Manejar roles para usuarios con el paquete spatie/laravel-permission

Que es spatie/laravel-permission?

Es un paquete que nos permite asociar a nuestros usuarios roles y permisos que serán guardados en nuestra base de datos sin tener que crear las migraciones manualmente, sino que ya el paquete nos las trae listas, además nos ofrece un par de modelos para los roles y permisos con una serie de métodos que nos garantizan mucha simplicidad.

Instalación

La instalación se hace a través de Composer con el siguiente comando

```
composer require spatie/laravel-permission
```

Después de la instalación, debes agregar el proveedor de servicios y publicar las migraciones y configuraciones. Ejecuta los siguientes comandos:

```
php artisan vendor:publish --
provider="Spatie\Permission\PermissionServiceProvider"
```

```
php artisan migrate
```

Uso del paquete

Debemos modificar nuestro modelo User para agregar el trait del paquete llamado **HasRoles**, el cual es el encargado de ofrecernos una serie de métodos para trabajar con roles y permisos que estarán asociados a nuestro modelo. No olvides agregar el namespace antes de usar el trait.

```
use Spatie\Permission\Traits\HasRoles;

class User extends Authenticatable
{
    use HasRoles;

    // ...
}
```

Definición de roles

Podemos definir los roles dentro de un seeder ya que este nos proporciona una forma de predefinir los roles en la base de datos cuando se está desarrollando la aplicación.

Se hace de la siguiente manera:

1. Creamos el seeder:

php artisan make:seeder <nombre_seeder>

```
php artisan make:seeder RolesSeeder
```

2. Dentro de la carpeta database/seeders podemos encontrar el seeder creado. Allí podemos encontrar una función llamada 'run' en el cual vamos a crear los roles:

```
namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Spatie\Permission\Models\Role;

class RolesSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        // CREAR LOS ROLES
       Role::create(['name' => 'admin']);
       Role::create(['name' => 'cafeteria']);
       Role::create(['name' => 'aprendiz']);
    }
}
```

3. Lo siguiente es configurar este seeder dentro del DatabaseSeeder.php de la siguiente manera:

```
class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     *
     * @return void
     */
    public function run()
    {
        $this->call(RolesSeeder::class);
    }
}
```

4. Y por último proceder a ejcutar el seeder:

```
php artisan db:seed
```

Asignacion de roles a los usuarios

La asignación de roles se puede hacer en cualquier lugar en la que usted vaya a crear un usuario. Pero lo mas común es hacerlo dentro del **registro** del usuario en el **AuthController**.

Recuerde adaptar la función del registro según sus requisitos y logica de negocio. Esto es solo un ejemplo.

```
class AuthController extends Controller
{
    /**
     * Registro de usuario
     */
    public function signUp(Request $request)
    {
        $request->validate([
            'name' => 'required|string',
            'email' => 'required|string|email',
            'password' => 'required|string',
            'esAdmin' => 'required|boolean'
        ]);

        $user = User::create([
            'name' => $request->name,
            'email' => $request->email,
            'password' => bcrypt($request->password)
        ]);

        // Podemos diferenciar el rol que quiere elegir el usuario a través del
uso de booleanos
        $esAdmin = $request->esAdmin;

        if ($esAdmin) {
            // Si es admin podemos asignar el rol de admin
            $user->assignRole('admin');
        } else {
            // Si no es porque es aprendiz
            $user->assignRole('aprendiz');
        }

        return response()->json([
            'message' => 'Successfully created user!'
        ], 201);
    }
}
```

Implementación de middleware con roles

Antes de implementar nuestro middleware, debemos indicar donde está el middleware que nos proporciona spatie/permission:

Vamos a esta ruta: app > Http > Kernel.php. Allí colocamos lo siguiente:

```
'role' => \Spatie\Permission\Middlewares\RoleMiddleware::class,
```

dentro de:

```
protected $routeMiddleware = [

    //...
    'role' => \Spatie\Permission\Middlewares\RoleMiddleware::class,
];
```

Ahora si podemos a proteger las rutas según el rol del usuario. Lo podemos hacer dentro del grupo de rutas con el prefijo auth dentro de **api.php**.

```
Route::group([
    'prefix' => 'auth'
], function () {
    Route::post('login', [AuthController::class, "login"]);
    Route::post('signup', [AuthController::class, "signup"]);

    Route::group([
        'middleware' => 'auth:api'
    ], function () {
        Route::get('logout', [AuthController::class, "logout"]);
        Route::get('user', [AuthController::class, "user"]);
    });

    // CREACION DEL MIDDLEWARE PARA ADMIN CON AUTENTICACION DEL USUARIO
    Route::middleware(['auth:api', 'role:admin'])->group(function () {

        // Rutas protegidas para usuarios con el rol "admin" y que este
autenticado.
        Route::apiResource('ficha', FichaController::class);
    });

});
```

Validar

Por ultimo podemos validar con Postman si la ruta protegida de 'ficha' este disponible solo para administradores.

Esto se hace creando un usuario con rol de 'admin' y logueandose para obtener el token de acceso, y ser usado en la ruta de 'ficha'. El resultado deberia ser que el usuario pueda hacer el crud de ficha.

También podemos probar que el acceso a 'ficha' este restringido para otros usuarios. Hacemos lo mismo que el paso anterior, solo que en lugar de escoger el rol de 'admin' escogemos 'aprendiz', nos logueamos, obtenemos el token, lo usamos en la ruta 'ficha' y nos deberia arrojar un error **403 Forbidden**.

Bibliografía

https://spatie.be/docs/laravel-permission/v6/introduction

https://styde.net/roles-y-permisos-con-spatie-laravel-permission/