CREACION Y LOGUEO DEL SUPER ADMIN CON SU DASHBOARD INDEPENDIENTE

**1 PASO UNO CREAR LA PROTECCION: EL GUARD Y EL MIDDLEWARE**

Crear Authentication para admins

Lo que queremos hacer es crear un middleware admin para que al acceder a la ruta /admin compruebe que está logueado y además es admin

1 En el Modelo Admin (o como se llame):

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\*HasFactory*;

use Illuminate\Database\Eloquent\*Model*;

use Illuminate\Foundation\Auth\*User* as Authenticatable;

*class* Admin extends *Authenticatable*

{

Con use Illuminate\Foundation\Auth\User as Authenticatable y extiende de esa clase, lo que hace es usar el Auth normal de Laravel es decir para que puedas entrar tienes que estar autentificado y a partir de ahora vemos si eres admin o no

Creamos el guard Admin.php

 protected $guard = 'admin';

Cuando defines **protected $guard = 'admin';** dentro del modelo **Admin** en Laravel, estás indicando que este modelo debería utilizar un guard específico llamado **'admin'** para la autenticación. Aquí hay algunos pasos que puedes seguir para continuar:

Configuración del Guard:

Asegúrate de que tengas configurado un guard llamado 'admin' en el archivo config/auth.php. Deberías ver algo como esto:

 'guards' => [

        'web' => [

            'driver' => 'session',

            'provider' => 'users',

        ],

        'admin' => [

            'driver' => 'session',

            'provider' => 'admins',

        ],

    ],

1. En este ejemplo, se ha configurado un guard llamado **'admin'** que utiliza el driver **'session'** y el proveedor **'admins'**. Esto significa que cuando se autentica un usuario usando el guard **'admin'**, Laravel buscará las credenciales en el proveedor **'admins'**.
2. **Configuración del Proveedor:**
   * A continuación, asegúrate de que tengas configurado el proveedor correspondiente **'admins'** en el mismo archivo **config/auth.php**. Debería verse así:
3. 'providers' => [
4. 'users' => [
5. 'driver' => 'eloquent',
6. 'model' => App\Models\*User*::class,
7. ],
8. 'admins' => [
9. 'driver' => 'eloquent',
10. 'model' => App\Models\*Admin*::class,
11. ],

En este caso, el proveedor **'admins'** está configurado para utilizar el modelo **Admin** que mencionaste.

**Autenticación:**

Cuando quieras autenticar un usuario como administrador, utiliza el guard **'admin'** en lugar del guard predeterminado **'web'**. Por ejemplo:

if (Auth::guard('admin')->attempt(['email' => $email, 'password' => $password])) {

// Autenticación exitosa como administrador

}

De esta manera, Laravel usará el guard **'admin'** para autenticar al usuario y buscará las credenciales en el proveedor **'admins'**.

Con estos pasos, habrás configurado correctamente el guard **'admin'** para el modelo **Admin** en Laravel. Ahora, cuando autentiques usuarios como administradores, Laravel utilizará esta configuración específica. ¡Espero que esto te ayude a continuar! Si tienes más preguntas o necesitas más ayuda, no dudes en preguntar.

4 Una vez que tenemos el guard creamos el middleware Admin.

*class* Admin

{

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

    {

        if(!auth()->guard('admin')->check())

        {

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

        }

        return $next($request);

    }

Que da una capa de protección extra a las rutas de admin. Si está logueado y además es admin pasa si no redirige a admin/login. Es independiente de si tu te logueas por el login normal. Aquí se comprueba que estás logueado como admin

5. Para que el middleware esté operativo, como siempre hay que registrarlo en el Kernel.php

'admin' => \App\Http\Middleware\*Admin*::class,

En

protected $middlewareAliases = [

**6 Por ultimo protegemos las rutas**

//Admin Routes

*Route*::prefix('admin')->namespace('App\Http\Controllers\Admin')->name('admin.')->group(*function* () {

    //Admin Login Routes

*Route*::match(['get', 'post'],'login', [*AdminController*::class, 'login'])->name('login');

*Route*::group(['middleware' => ['admin']], *function* () {

         //Admin Dashboard Routes

*Route*::get('dashboard', [*AdminController*::class, 'dashboard'])->name('dashboard');

*Route*::post('logout', [*AdminController*::class, 'logout'])->name('logout');

     }

     );

});

**Con el middleware admin y todo listo**

**2 CREAR EL SUPERADMIN**

**Ye he creado migracion modelo….**

**2.2 Creo el seeder AdminsTableSeeder**

*class* AdminsTableSeeder extends *Seeder*

{

    /\*\*

     \* Run the database seeds.

     \*/

    public *function* run(): void

    {

         $adminRecords = [

            [

                'id' => 1,

                'name' => 'Super Admin',

                'type' => 'superadmin',

                'vendor\_id' => 0,

                'mobile' => '1234567890',

                'email' => 'admin@admin.com',

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

                'image' => '',

                'status' => 1,

            ],

            ];

*Admin*::insert($adminRecords);

    }

}

**2.3 Importo el seeder en el DatabaseSeeder y hago la migración**

public *function* run(): void

    {

        $this->call(*AdminsTableSeeder*::class);

    }

**2.4 Creo los métodos login y logout en el AdminController**

 public *function* login (*Request* $request){

        if($request->isMethod('post')){

            $data = $request->all();

             //echo "<pre>"; print\_r($data); die;

            if(*Auth*::guard('admin')->attempt(['email'=>$data['email'],'password'=>$data['password'],'status'=>1])){

                return redirect('admin/dashboard');

            }else{

                return redirect()->back()->with('error\_message','Invalid Username or Password');

            }

        }

        return view('admin.login');

    }

    public *function* logout (*Request* $request){

*Auth*::guard('admin')->logout();

        $request->session()->invalidate();

        $request->session()->regenerateToken();

        return redirect(route('admin.login'));

    }

**Al usar:**

*Route*::match(['get', 'post'],'login', [*AdminController*::class, 'login'])->name('login');

**En las rutas, en el método login hago la comprobación de por que método viene el request**

**2.4 el logout nuevo por defecto en la vista es**

<x-dropdown-link class="dropdown-item" :href="route('admin.logout')"

                                onclick="event.preventDefault();

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

                                             <i class="ti-power-off text-primary"></i>

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

                        </x-dropdown-link>