

Modul Authentication Laravel

1.1. Tujuan

Setelah mengikuti modul ini mahasiswa diharapkan dapat:

1. Memahami konsep autentikasi dan otorisasi dalam pengembangan REST API.
2. Mengimplementasikan sistem autentikasi berbasis token menggunakan Laravel Sanctum.
3. Mengamankan endpoint API berdasarkan peran (role) atau kepemilikan data (ownership).
4. Menguji endpoint menggunakan postman.

1.2. Alat dan Bahan

1. XAMPP / Laravel Sail / Laragon / Valet
2. Composer
3. Laravel 12
4. Postman
5. MySQL / MariaDB
6. Laravel Sanctum

1.3. Tugas Pendahuluan

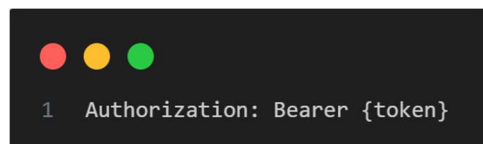
1. Apa perbedaan antara autentikasi dan otorisasi dalam konteks REST API?
2. Apa keuntungan menggunakan Laravel Sanctum dibandingkan metode autentikasi lainnya di Laravel?
3. Jelaskan apa yang dimaksud dengan Policy di Laravel dan dalam kasus apa kita menggunakannya.
4. Mengapa perlu membatasi akses API berdasarkan peran atau kepemilikan data?

1.4. Dasar Teori

1.4.1. Autentikasi (Authentication)

Autentikasi adalah proses memverifikasi identitas pengguna sebelum memberikan akses ke sistem. Dalam konteks Laravel REST API, autentikasi sering dilakukan menggunakan token untuk memastikan setiap request berasal dari pengguna yang sah. Laravel Sanctum menyediakan sistem autentikasi berbasis token ringan dan aman, ideal untuk SPA (Single Page Application) dan Mobile Application.

Sanctum memungkinkan pengguna login dan menerima token yang harus dikirim dalam header setiap permintaan API berikutnya.



1.4.2. Otorisasi (Authorization)

Otorisasi adalah proses menentukan apakah pengguna yang telah terautentikasi diizinkan melakukan aksi tertentu pada resource tertentu. Misalnya, hanya pemilik data yang bisa mengedit atau menghapus datanya, atau hanya admin yang boleh mengakses data pengguna lain. Di Laravel, otorisasi dapat diatur menggunakan Policy atau Gate, dengan Policy lebih direkomendasikan untuk REST API berbasis resource.

1.4.3. Laravel Sanctum

Laravel Sanctum adalah package resmi Laravel yang menyediakan sistem autentikasi token sederhana. Sanctum sangat cocok untuk REST API karena fleksibel dan mudah dikonfigurasi. Setiap pengguna yang berhasil login akan diberikan token yang dapat digunakan untuk mengakses endpoint yang dilindungi.

Sanctum bekerja dengan cara menyimpan token di database (personal_access_tokens) dan memverifikasi token yang dikirim melalui header Authorization.

1.4.4. Policy

Policy adalah class di Laravel yang berisi aturan otorisasi untuk model tertentu. Setiap aksi seperti view, create, update, dan delete bisa dibatasi dengan kondisi tertentu, seperti apakah user adalah pemilik data atau memiliki role tertentu.

Policy dihubungkan dengan model tertentu dan dipanggil menggunakan `$this->authorize('aksi', $model)` di dalam controller.

```
1 $this->authorize('update', $post);
```

1.5. Jurnal

Dalam modul ini, mahasiswa akan mengembangkan API berbasis Laravel Sanctum untuk manajemen postingan. Fitur yang akan dibangun meliputi:

- Register, Login, Logout menggunakan Laravel Sanctum
- Melihat daftar postingan
- Menambahkan, mengubah, dan menghapus postingan
- Hanya user pemilik postingan yang dapat mengubah postingannya
- Admin dapat menghapus semua postingan

1.5.1. Clone Project Github

Pertama-tama clone project yang telah tersedia di github, kemudian masuk ke proyek.

```
git clone https://github.com/cherryjayhawk/modul-api-auth.git
cd modul-api-auth
code .
```

1.5.2. Install Dependency

Instalasi dependency untuk Laravel dengan composer

```
composer install
```

1.5.3. Konfigurasi file environment variable

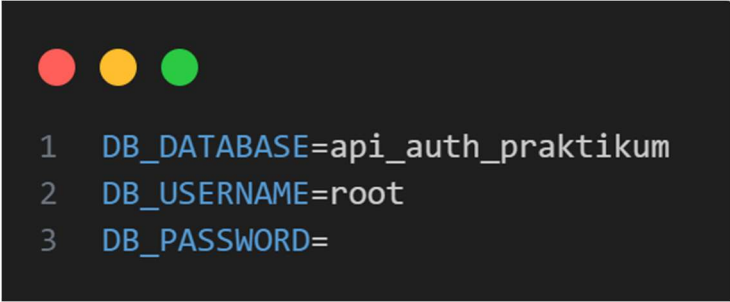
Copy .env.example dengan file baru bernama .env.

```
cp .env.example .env
```

Kemudian generate key APP_KEY di file .env

```
php artisan key:generate
```

Selanjutnya ubah konfigurasi database seperti berikut:



```
1 DB_DATABASE=api_auth_praktikum
2 DB_USERNAME=root
3 DB_PASSWORD=
```

1.5.4. Buat database api_auth_praktikum di MySQL/phpMyAdmin

1.5.5. Menginstall Laravel Sanctum dan melakukan setup API

```
php artisan install:api
```

Perintah ini akan:

- Install Sanctum.
- Mengatur API routes dan middleware.
- Publish konfigurasi Sanctum.

1.5.6. Migrasi dan seeding database

Lakukan migrate untuk membuat tabel yang dibutuhkan oleh Sanctum:

```
php artisan migrate
```

Selanjutnya seeding untuk mendaftarkan admin

```
php artisan db:seed
```

**(untuk keamanan dan best practice, sebaiknya akun admin dibuat langsung melalui seeder atau manual lewat database, bukan melalui endpoint publik seperti register)*

1.5.7. Memperbarui model User

Import HasApiTokens:

```
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Foundation\Auth\User as Authenticatable;
7  use Illuminate\Notifications\Notifiable;
8  use Laravel\Sanctum\HasApiTokens;
```

Gunakan HasApiTokens sebagai Traits di User model, dan update fillable dengan kolom role:

```
1  use HasFactory, Notifiable, HasApiTokens;
2
3  protected $fillable = [
4      'name',
5      'email',
6      'password',
7      'role'
8  ];
```

1.5.8. Membuat API AuthController

Buat AuthController dengan perintah:

```
php artisan make:controller Api/AuthController
```

Import package yang diperlukan:

```
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use App\Http\Controllers\Controller;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Hash;
8  use Illuminate\Support\Facades\Auth;
9  use App\Models\User;
10 use Illuminate\Validation\ValidationException;
```

Buat method register:

```
1 public function register(Request $request)
2 {
3     try {
4         $request->validate([
5             'name' => 'required|string|max:255',
6             'email' => 'required|email|unique:users',
7             'password' => 'required|confirmed|min:6',
8         ]);
9     } catch (ValidationException $e) {
10         return response()->json([
11             'success' => false,
12             'message' => 'Input is not valid',
13             'errors' => $e->errors(),
14         ], 422);
15     }
16
17     $user = User::create([
18         'name' => $request->name,
19         'email' => $request->email,
20         'password' => Hash::make($request->password),
21         'role' => 'user',
22     ]);
23
24     return response()->json([
25         'message' => 'Registration success',
26     ], 201);
27 }
```

Buat method login:

```
1 public function login(Request $request)
2 {
3     try {
4         $credentials = $request->validate([
5             'email' => 'required|email',
6             'password' => 'required',
7         ]);
8     } catch (ValidationException $e) {
9         return response()->json([
10             'success' => false,
11             'message' => 'Input is not valid',
12             'errors' => $e->errors(),
13         ], 422);
14     }
15
16     if (Auth::attempt($credentials)) {
17         $user = Auth::user();
18         $token = $user->createToken('api_token')->plainTextToken;
19
20         return response()->json([
21             'token' => $token,
22             'message' => 'Login success',
23         ], 200);
24     }
25
26     return response()->json(['message' => 'Invalid credentials'], 401);
27 }
```

Buat method logout:

```
1 public function logout(Request $request)
2 {
3     $user = $request->user();
4
5     $user->currentAccessToken()->delete();
6
7     return response()->json(['message' => 'Logged out. Token deleted.'], 200);
8 }
```

1.5.9. Memperbarui api routes

Isi **api.php** di folder **routes**:

```
1 use App\Http\Controllers\Api\AuthController;
2
3 Route::post('/register', [AuthController::class, 'register']);
4 Route::post('/login', [AuthController::class, 'login']);
5 Route::middleware('auth:sanctum')->post('/logout', [AuthController::class, 'logout']);
```

1.5.10. Testing API dengan Postman

Buat Collection baru dengan nama: **PRAKTIKUM_AUTH**

Kemudian **uji endpoint** seperti di bawah ini.

Screenshot hasil setiap request.

a. Register

- Method : POST
- URL : /api/register
- Body (x-www-form-urlencoded):
 - name: nama praktikan
 - email: email praktikan
 - password: 123456
 - password_confirmation: 123456

b. Login

- Method : POST
- URL : /api/login
- Body (x-www-form-urlencoded):
 - isikan email dan password yang digunakan untuk mendaftar
- **Copy token dari response/hasil request. Misal:**
1|ZjAe8tJFJC5MmZ0584cGGBkhV6mplaCdNFibJqnyf957efc5

c. Logout

- Method : POST
- URL : /api/logout
- Authorization:
 - Type: Bearer Token
 - Isi Token yang telah di copy dari response saat login

1.6. Tugas Akhir

1.6.1. Buat controller API PostController

```
php artisan make:controller Api/PostController --api
```

Kemudian isi:

```
1  <?php
2
3  namespace App\Http\Controllers\Api;
4
5  use Illuminate\Routing\Controller;
6  use Illuminate\Http\Request;
7  use App\Models\Post;
8
9  class PostController extends Controller
10 {
11     public function index()
12     {
13         return Post::with('user')->latest()->get();
14     }
15
16     public function store(Request $request)
17     {
18         $validated = $request->validate([
19             'title' => 'required',
20             'content' => 'required',
21         ]);
22
23         $post = $request->user()->posts()->create($validated);
24
25         return response()->json($post, 201);
26     }
27
28     public function show(Post $post)
29     {
30         return $post;
31     }
32
33     public function update(Request $request, Post $post)
34     {
35         $user = $request->user();
36
37         // Authorization sederhana
38         if ($user->id !== $post->user_id && $user->role !== 'admin') {
39             return response()->json(['message' => 'Unauthorized'], 403);
40         }
41
42         $post->update($request->only(['title', 'content']));
43         return $post;
44     }
45
46     public function destroy(Request $request, Post $post)
47     {
48         $user = $request->user();
49
50         if ($user->id !== $post->user_id && $user->role !== 'admin') {
51             return response()->json(['message' => 'Unauthorized'], 403);
52         }
53
54         $post->delete();
55         return response()->json(['message' => 'Deleted']);
56     }
57 }
58
```


1.6.2. Menambahkan resource Post dengan middleware pada API routes

```
1 Route::middleware('auth:sanctum')->group(function () {  
2     Route::apiResource('posts', PostController::class);  
3 });
```

1.6.3. Tes Menggunakan Postman

Buat Collection baru dengan nama: **PRAKTIKUM_AUTH**

Kemudian **uji endpoint** seperti di bawah ini.

Screenshot hasil setiap request.

a. Mengakses endpoint yang diproteksi oleh sanctum

- Method : GET
- URL : /api/posts

(User yang tidak login atau tidak memiliki token tidak bisa melihat post)

b. Login sebagai user biasa

- Method : POST
- URL : /api/login
- Body (x-www-form-urlencoded):
 - isikan email dan password yang digunakan untuk mendaftar
- **Copy token dari response/hasil request. Misal:**
1|ZjAe8tJFJC5MmZ0584cGGBkhV6mplaCdNFibJqnyf957efc5

c. Buat post baru (user dapat membuat post)

- Method : POST
- URL : /api/posts
- Authorization:
 - Type: Bearer Token
 - Isi Token yang telah di copy dari response saat login user
- Body (x-www-form-urlencoded):
 - title: "Post Authorization"
 - content: "Authorization sederhana"

d. Logout

- Method : POST
- URL : /api/logout
- Authorization:
 - Type: Bearer Token
 - Isi Token yang telah di copy dari response saat login

e. Register user lain

- Method : POST
- URL : /api/register
- Body (x-www-form-urlencoded):
 - name: new user
 - email: user@example.com
 - password: 123456
 - password_confirmation: 123456

f. Login sebagai user baru

- Method : POST
- URL : /api/login
- Body (x-www-form-urlencoded):
 - email: user@example.com
 - password: 123456
- **Copy token dari response/hasil request. Misal:**
1|ZjAe8tJFJC5MmZ0584cGGBkhV6mplaCdNFibJqnyf957efc5

g. User baru mengupdate post user lain

- Method : PUT
- URL : /api/posts/1
- Authorization:
 - Type: Bearer Token
 - Isi Token yang telah di copy dari response saat login user baru
- Body (x-www-form-urlencoded):
 - title: "Saya mau ubah post orang"
 - Content: "Coba ubah post"

(User tidak berhak mengubah post orang lain.)

h. User baru menghapus post user lain

- Method : DELETE
- URL : /api/posts/1
- Authorization:
 - Type: Bearer Token
 - Isi Token yang telah di copy dari response saat login user baru

(User tidak berhak menghapus post orang lain.)

i. Logout

- Method : POST
- URL : /api/logout
- Authorization:
 - Type: Bearer Token
 - Isi Token yang telah di copy dari response saat login

j. Login admin

- Method : POST
- URL : /api/login
- Body (x-www-form-urlencoded):
 - email: admin@example.com
 - password: admin123
- **Copy token dari response/hasil request. Misal:**
1|ZjAe8tJFJC5MmZ0584cGGBkhV6mplaCdNFibJqnyf957efc5

k. Admin menghapus post

- Method : DELETE
- URL : /api/posts/1
- Authorization:
 - Type: Bearer Token
 - Isi Token yang telah di copy dari response saat login admin

(Admin berhak menghapus post.)