**Aula 8 – CRUD-Produto em Laravel**

**Controlde de Permissão**

**Objetivos:**

* Implementar o controle de permissão de acesso, o qual controla o acesso às funcionalidades da Aplicação de acordo com o tipo de permissão do usuário.

Existem pacotes prontos para Controle de Permissão e um dos mais utilizados é o Laravel Permission mantido pela Spatie (<https://spatie.be/docs/laravel-permission/v6/introduction>). Este pacote é bem completo e trabalha com atribuição de roles (papéis) aos usuários e permissões associadas aos roles. Também já vem com um Middleware de Autorização integrado.

Da mesma forma que fizemos no Controle de Sessão (Login), aqui também vamos criar nosso próprio Sistema de Controle de Permissão através de um Middleware nosso.

**8.1. Definindo as Permissões na nossa Aplicação**

Lembrando que temos três tipos de permissão:

• Administrador (Admin);

• Normal;

• Somente Leitura (Leitura).

É preciso definir quais partes da Aplicação exigirão um controle de acesso que, em geral, é definido pelas regras de negócio da Aplicação. Vamos utilizar os seguintes critérios:

• Página Home: liberado acesso para qualquer usuário;

• Listar Produtos: Administrador e Normal têm acesso completo. Usuário Somente Leitura poderá Listar, mas não poderá Editar e Excluir produtos;

• Cadastrar Produtos: Administrador e Normal têm acesso completo. Usuário Somente Leitura Não tem acesso;

• Usuários: Somente o Administrador tem acesso à Manutenção de Usuários. Normal e Somente Leitura não podem acessar as opções desse item do menu.

* 1. **Implementando o Middleware de Permissão**

Primeiro é preciso criar o Middleware via artisan:

php artisan make:middleware Permissao

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

O comando acima vai criar o arquivo app\Http\Middleware\Permissao.php que deve ter o seguinte conteúdo:

Texto

O conteúdo gerado por IA pode estar incorreto.

É dentro do método handle() que devemos colocar as regras do nosso mddleware para decidir se o pedido deve prosseguir ou não. Em nosso caso, devemos verificar se a permissão do usuário é suficiente para acessar o recurso. Podemos fazer isso verificando a informação usuario.permissao na Sessão, porém temos que comparar com a permissão registrada na rota. Assim, vamos receber essa permissão em um parâmetro chamado $permissaoNecessaria.

Troque todo o conteúdo do middleware Autenticacao.php para:

<?php

namespace App\Http\Middleware;

use Closure;

use Illuminate\Http\Request;

use Symfony\Component\HttpFoundation\Response;

class Permissao

{

/\*\*

\* Handle an incoming request.

\*

\* @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next

\*/

public function handle(Request $request, Closure $next, string $permissaoNecessaria): Response

{

$permissaoUsuario = session('usuario.permissao');

$hierarquia = [

'Admin' => 3,

'Normal' => 2,

'Leitura' => 1,

];

// Se a permissão do usuário for maior ou igual à exigida

if ($hierarquia[$permissaoUsuario] >= $hierarquia[$permissaoNecessaria]) {

return $next($request);

}

abort(403, 'Permissão insuficiente');

}

}

Para que o Middleware faça seu trabalho, é preciso que seja registrado no mecanismo de Middleware do Laravel. Isso é feito no arquivo bootstrap\app.php:

Texto

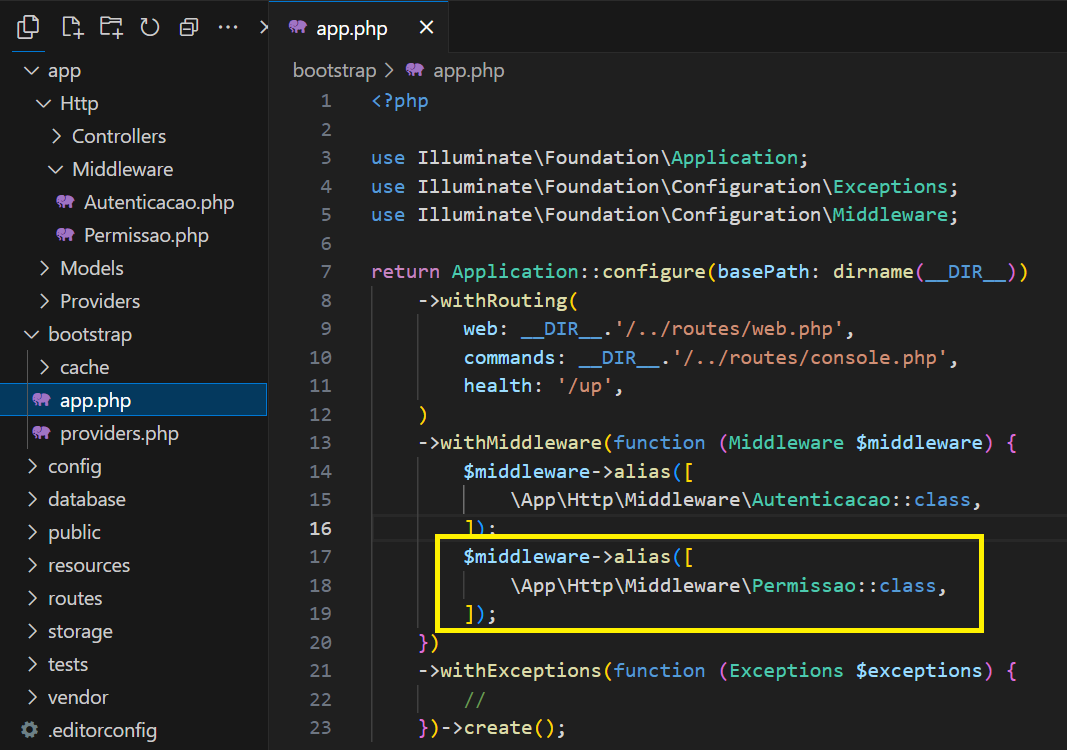
O conteúdo gerado por IA pode estar incorreto.

O registro é feito dentro da função ->withMiddleware(function (Middleware $middleware). Note que já existe o nosso Middleware de Autenticação registrado. Então acrescente o middleware de Permissão da seguinte forma:

$middleware->alias([

\App\Http\Middleware\Permissao::class,

]);



* 1. **Atribuindo o Middleware à Rota**

Vamos atribuir o Middleware para as rotas de acordo com a permissão necessária. Por exemplo, toda a parte de Gerenciamento de Usuários (rotas iniciando com /user) necessitam de permissão Admin. Assim, no arquivo routes\web.php coloque todas as rotas que iniciam com /user dentro do grupo que passa pelo middleware Permissão, como a seguir:

Route::middleware(Permissao::class.':Admin')->group(function () {

Route::get('/user',[UserController::class,'index']);

Route::post('/user/salvarnovo',[UserController::class,'salvarnovo']);

Route::post('/user/salvarEditado',[UserController::class,'salvarEditado']);

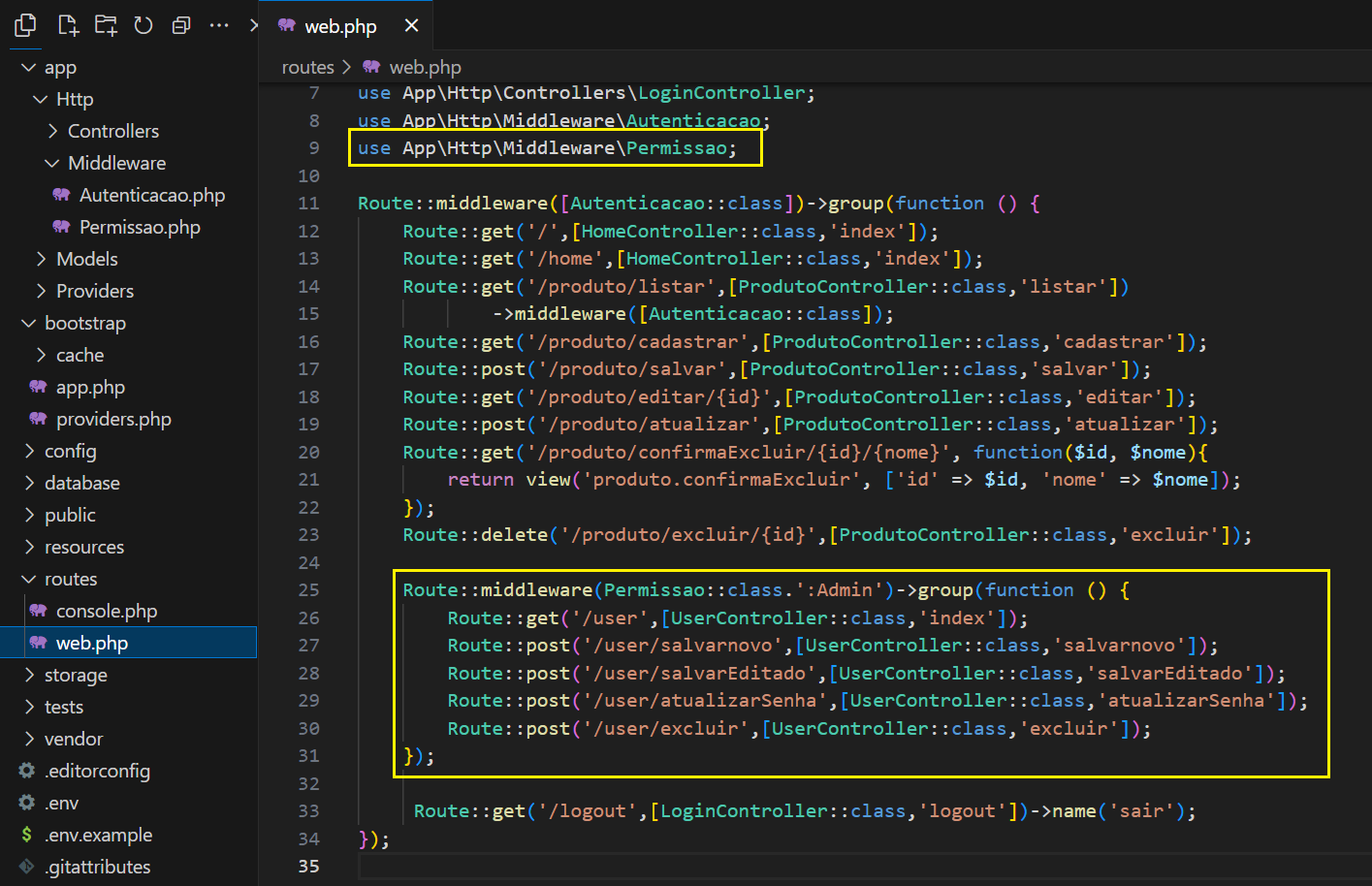
Route::post('/user/atualizarSenha',[UserController::class,'atualizarSenha']);

Route::post('/user/excluir',[UserController::class,'excluir']);

});

E também acrescente a linha a seguir no início do arquivo:

use App\Http\Middleware\Permissao;



Salve. Vá até o navegador e, caso já esteja logado, saia da Aplicação pelo Sair do menu Perfil do usuário. Agora faça login com uma conta de um usuário com permissão Admin. Tente acessar o menu Usuários e veja que será permitido visualizar a lista de usuários, bem como Editar, Ressetar a Senha, Excluir e Cadastrar um novo Usuário.

Faça o mesmo para a permissão Normal, a qual deve proteger o acesso às rotas associadas à manutenção de Produtos relacionadas ao Cadastro, Edição e Exclusão de Produtos. Para isso, coloque as rotas relacionadas dentro de um grupo que chama o middleware Permissão com acesso Normal, como a seguir:

Route::middleware(Permissao::class.':Normal')->group(function () {

Route::get('/produto/cadastrar',[ProdutoController::class,'cadastrar']);

Route::post('/produto/salvar',[ProdutoController::class,'salvar']);

Route::get('/produto/editar/{id}',[ProdutoController::class,'editar']);

Route::post('/produto/atualizar',[ProdutoController::class,'atualizar']);

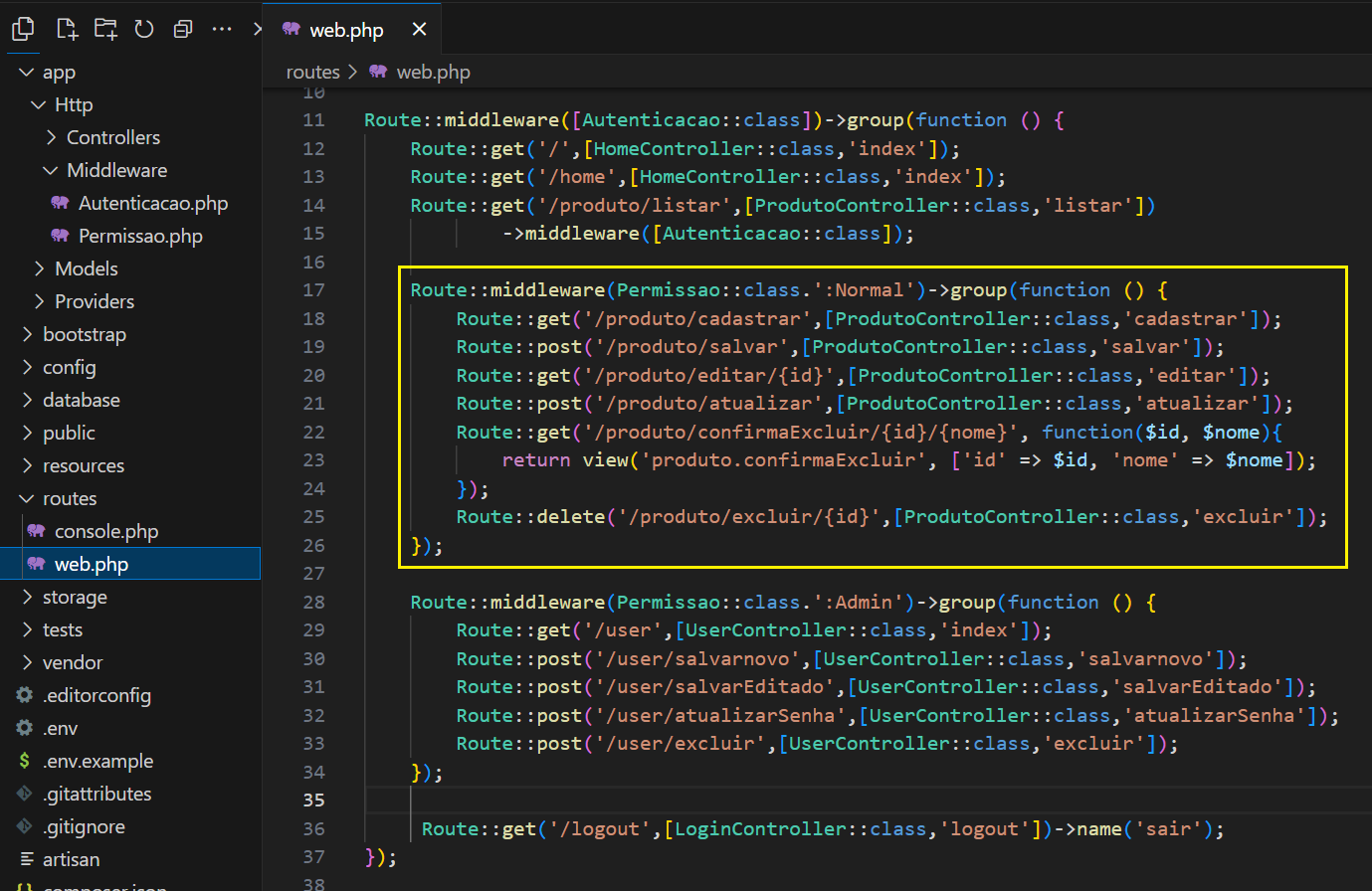
Route::get('/produto/confirmaExcluir/{id}/{nome}', function($id, $nome){

return view('produto.confirmaExcluir', ['id' => $id, 'nome' => $nome]);

});

Route::delete('/produto/excluir/{id}',[ProdutoController::class,'excluir']);

});



Salve e teste. Faça login com um usuário com permissão Normal. Note que todas as funcionalidades do gerenciamento de Produtos serão permitidas. Depois faça login com um usuário com permissão Leitura. Note que este usuário só terá acesso ao Listar Produtos. As demais funcionalidades NÃO serão permitidas.