Documentation : Mise en place d’un Guard dans Laravel avec Sanctum

Contexte :

Laravel est livré par défaut avec un système d’authentification basé sur un modèle User. Ce système suffit dans la plupart des cas lorsqu’il n’existe qu’un **type unique d’utilisateur** dans l’application.

Cependant, dans certains projets, il devient nécessaire de gérer **plusieurs types d’utilisateurs** disposant de règles d’accès ou d’espaces différents , par exemple :

- des **clients** qui utilisent l’application côté front,
- et des **administrateurs** qui gèrent la plateforme depuis un espace d’administration sécurisé.

Dans ce genre de contexte, créer un seul modèle User pour tout le monde devient vite limitant. C’est là qu’intervient la notion de **Guard**.

Un **Guard** dans Laravel définit **la façon dont un utilisateur est authentifié pour chaque requête**. Autrement dit, il indique :

- **quel “provider” utiliser** (c’est-à-dire depuis quelle table / modèle on charge les utilisateurs),
- **et quel mécanisme d’authentification** employer (session, token, Sanctum, Passport, etc.).

Etape 1 : Création du modèle et de la table

Dans cette documentation, nous allons supposer que nous travaillons avec un modèle Admin.

Commande à taper dans votre terminal : php artisan make:model Admin -m

Dans votre fichier de migration create_admins_tables,php

```php
public function up(): void
{
    Schema::create('admins', function (Blueprint $table) {
        $table->id();
        $table->string('name');
        $table->string('email')->unique();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}
```

Exécuter ensuite la migration avec cette commande : **php artisan migrate**

Etape 2 : Mettre à jour votre modèle

```php
php

<?php

namespace App\Models;

use Illuminate\Foundation\Auth\User as Authenticatable;
use Laravel\Sanctum\HasApiTokens;
use Illuminate\Notifications\Notifiable;

class Admin extends Authenticatable
{
    use HasApiTokens, Notifiable;

    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];
}
```

Etape 3 : Configuration du guard dans config/auth.php

```php
'providers' => [
    'users' => [
        'driver' => 'eloquent',
        'model' => App\Models\User::class,
    ],

    'admins' => [
        'driver' => 'eloquent',
        'model' => App\Models\Admin::class,
    ],
],
```

```php
'guards' => [
    'web' => [
        'driver' => 'session',
        'provider' => 'users',
    ],

    'api' => [
        'driver' => 'sanctum',
        'provider' => 'users',
    ],

    'admin' => [
        'driver' => 'sanctum',
        'provider' => 'admins',
    ],
],
```

Etape 4 : Création de votre contrôleur d’authentification Admin

À cette étape, chacun est libre d’implémenter sa logique selon l’architecture de son projet (controllers dédiés, services, ou classes internes).

Etape 5 : Configuration des routes Admin

Selon les besoins du projet, plusieurs approches sont possibles.

```php
<?php

use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;
use Perxo\v1\Admin\Http\Controllers\Api\AdminController;

Route::prefix('api')->group(function () {

    // Authentification basique
    Route::post('login', [AdminController::class, 'login']);

    // Routes protégées par Sanctum
    Route::middleware('auth:sanctum')->group(function () {
        Route::post('/logout', [AdminController::class, 'logout']);
        Route::post('change-password', [AdminController::class, 'changePassword']);
    });

    // Routes réservées aux administrateurs authentifiés
    Route::middleware('auth:admin')->group(function () {
        Route::post('store/admin', [AdminController::class, 'store']);
        Route::get('index/admins', [AdminController::class, 'index']);
        Route::get('show/admin/{id}', [AdminController::class, 'show']);
        Route::put('update/admin/{id}', [AdminController::class, 'update']);
        Route::delete('destroy/admin/{id}', [AdminController::class, 'destroy']);
    });
});
```

Le choix entre auth:santum et auth:admin dépend du contexte d’authentification.