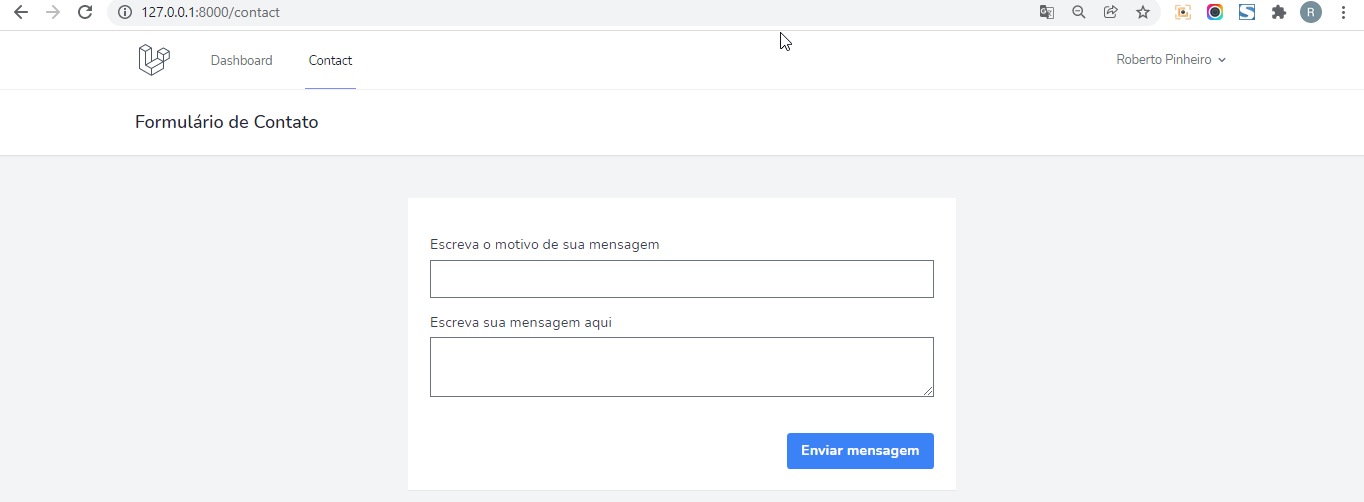
</form>

</div>

</div>

</div>

</x-app-layout>



# Aula 29 - Validar o formulário de contato e traduzir os erros

**app\Http\Controllers\ContactController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ContactController extends Controller

{

public function index(){

return view("contact.form");

}

public function send(Request $request){

$this->validate($request, [

"subject" => "required|string|min:5|max:100",

"message" => "required|string|min:20|max:3000"

]);

dd($request->input());

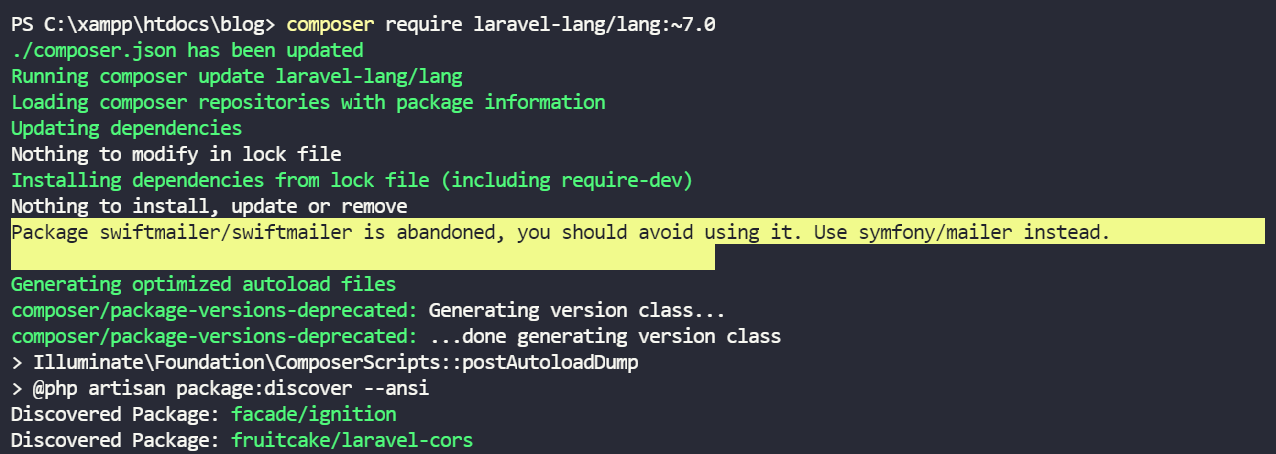
}

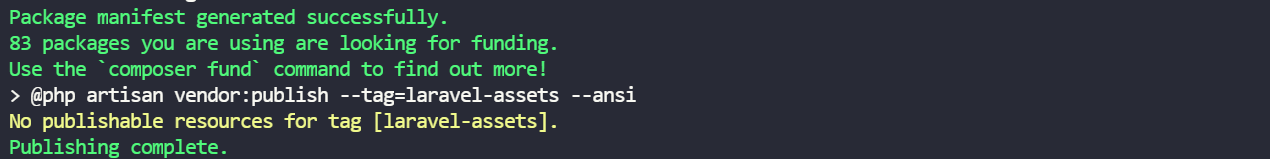
}

https://github.com/Laravel-Lang/lang

## Instalando laravel-lang

composer require laravel-lang/lang:~7.0





- Acesse vendor\laravel-lang\lang\src e copie a pasta pt\_BR para a pasta resources\lang.

## Configurando o idioma

**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' => 'UTC',

/\*

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

| 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',

/\*

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

| Application Fallback Locale

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

|

| The fallback locale determines the locale to use when the current one

| is not available. You may change the value to correspond to any of

| the language folders that are provided through your application.

|

\*/

'fallback\_locale' => 'pt\_BR',

/\*

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

| Faker Locale

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

|

| This locale will be used by the Faker PHP library when generating fake

| data for your database seeds. For example, this will be used to get

| localized telephone numbers, street address information and more.

|

\*/

'faker\_locale' => 'en\_US',

/\*

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

| Encryption Key

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

|

| This key is used by the Illuminate encrypter service and should be set

| to a random, 32 character string, otherwise these encrypted strings

| will not be safe. Please do this before deploying an application!

|

\*/

'key' => env('APP\_KEY'),

'cipher' => 'AES-256-CBC',

/\*

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

| Autoloaded Service Providers

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

|

| The service providers listed here will be automatically loaded on the

| request to your application. Feel free to add your own services to

| this array to grant expanded functionality to your applications.

|

\*/

'providers' => [

/\*

\* Laravel Framework Service Providers...

\*/

Illuminate\Auth\AuthServiceProvider::class,

Illuminate\Broadcasting\BroadcastServiceProvider::class,

Illuminate\Bus\BusServiceProvider::class,

Illuminate\Cache\CacheServiceProvider::class,

Illuminate\Foundation\Providers\ConsoleSupportServiceProvider::class,

Illuminate\Cookie\CookieServiceProvider::class,

Illuminate\Database\DatabaseServiceProvider::class,

Illuminate\Encryption\EncryptionServiceProvider::class,

Illuminate\Filesystem\FilesystemServiceProvider::class,

Illuminate\Foundation\Providers\FoundationServiceProvider::class,

Illuminate\Hashing\HashServiceProvider::class,

Illuminate\Mail\MailServiceProvider::class,

Illuminate\Notifications\NotificationServiceProvider::class,

Illuminate\Pagination\PaginationServiceProvider::class,

Illuminate\Pipeline\PipelineServiceProvider::class,

Illuminate\Queue\QueueServiceProvider::class,

Illuminate\Redis\RedisServiceProvider::class,

Illuminate\Auth\Passwords\PasswordResetServiceProvider::class,

Illuminate\Session\SessionServiceProvider::class,

Illuminate\Translation\TranslationServiceProvider::class,

Illuminate\Validation\ValidationServiceProvider::class,

Illuminate\View\ViewServiceProvider::class,

/\*

\* Package Service Providers...

\*/

/\*

\* Application Service Providers...

\*/

App\Providers\AppServiceProvider::class,

App\Providers\AuthServiceProvider::class,

// App\Providers\BroadcastServiceProvider::class,

App\Providers\EventServiceProvider::class,

App\Providers\RouteServiceProvider::class,

],

/\*

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

| Class Aliases

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

|

| This array of class aliases will be registered when this application

| is started. However, feel free to register as many as you wish as

| the aliases are "lazy" loaded so they don't hinder performance.

|

\*/

'aliases' => [

'App' => Illuminate\Support\Facades\App::class,

'Arr' => Illuminate\Support\Arr::class,

'Artisan' => Illuminate\Support\Facades\Artisan::class,

'Auth' => Illuminate\Support\Facades\Auth::class,

'Blade' => Illuminate\Support\Facades\Blade::class,

'Broadcast' => Illuminate\Support\Facades\Broadcast::class,

'Bus' => Illuminate\Support\Facades\Bus::class,

'Cache' => Illuminate\Support\Facades\Cache::class,

'Config' => Illuminate\Support\Facades\Config::class,

'Cookie' => Illuminate\Support\Facades\Cookie::class,

'Crypt' => Illuminate\Support\Facades\Crypt::class,

'Date' => Illuminate\Support\Facades\Date::class,

'DB' => Illuminate\Support\Facades\DB::class,

'Eloquent' => Illuminate\Database\Eloquent\Model::class,

'Event' => Illuminate\Support\Facades\Event::class,

'File' => Illuminate\Support\Facades\File::class,

'Gate' => Illuminate\Support\Facades\Gate::class,

'Hash' => Illuminate\Support\Facades\Hash::class,

'Http' => Illuminate\Support\Facades\Http::class,

'Js' => Illuminate\Support\Js::class,

'Lang' => Illuminate\Support\Facades\Lang::class,

'Log' => Illuminate\Support\Facades\Log::class,

'Mail' => Illuminate\Support\Facades\Mail::class,

'Notification' => Illuminate\Support\Facades\Notification::class,

'Password' => Illuminate\Support\Facades\Password::class,

'Queue' => Illuminate\Support\Facades\Queue::class,

'RateLimiter' => Illuminate\Support\Facades\RateLimiter::class,

'Redirect' => Illuminate\Support\Facades\Redirect::class,

// 'Redis' => Illuminate\Support\Facades\Redis::class,

'Request' => Illuminate\Support\Facades\Request::class,

'Response' => Illuminate\Support\Facades\Response::class,

'Route' => Illuminate\Support\Facades\Route::class,

'Schema' => Illuminate\Support\Facades\Schema::class,

'Session' => Illuminate\Support\Facades\Session::class,

'Storage' => Illuminate\Support\Facades\Storage::class,

'Str' => Illuminate\Support\Str::class,

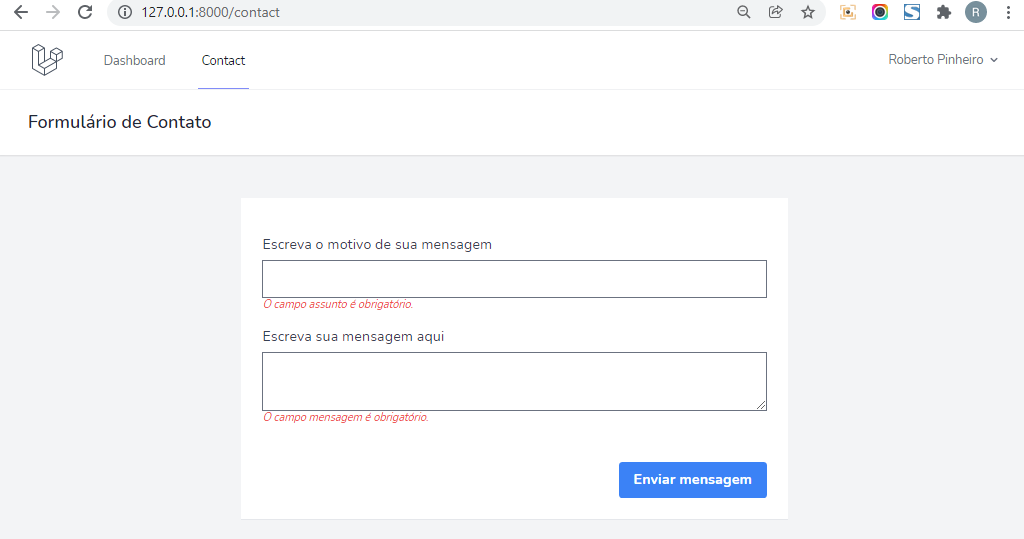
'URL' => Illuminate\Support\Facades\URL::class,

'Validator' => Illuminate\Support\Facades\Validator::class,

'View' => Illuminate\Support\Facades\View::class,

],

];



# Aula 30 - Mailable com Markdown para enviar formulário de contato por email

php artisan make:mail SendContactForm



**app\Mail\SendContactForm.php**

<?php

namespace App\Mail;

use Illuminate\Bus\Queueable;

use Illuminate\Contracts\Queue\ShouldQueue;

use Illuminate\Mail\Mailable;

use Illuminate\Queue\SerializesModels;

class SendContactForm extends Mailable

{

use Queueable, SerializesModels;

public string $textSubject;

public string $textMessage;

/\*\*

\* Create a new message instance.

\*

\* @return void

\*/

public function \_\_construct(string $subject, string $message)

{

$this->textSubject = $subject;

$this->textMessage = $message;

}

/\*\*

\* Build the message.

\*

\* @return $this

\*/

public function build()

{

return $this

->subject("Formulário de contato - " . config("app.name"))

->markdown('emails.contact');

}

}

**resources\views\emails\contact.blade.php**

@component('mail::message')

# Olá admin,

<br>

<p>Você recebeu uma nova mensagem a partir do formulário de contato em {{ config('app.name') }}</p>

<p>Motivo da mensagem: {{ $textSubject }}</p>

<p>{{ $textMessage }}</p>

@endcomponent

Atenção: Todo o texto tem que estar tabulado à esquerda.

php artisan vendor:publish

- Escolha a opção laravel-mail



- Será criada a pasta resources\views\vendor\mail

# Aula 31 - Renderizar no browser e enviar por email Mailables

**routes\web.php**

<?php

use App\Http\Controllers\ContactController;

use Illuminate\Support\Facades\Route;

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

return view('welcome');

});

Route::group(["middleware" => ['auth', 'verified']], function(){

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

return view('dashboard');

})->name('dashboard');

Route::get("/contact", [ContactController::class, "index"])->name("contact.index");

Route::post("/contact", [ContactController::class, "send"])->name("contact.send");

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

return new \App\Mail\SendContactForm("Motivo", "Mensagem");

});

});

require \_\_DIR\_\_.'/auth.php';



**app\Http\Controllers\ContactController.php**

<?php

namespace App\Http\Controllers;

use App\Mail\SendContactForm;

use App\Models\User;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Mail;

class ContactController extends Controller

{

public function index(){

return view("contact.form");

}

public function send(Request $request){

$this->validate($request, [

"subject" => "required|string|min:5|max:100",

"message" => "required|string|min:20|max:3000"

]);

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

Mail::to(User::first())->send(

new SendContactForm(

$request->input("subject"),

$request->input("message"),

)

);

}

}

# Aula 32 - Mostrar mensagens flash com componente Flash Messages

**resources\views\components\flash-messages.blade.php**

@if($message = Session::get("success"))

<div class="text-white px-6 py-4 border-0 rounded relative mb-4 bg-green-500">

<span class="text-xl inline-block mr-5 align-middle">

<i class="fas fa-bell">

</span>

<span class="inline-block align-middle mr-8">

<b class="capitalize">{{ $message }}</b>

</span>

</div>

@endif

@if ($message = Session::get("error"))

<div class="text-white px-6 py-4 border-0 rounded relative mb-4 bg-red-500">

<span class="text-xl inline-block mr-5 align-middle">

<i class="fas fa-bell">

</span>

<span class="inline-block align-middle mr-8">

<b class="capitalize">{{ $message }}</b>

</span>

</div>

@endif

**resources\views\layouts\app.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">

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

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

<!-- Fonts -->

<link rel="stylesheet" href="https://fonts.googleapis.com/css2?family=Nunito:wght@400;600;700&display=swap">

<!-- Styles -->

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

<!-- Scripts -->

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

</head>

<body class="font-sans antialiased">

<div class="min-h-screen bg-gray-100">

@include('layouts.navigation')

<!-- Page Heading -->

<header class="bg-white shadow">

<div class="max-w-7xl mx-auto py-6 px-4 sm:px-6 lg:px-8">

{{ $header }}

</div>

</header>

<!-- Page Content -->

<main>

<x-flash-messages />

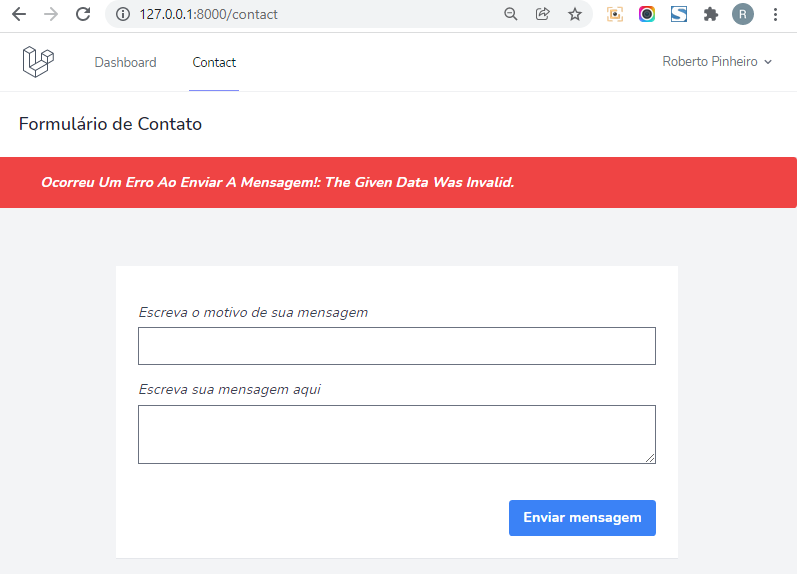
{{ $slot }}

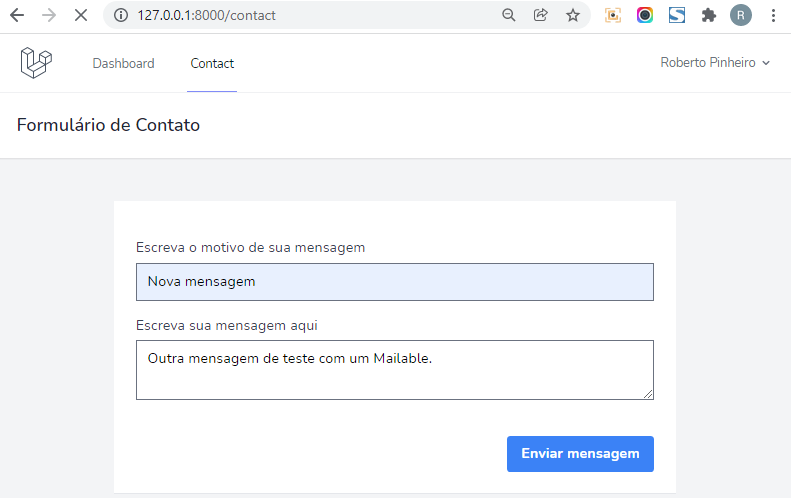
</main>

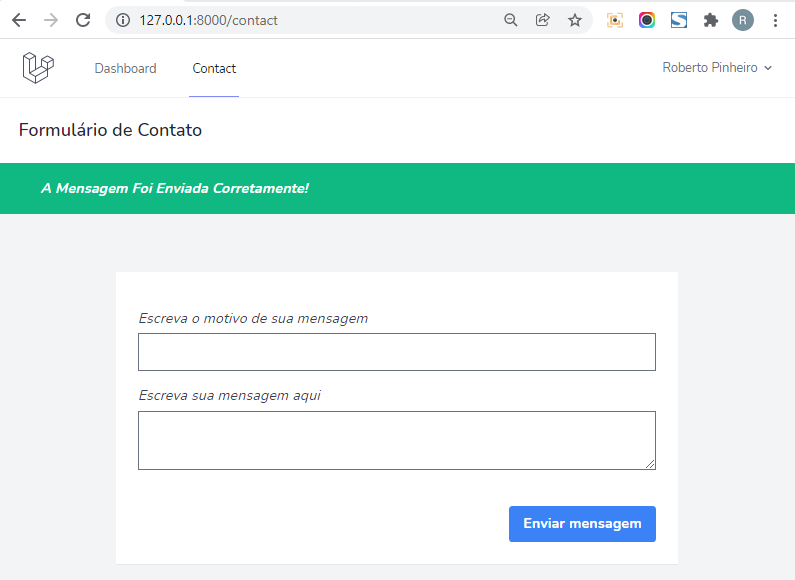
</div>

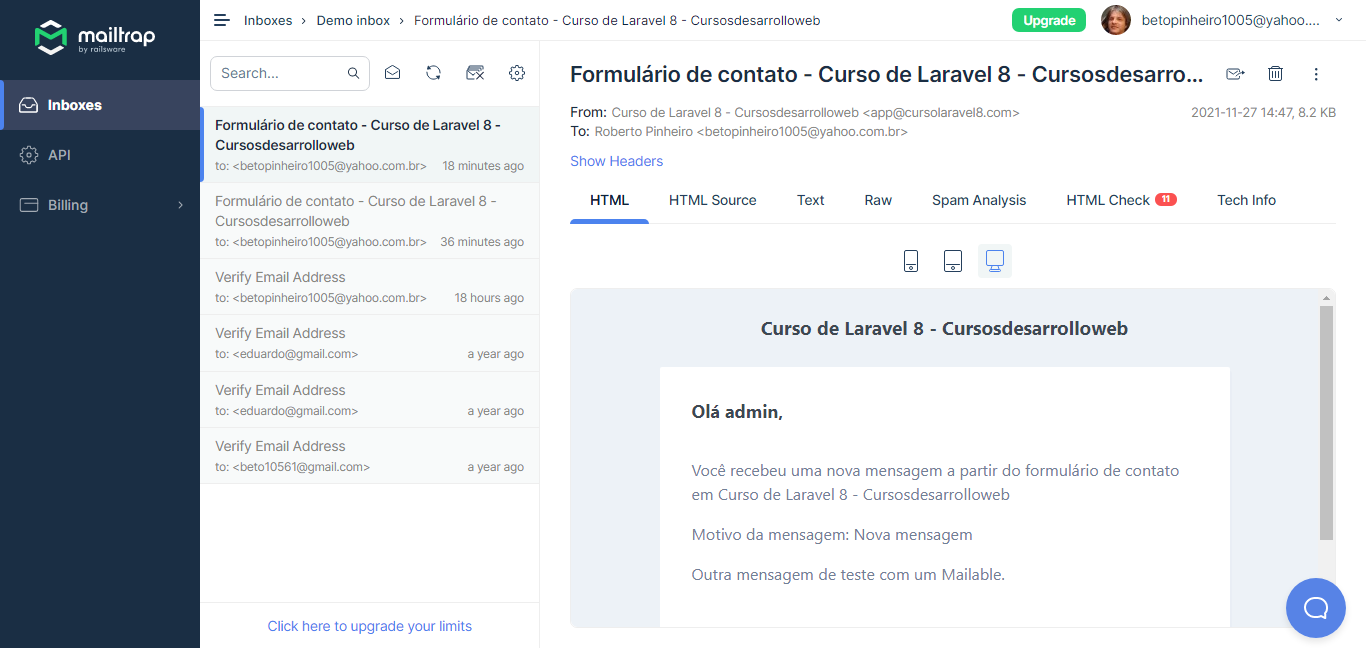
</body>

</html>









# Aula 33 - Validar o formulário de contato com uma classe FormRequest

php artisan make:request ContactFormRequest



**app\Http\Requests\ContactFormRequest.php**

<?php

namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class ContactFormRequest extends FormRequest

{

public function authorize()

{

return true;

}

public function rules()

{

return [

"subject" => "required|string|min:5|max:100",

"message" => "required|string|min:20|max:3000"

];

}

}

**app\Http\Controllers\ContactController.php**

<?php

namespace App\Http\Controllers;

use App\Http\Requests\ContactFormRequest;

use App\Mail\SendContactForm;

use App\Models\User;

use Illuminate\Support\Facades\Mail;

class ContactController extends Controller

{

public function index(){

return view("contact.form");

}

public function send(ContactFormRequest $request){

try {

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

Mail::to(User::first())->send(

new SendContactForm(

$request->input("subject"),

$request->input("message"),

)

);

return back()->with("success", "A mensagem foi enviada corretamente!");

} catch (\Exception $exception) {

return back()->with("error", "Ocorreu um erro ao enviar a mensagem!: " . $exception->getMessage());

}

}

}

