

# Laporan Modul 8: Authentication & Authorization

---

**Mata Kuliah:** Workshop Web Lanjut

**Nama:** M. Iqbal Sayuti

**NIM:** 2024573010057

**Kelas:** TI-2C

---

## Abstrak

Pada praktikum ini dipelajari penerapan Authentication dan Authorization dalam pengembangan aplikasi Laravel untuk memastikan keamanan akses pengguna. Authentication berfungsi untuk memverifikasi identitas pengguna melalui proses login, registrasi, dan manajemen sesi menggunakan fitur bawaan Laravel seperti Laravel Breeze atau Fortify. Sementara itu, Authorization digunakan untuk mengatur hak akses berdasarkan peran atau izin tertentu melalui mekanisme gates dan policies. Dengan mengimplementasikan kedua konsep tersebut, aplikasi menjadi lebih aman, terstruktur, dan mampu membatasi tindakan pengguna sesuai otorisasi yang telah ditentukan. Praktikum ini memberikan pemahaman dasar mengenai bagaimana mengelola identitas dan kontrol akses sehingga aplikasi dapat berjalan dengan aman dan sesuai standar pengembangan modern.

---

## 1. Dasar Teori

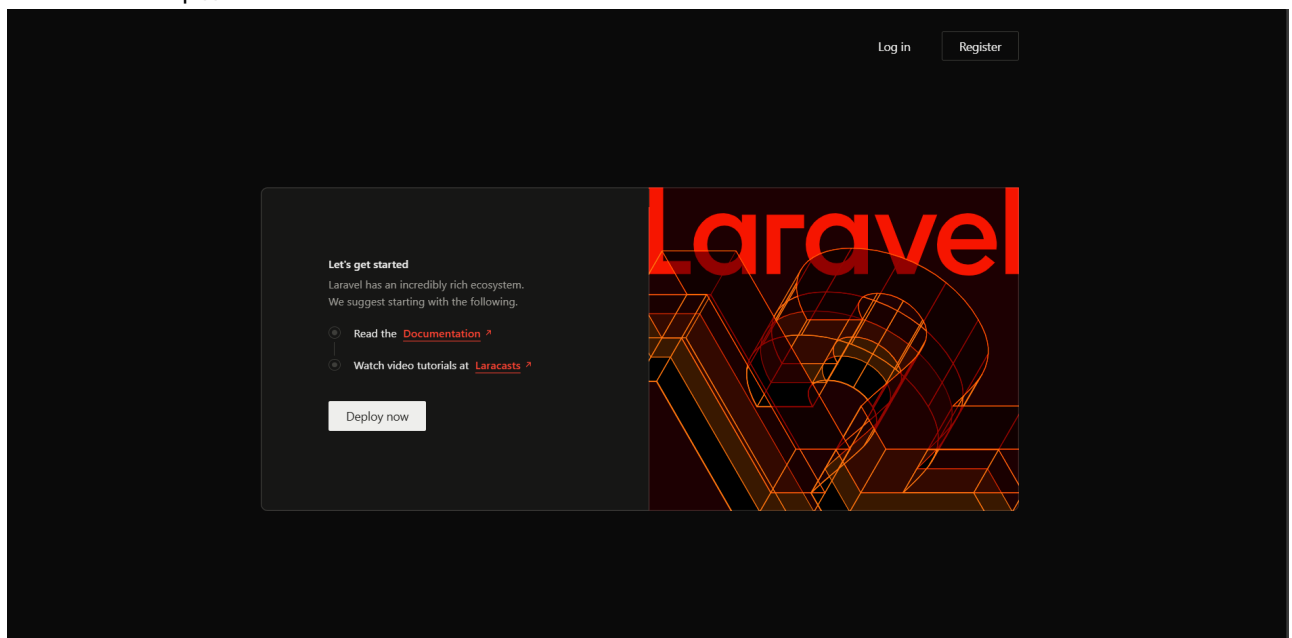
Dalam pengembangan aplikasi berbasis Laravel, Authentication dan Authorization merupakan dua mekanisme penting untuk memastikan keamanan akses pengguna, mulai dari proses verifikasi identitas hingga pemberian hak akses sesuai perannya.

1. Authentication = Proses untuk memverifikasi identitas pengguna sebelum mengakses fitur aplikasi, biasanya melalui login, registrasi, dan manajemen sesi.
  2. Laravel Auth System = Menyediakan sistem autentikasi bawaan melalui Breeze, Jetstream, atau Fortify yang menangani proses login, logout, registrasi, dan reset password secara otomatis.
  3. Password Hashing = Laravel menggunakan algoritma bcrypt atau argon2 untuk menyimpan password secara aman di database.
  4. Middleware auth = Digunakan untuk melindungi rute sehingga hanya pengguna yang sudah terverifikasi yang dapat mengakses halaman tertentu.
  5. Session & Token = Setelah login, Laravel membangun sesi atau token untuk mempertahankan status autentikasi pengguna.
  6. Authorization = Mekanisme untuk menentukan apakah pengguna yang telah terautentikasi memiliki hak atau izin untuk melakukan suatu aksi dalam aplikasi.
  7. Roles & Permissions = Digunakan untuk membedakan tingkat akses pengguna, seperti admin, editor, atau user biasa.
  8. Gates = Mekanisme otorisasi sederhana berbasis closure yang digunakan untuk memeriksa izin sebelum suatu tindakan dilakukan.
  9. Policies = Kelas khusus yang mengelola aturan akses terhadap model tertentu, seperti izin untuk membuat, mengedit, atau menghapus data.
-

## 2. Langkah-Langkah Praktikum

### Praktikum 1: Autentikasi dan Otorisasi dengan Laravel 12 Breeze

- Langkah 1: Buat dan Buka Proyek laravel  
laravel new auth-lab  
cd auth-lab  
code .
- Langkah 2 : Buat database `Create database authlab_db;`  
Kemudian install dependency MySQL dan setelahnya ubah file `.env` sesuai nama database  
`php artisan config:clear`
- Langkah 3 : Instalasi laravel breeze  
`composer require laravel/breeze --dev`  
Kemudian, jalankan perintah berikut untuk menginstal breeze  
`php artisan breeze:install`  
Selama proses instalasi, anda akan diberikan beberapa opsi:
  - pilih frontend framework : blade
  - dark mode : pilih yesLanjutkan dengan langkah ini :  
`npm install`  
`php artisan migrate`
- Langkah 4 : Akses Register and Login via Web Interface Jalankan `php artisan serve` dan cek ke `http://localhost:8000`



- Langkah 5 : Membuat Rute Profil yang Dilindungi

```

routes > web.php
1  <?php
2
3  use App\Http\Controllers\ProfileController;
4  use Illuminate\Support\Facades\Route;
5  use Illuminate\Support\Facades\Auth;
6
7  Route::get('/', function () {
8      return view('welcome');
9  });
10
11 Route::get('/dashboard', function () {
12     return view('dashboard');
13 })->middleware(['auth', 'verified'])->name('dashboard');
14
15 Route::middleware('auth')->group(function () {
16     Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
17     Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
18     Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
19
20     // Tambahkan rute myprofile baru
21     Route::get('/myprofile', function () {
22         return Auth::user();
23     })->name('myprofile');
24 });
25
26 require __DIR__.'/auth.php';

```

Kemudian coba akses ke <http://localhost:8000/myprofile>

```

{"id":1,"name":"M. Iqbal Sayuti","email":"iqbalsayuti@gmail.com","email_verified_at":null,"created_at":"2025-11-19T01:11:48.000000Z","updated_at":"2025-11-19T01:11:48.000000Z"}

```

## Praktikum 2: Membatasi Akses Berdasarkan Peran di Laravel 12

- Langkah 1: Buat dan Buka Proyek laravel

laravel new role-lab

cd role-lab

code .

- Langkah 2: Konfigurasi dan buat database Buat database bernama authrole\_db dan isikan ke file .env  
php artisan config:clear

- Langkah 3 : Instalasi Laravel Breeze composer require laravel/breeze --dev

Kemudian jalankan php artisan breeze:install

srpti tadi, jika ada opsi yg muncul maka kita akan memilih framework blade dan dark mode

Jalankan perintah :

npm install

php artisan migrate

- Langkah 4 : Menambahkan Field Role ke Tabel Users

Buat migration php artisan make:migration add\_role\_to\_users\_table --table=users

buka file yg migrasi yg telah dibuat dan edit :

```
database > migrations > 2025_11_11_072753_add_role_to_users_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::table('users', function (Blueprint $table) {
15             $table->string('role')->default('user');
16         });
17     }
18
19     /**
20      * Reverse the migrations.
21      */
22     public function down(): void
23     {
24         Schema::table('users', function (Blueprint $table) {
25             $table->dropColumn('role');
26         });
27     }
28 };
29
```

dan jalankan `php artisan migrate`

- Langkah 5 : Seeding Pengguna dengan Peran Berbeda  
buka dan Edit database/seeders/DatabaseSeeder.php

```

database > seeders > DatabaseSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use App\Models\User;
6  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
7  use Illuminate\Database\Seeder;
8  use Illuminate\Support\Facades\Hash;
9
10 class DatabaseSeeder extends Seeder
11 {
12     use WithoutModelEvents;
13
14     /**
15      * Seed the application's database.
16      */
17     public function run(): void
18     {
19         // User::factory(10)->create();
20
21         User::factory()->create([
22             'name' => 'Test User',
23             'email' => 'test@example.com',
24         ]);
25
26         User::create([
27             'name' => 'Admin User',
28             'email' => 'admin@ilmudata.id',
29             'password' => Hash::make('password123'),
30             'role' => 'admin',
31         ]);
32
33         User::create([
34             'name' => 'Manager User',
35             'email' => 'manager@ilmudata.id',
36             'password' => Hash::make('password123'),
37             'role' => 'manager',
38         ]);
39
40         User::create([
41             'name' => 'General User',
42             'email' => 'user@ilmudata.id',
43             'password' => Hash::make('password123'),
44             'role' => 'user',
45         ]);
46     }
47 }
48

```

Jalankan `php artisan db:seed`

- Langkah 6 : Membuat Role Middleware

Generate middleware : `php artisan make:middleware RoleMiddleware`

Kemudian edit :

```

app > Http > Middleware > RoleMiddleware.php
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Closure;
6  use Illuminate\Http\Request;
7  use Symfony\Component\HttpFoundation\Response;
8
9  class RoleMiddleware
10 {
11     public function handle(Request $request, Closure $next, string $role): Response
12     {
13         if ($request->user() && $request->user()->role === $role) {
14             return $next($request);
15         }
16
17         abort(403, 'Unauthorized');
18     }
19 }

```

Kemudian buka `bootstrap/app.php` dan edit :

```
bootstrap > app.php
1  <?php
2
3  use Illuminate\Foundation\Application;
4  use Illuminate\Foundation\Configuration\Exceptions;
5  use Illuminate\Foundation\Configuration\Middleware;
6
7  use App\Http\Middleware\RoleMiddleware;
8
9  return Application::configure(basePath: dirname(__DIR__))
10     ->withRouting(
11         web: __DIR__.'/../routes/web.php',
12         commands: __DIR__.'/../routes/console.php',
13         health: '/up',
14     )
15     ->withMiddleware(function (Middleware $middleware): void {
16         $middleware->alias([
17             'role' => RoleMiddleware::class,
18         ]);
19     })
20     ->withExceptions(function (Exceptions $exceptions): void {
21         //
22     }->create();
```

- Langkah 7 : Membuat View untuk Setiap Role
  - Buat file `admin.blade.php` pada `resources/views/` dan edit :

```
resources > views > admin.blade.php
1  <x-app-layout>
2      <x-slot name="header">
3          <h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">
4              {{ __('Admin Dashboard') }}
5          </h2>
6      </x-slot>
7
8      <div class="py-12">
9          <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
10             <div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg">
11                 <div class="p-6 text-gray-900 dark:text-gray-100">
12                     {{ __("Welcome, Admin! You have full access.") }}
13                 </div>
14             </div>
15         </div>
16     </div>
17 </x-app-layout>
```

- Buat file **Manager.blade.php** pada **resources/views/** dan edit :

```
resources > views > manager.blade.php
1  <x-app-layout>
2      <x-slot name="header">
3          <h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">
4              {{ __('Manager Dashboard') }}
5          </h2>
6      </x-slot>
7
8      <div class="py-12">
9          <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
10             <div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg">
11                 <div class="p-6 text-gray-900 dark:text-gray-100">
12                     {{ __("Welcome, Manager! You can manage and monitor resources.") }}
13                 </div>
14             </div>
15         </div>
16     </div>
17 </x-app-layout>
```

- Buat file **User.blade.php** pada **resources/views/** dan edit :

```
resources > views > user.blade.php
1  <x-app-layout>
2      <x-slot name="header">
3          <h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">
4              {{ __('User Dashboard') }}
5          </h2>
6      </x-slot>
7
8      <div class="py-12">
9          <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
10             <div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg">
11                 <div class="p-6 text-gray-900 dark:text-gray-100">
12                     {{ __("Welcome, User! You have limited access.") }}
13                 </div>
14             </div>
15         </div>
16     </div>
17 </x-app-layout>
```

- Buat file **all.blade.php** pada **resources/views/** dan edit :

```
resources > views > all.blade.php
1  <x-app-layout>
2      <x-slot name="header">
3          <h2 class="font-semibold text-xl text-gray-800 dark:text-gray-200 leading-tight">
4              {{ __('General Dashboard') }}
5          </h2>
6      </x-slot>
7
8      <div class="py-12">
9          <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
10             <div class="bg-white dark:bg-gray-800 overflow-hidden shadow-sm sm:rounded-lg">
11                 <div class="p-6 text-gray-900 dark:text-gray-100">
12                     {{ __("Welcome! This view is accessible by all authenticated roles.") }}
13                 </div>
14             </div>
15         </div>
16     </div>
17 </x-app-layout>
```

- Langkah 8 : Mendefinisikan Rute untuk View Berbasis Peran Buka routes/web.php dan tambahkan:

```
routes > web.php
1  <?php
2
3  use App\Http\Controllers\ProfileController;
4  use Illuminate\Support\Facades\Route;
5
6  Route::get('/', function () {
7      return view('welcome');
8  });
9
10 Route::get('/dashboard', function () {
11     return view('dashboard');
12 })->middleware(['auth', 'verified'])->name('dashboard');
13
14 Route::middleware('auth')->group(function () {
15     Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
16     Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
17     Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
18 });
19
20 Route::middleware('auth')->group(function () {
21     // Rute yang dapat diakses oleh semua pengguna terautentikasi
22     Route::get('/all', function () {
23         return view('all');
24     });
25
26     // Rute khusus admin dengan middleware role
27     Route::get('/admin', function () {
28         return view('admin');
29     })->middleware('role:admin');
30
31     // Rute khusus manager dengan middleware role
32     Route::get('/manager', function () {
33         return view('manager');
34     })->middleware('role:manager');
35
36     // Rute khusus user dengan middleware role
37     Route::get('/user', function () {
38         return view('user');
39     })->middleware('role:user');
40
41     require __DIR__.'/auth.php';
42 }
```

- Langkah 9 : Jalankan Aplikasi

Jalankan `php php artisan serve`

Akses aplikasi di:

`http://127.0.0.1:8000`

Coba login menggunakan data akun :

- Admin: admin@ilmudata.id / password123
- Manager: manager@ilmudata.id / password123
- User: user@ilmudata.id / password123

---

## 3. Hasil dan Pembahasan

Pada praktikum 8 ini, Anda telah berhasil:

- Mengkonfigurasi sistem Authentication menggunakan Laravel Breeze untuk menyediakan fitur login, registrasi, dan logout secara otomatis.
- Mengamankan halaman tertentu menggunakan middleware auth sehingga hanya pengguna yang sudah login dapat mengakses fitur tertentu.
- Mengimplementasikan Authorization melalui gates dan policies untuk membatasi tindakan pengguna berdasarkan peran atau izin yang telah ditentukan.
- Membuat role atau level akses dasar (misalnya admin dan user) untuk membedakan hak akses pada beberapa fitur aplikasi.
- Menggunakan middleware can untuk memastikan hanya pengguna yang memiliki izin tertentu yang dapat mengakses rute atau melakukan aksi tertentu.
- Menguji hasil autentikasi dan otorisasi dengan mencoba login sebagai berbagai tipe pengguna dan memastikan sistem membatasi akses sesuai aturan.



- Menampilkan feedback kepada pengguna ketika akses ditolak, misalnya melalui pesan error atau redirect ke halaman khusus.

Hasil praktikum menunjukkan bahwa penerapan Authentication dan Authorization pada Laravel secara signifikan meningkatkan keamanan aplikasi dengan memastikan hanya pengguna terverifikasi yang dapat masuk, dan hanya pengguna berizin yang dapat melakukan tindakan tertentu. Integrasi antara autentikasi dan otorisasi membuat sistem menjadi lebih terkontrol, rapi, serta sesuai standar keamanan pengembangan web modern.

---

## 4. Kesimpulan

Pada praktikum ini dapat disimpulkan bahwa penerapan Authentication dan Authorization sangat penting untuk menjaga keamanan aplikasi Laravel. Authentication memastikan bahwa hanya pengguna yang terverifikasi yang dapat mengakses sistem, sedangkan Authorization menentukan hak akses setiap pengguna berdasarkan peran atau izin yang telah ditetapkan. Dengan memanfaatkan fitur Laravel seperti Breeze, middleware auth, gates, policies, serta middleware can, proses pengamanan aplikasi dapat dilakukan dengan mudah, terstruktur, dan efisien. Secara keseluruhan, kedua mekanisme ini memungkinkan aplikasi berjalan lebih aman, terkontrol, dan sesuai standar keamanan dalam pengembangan web modern.

---

## 5. Referensi

- chatgpt.com
  - <https://hackmd.io/@mohdrzu/BypBawklWg>
-