

Bab 10 | Laravel Authentication, Authorization, and Middleware

10.1 Pengantar

Di Laravel, autentikasi, otorisasi, dan middleware adalah konsep utama dalam pengembangan web, terutama dalam konteks membangun aplikasi akses yang aman dan terkontrol. Dengan memahami dan menerapkan konsep-konsep ini, pengembang dapat menciptakan aplikasi web yang tidak hanya fungsional, tetapi juga aman dari ancaman dan dapat mengontrol akses pengguna dengan baik.

10.1.1 Autentikasi (Authentication)

Autentikasi adalah proses verifikasi identitas pengguna sebelum mereka diizinkan mengakses suatu aplikasi atau layanan. Dalam konteks web, autentikasi sering kali melibatkan proses login, di mana pengguna memasukkan kredensial seperti email dan kata sandi yang kemudian diverifikasi oleh sistem. Laravel menyediakan berbagai alat dan fitur untuk mempermudah proses autentikasi, termasuk sistem login bawaan, registrasi, dan reset kata sandi. Dengan memanfaatkan Laravel Authentication, pengembang dapat dengan mudah mengelola otentikasi pengguna secara aman dan efisien.

10.1.2 Otorisasi (Authorization)

Otorisasi adalah proses menentukan apakah pengguna yang telah terautentikasi memiliki izin untuk mengakses sumber daya tertentu atau melakukan tindakan tertentu dalam aplikasi. Sementara autentikasi menentukan siapa pengguna itu, otorisasi menentukan apa yang dapat dilakukan pengguna tersebut. Laravel menyediakan mekanisme otorisasi yang kuat melalui kebijakan (policies) dan otorisasi berbasis peran (role-based access control). Dengan menggunakan kebijakan, pengembang dapat mendefinisikan aturan yang menentukan akses ke berbagai bagian aplikasi.

10.1.3 Middleware (Middleware)

Middleware adalah lapisan logika yang berjalan di antara permintaan HTTP yang masuk dan respons HTTP yang keluar dalam aplikasi Laravel. Middleware digunakan untuk memeriksa dan mengolah permintaan sebelum diteruskan ke rute atau controller. Dalam konteks autentikasi dan otorisasi, middleware seperti auth digunakan untuk memastikan bahwa hanya

pengguna yang telah diautentikasi yang dapat mengakses rute tertentu. Middleware sangat penting dalam menjaga keamanan dan kelancaran alur kerja aplikasi web.

10.2 Ekosistem Laravel

Laravel dikenal sebagai framework PHP yang sangat kaya akan fitur dan ekosistem yang komprehensif. Ekosistem Laravel mencakup berbagai alat dan paket yang mempermudah pengembangan aplikasi, mulai dari sistem autentikasi, routing, hingga pengelolaan database. Laravel memberikan kemudahan bagi pengembang dengan menyediakan berbagai starter kit dan alat tambahan yang dapat digunakan untuk mempercepat pengembangan.

10.2.1 Pengenalan Breeze sebagai Salah Satu Ekosistem Laravel

Breeze adalah salah satu starter kit yang disediakan oleh Laravel untuk memudahkan pengembangan fitur autentikasi dasar dalam aplikasi. Breeze menyediakan implementasi yang sederhana dan mudah dipahami untuk autentikasi, termasuk login, registrasi, reset kata sandi, dan verifikasi email. Breeze sangat cocok digunakan sebagai fondasi bagi pengembang yang ingin membangun fitur autentikasi dengan cepat tanpa harus mengorbankan fleksibilitas.

10.2.2 Fitur-Fitur Breeze dalam Ekosistem Laravel

Breeze hadir dengan front-end yang minimalis berbasis Tailwind CSS, sehingga memudahkan pengembang untuk melakukan kustomisasi sesuai kebutuhan. Fitur-fitur utama dari Breeze meliputi:

- Autentikasi Pengguna
 - Breeze menyediakan semua rute, kontroler, dan tampilan yang dibutuhkan untuk mengelola autentikasi pengguna.
- Registrasi Pengguna
 - Pengguna baru dapat mendaftar dan membuat akun dengan mudah menggunakan formulir registrasi yang disediakan.
- Reset Kata Sandi
 - Breeze mendukung fitur reset kata sandi, memungkinkan pengguna untuk mengatur ulang kata sandi mereka jika lupa.

- Verifikasi Email
 - Fitur verifikasi email dapat diimplementasikan dengan mudah untuk meningkatkan keamanan akun pengguna.

10.3 Memulai Pengembangan Laravel dengan Breeze

Sebelum memulai pengembangan, pastikan PHP dan Node.js beserta `npm` sudah terinstall di perangkat masing-masing.

Silahkan buka XAMPP lalu nyalakan Apache dan MySQL. Lalu buka Command Prompt lalu ketikkan `php -version`.

```
C:\Users\ASUS>php -version
PHP 8.2.4 (cli) (built: Mar 14 2023 17:54:25) (ZTS Visual C++ 2019 x64)
Copyright (c) The PHP Group
Zend Engine v4.2.4, Copyright (c) Zend Technologies
```

Jika sudah muncul seperti di atas dan versi PHP menunjukkan di atas “8.2”, maka artinya sudah aman. Jika belum, silahkan download XAMPP pada link berikut <https://www.apachefriends.org/download.html>.

Setelah itu, ketikkan `node -v` dan `npm -v`

```
C:\Users\ASUS>node -v
v18.18.0

C:\Users\ASUS>npm -v
9.8.1
```

Jika sudah seperti di atas, maka artinya sudah aman. Jika belum muncul, silahkan download Node.js terlebih dahulu pada link berikut <https://nodejs.org/en/download/source-code> , pastikan memilih versi yang “LTS”.

10.3.1 Menginstall Laravel

Silahkan buka file explorer lalu masuk ke folder kosong, setelah itu buka Command Prompt lalu ketik sintaks berikut untuk membuat proyek Laravel.

```
composer create-project laravel/laravel laravel-auth
```

10.3.2 Menginstall Breeze pada Proyek Laravel

Setelah proyek Laravel baru dengan nama “laravel-auth” berhasil diinstall, berikutnya alihkan Command Prompt ke dalam folder tersebut dengan cara:

```
cd laravel-auth
```

Setelah Command Prompt sudah dialihkan ke dalam folder proyek Laravel baru, masukkan sintaks di bawah untuk mulai menginstall Breeze.

```
composer require laravel/breeze --dev
```

Setelah loading selesai, lanjutkan dengan memasukkan sintaks berikut:

```
php artisan breeze:install
```

Pada saat menjalankan sintaks di atas, akan terdapat beberapa pilihan dalam menginstall Breeze ke proyek Laravel seperti di bawah. Silahkan mengikuti urutan seperti pada gambar berikut dalam menginstall Breeze.

```
Which Breeze stack would you like to install?
Blade with Alpine ..... blade
Livewire (Volt Class API) with Alpine ..... livewire
Livewire (Volt Functional API) with Alpine ..... livewire-functional
React with Inertia ..... react
Vue with Inertia ..... vue
API only ..... api
> blade

Would you like dark mode support? (yes/no) [no]
> no

Which testing framework do you prefer? [PHPUnit]
Pest ..... 0
PHPUnit ..... 1
> 1

INFO: Installing and building Node dependencies.
```

Input pertama isi dengan “blade”. Lalu inputan selanjutnya silahkan memilih apakah ingin mengaktifkan dark mode dalam proyek atau tidak, sebagai contoh di atas, dipilih “no” atau tidak untuk dark mode. Yang terakhir, untuk inputan seputar testing, silahkan input angka “1”.

```
added 144 packages, and audited 145 packages in 14s
36 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities

> build
> vite build

  vite v5.4.2 building for production...
    transforming...
      ✓ 54 modules transformed.
    rendering chunks...
computing gzip size...
public/build/manifest.json      0.27 kB | gzip:  0.14 kB
public/build/assets/app-cNy9yCjt.css 33.92 kB | gzip:  6.52 kB
public/build/assets/app-DCrXoRMQ.js 79.15 kB | gzip: 29.48 kB
✓ built in 1.74s

[INFO] Breeze scaffolding installed successfully.
```

Perhatikan bahwa jika proses instalasi sangat lama, itu artinya jaringan sedang kurang baik. Jangan lupa pula memantau XAMPP, pastikan Apache dan MySQL tetap aktif.

Setelah muncul informasi seperti pada gambar di atas, kita bisa melanjutkan ke tahap berikutnya. Silahkan lanjutkan dengan menjalankan sintaks berikut.

```
npm install
```

Setelah proses instalasi selesai melalui ‘npm’, silahkan buka VS Code pada folder proyek Laravel yang baru dibuat ini dengan cara:

```
code .
```

10.3.3 Membuat Database MySQL dan Melakukan Konfigurasi di Proyek Laravel

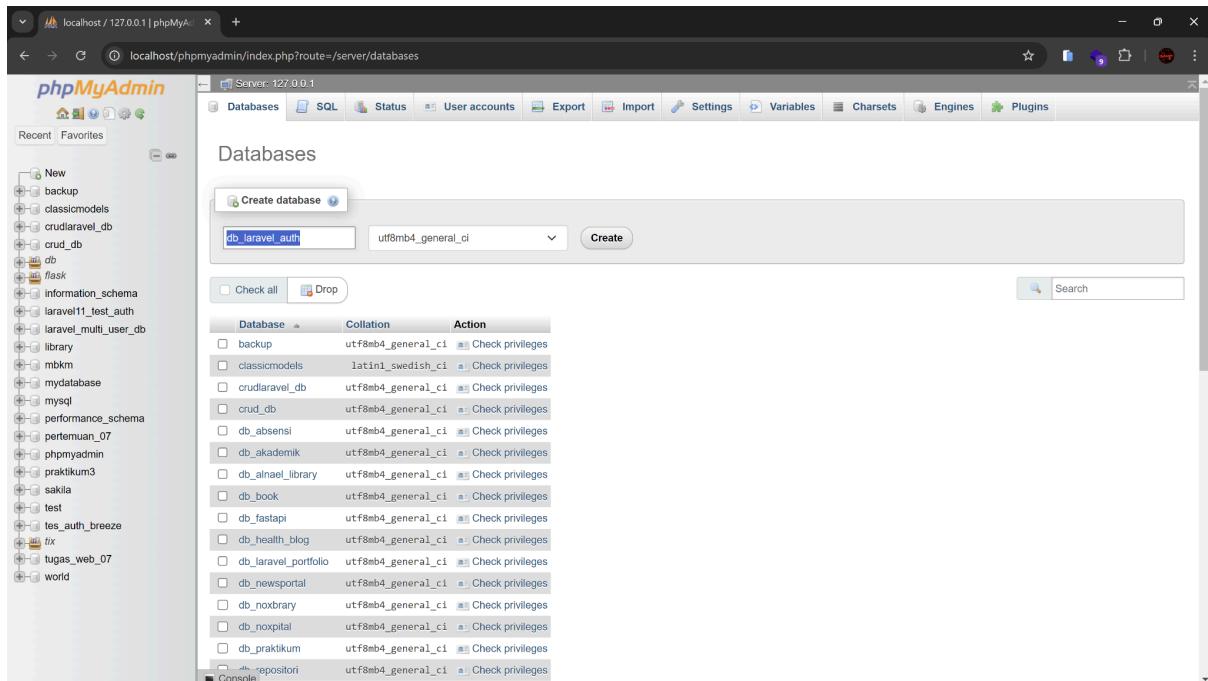
Pada proyek Laravel yang sudah terbuka di VS Code, silahkan masuk ke file “.env”

```
22  DB_CONNECTION=sqlite
23  # DB_HOST=127.0.0.1
24  # DB_PORT=3306
25  # DB_DATABASE=laravel
26  # DB_USERNAME=root
27  # DB_PASSWORD=
```

Silahkan ubah baris di atas menjadi seperti berikut:

```
22 DB_CONNECTION=mysql
23 DB_HOST=127.0.0.1
24 DB_PORT=3306
25 DB_DATABASE=db_laravel_auth
26 DB_USERNAME=root
27 DB_PASSWORD=
```

Setelah itu, buka browser lalu masukkan URL “localhost/phpmyadmin”, lalu silahkan buat database baru sesuai dengan nama database yang diinput di “.env” sebelumnya.



Setelah berhasil membuat database baru, kembali ke VS Code lalu buka file migrasi tabel user dengan nama “0001_01_01_000000_create_users_table.php” yang terletak di folder “database/migrations”. Tambahkan kolom “role” dengan nilai default “user” seperti baris 18 pada gambar berikut.

```

12     public function up(): void
13     {
14         Schema::create('users', function (Blueprint $table) {
15             $table->id();
16             $table->string('name');
17             $table->string('email')->unique();
18             $table->string('role')->default('user');
19             $table->timestamp('email_verified_at')->nullable();
20             $table->string('password');
21             $table->rememberToken();
22             $table->timestamps();
23         });

```

Setelah itu, buka Terminal pada VS Code dengan cara ‘Ctrl + J’. Silahkan masukkan perintah berikut untuk melakukan migrasi data yang dibuat di Laravel ke database “db_laravel_auth” di MySQL.

```
php artisan migrate
```

Setelah migrasi selesai tanpa error, buka kembali phpMyAdmin di browser pada database “db_laravel_auth”. Jika migrasi berhasil, maka database tersebut akan terisi dengan beberapa tabel seperti pada gambar di bawah.

Table	Action	Rows	Type	Collation	Size	Overhead
cache	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 Kib	-
job_batches	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
migrations	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
password_reset_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
sessions	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 Kib	-
users	Browse Structure Search Insert Empty Drop	3	InnoDB	utf8mb4_unicode_ci	32.0 Kib	-

9 tables Sum

Create new table

Table name: users Number of columns: 4 Create

Pastikan pada tabel user sudah terdapat kolom “role”

The screenshot shows the 'Structure' tab for the 'users' table in the 'db_laravel_auth' database. The table structure is as follows:

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	<code>id</code>	bigint(20)	utf8mb4_unicode_ci	UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	<code>name</code>	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
3	<code>email</code>	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
4	<code>role</code>	varchar(255)	utf8mb4_unicode_ci		No	user			Change Drop More
5	<code>email_verified_at</code>	timestamp			Yes	NULL			Change Drop More
6	<code>password</code>	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
7	<code>remember_token</code>	varchar(100)	utf8mb4_unicode_ci		Yes	NULL			Change Drop More
8	<code>created_at</code>	timestamp			Yes	NULL			Change Drop More
9	<code>updated_at</code>	timestamp			Yes	NULL			Change Drop More

Below the table structure, there are buttons for 'Check all', 'With selected:', and various actions like 'Browse', 'Change', 'Drop', 'Primary', 'Unique', 'Index', 'Spatial', and 'Fulltext'. There is also a 'Print' button and a 'Propose table structure' link.

The 'Indexes' section shows two indexes:

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
Edit Rename Drop	PRIMARY	BTREE	Yes	No	<code>id</code>	0	A	No	
Edit Rename Drop	users_email_unique	BTREE	Yes	No	<code>email</code>	0	A	No	

Buttons for 'Add', 'Move columns', 'Normalize', and 'Go' are present. Below the indexes, there is a 'Create an index on' input field with '1' and a 'columns' dropdown, and a 'Go' button.

10.3.4 Menjalankan Laravel dan Breeze

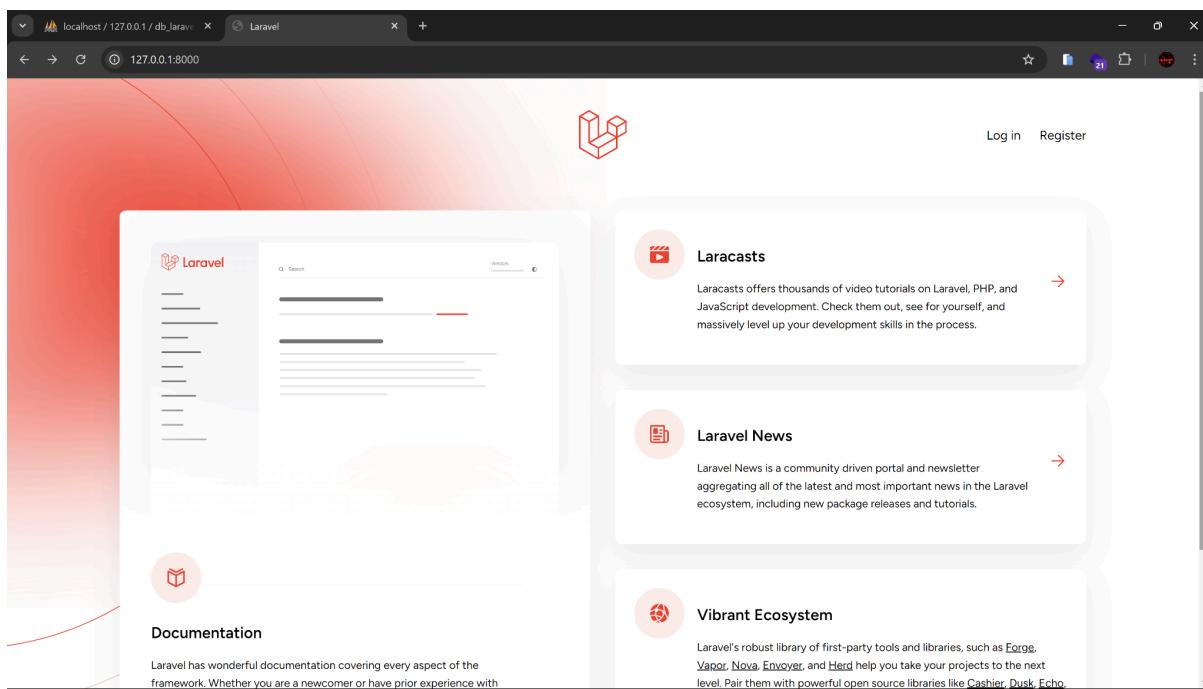
Berikutnya akan dilanjutkan dengan menjalankan proyek Laravel. Silahkan kembali ke VS Code, lalu pada terminal masukkan sintaks berikut untuk menjalankan Laravel.

```
php artisan serve
```

Setelah Laravel berjalan, berikutnya jalankan Breeze melalui Node.js .

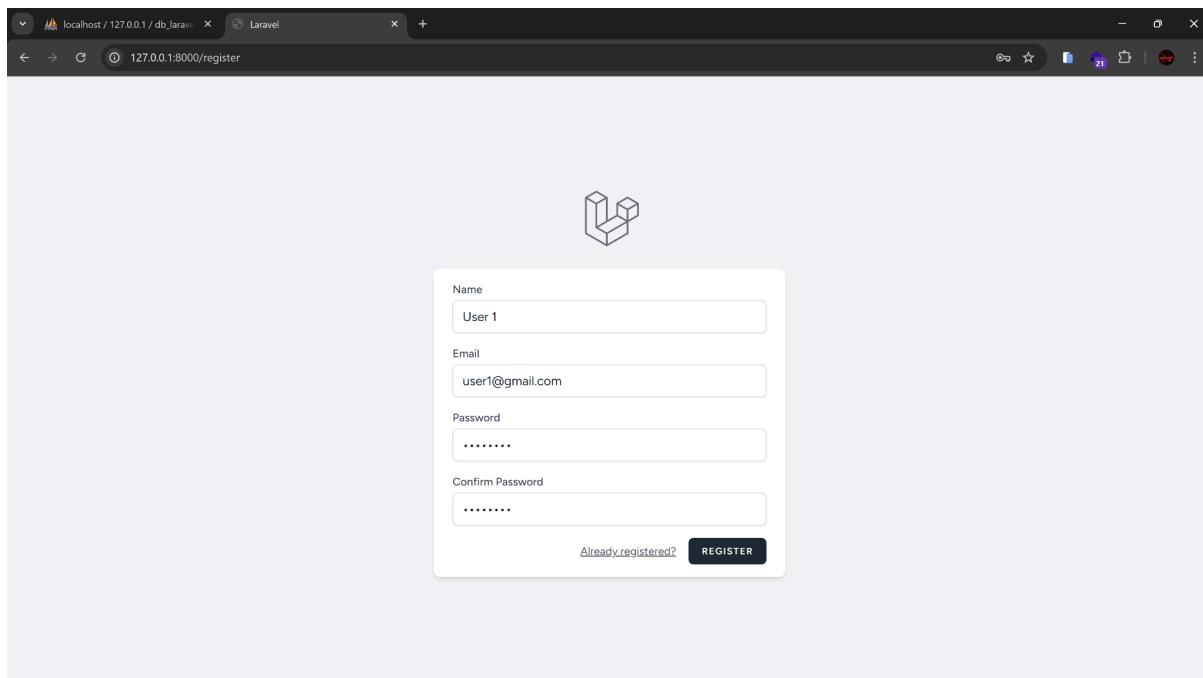
```
npm run dev
```

Setelah dua-duanya berjalan, berikutnya buka browser lalu masukkan url "<http://127.0.0.1:8000/>". Kita akan dialihkan ke 'welcome page' dari Laravel. Kita juga bisa lihat kalau di atas kanan sudah terdapat tombol "Log in" dan "Register" yang mana merupakan bagian dari hasil instalasi Breeze pada proyek Laravel.

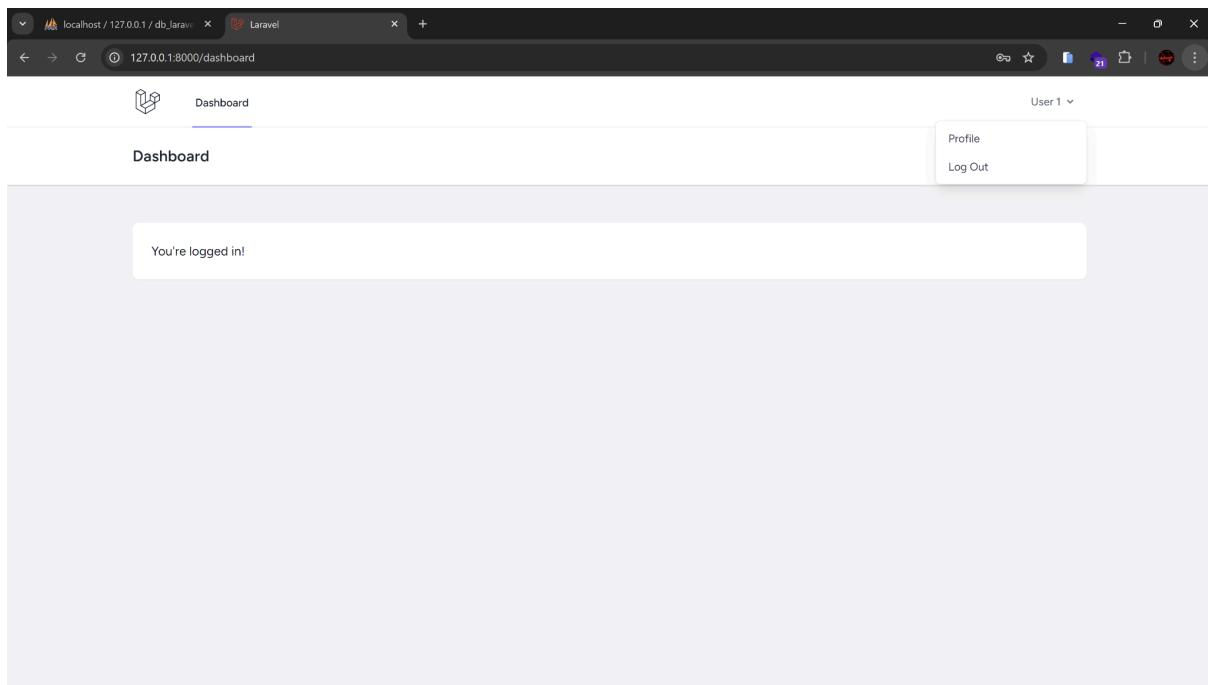


10.3.5 Mendaftarkan Akun User dan Akun Admin Melalui Fitur Register

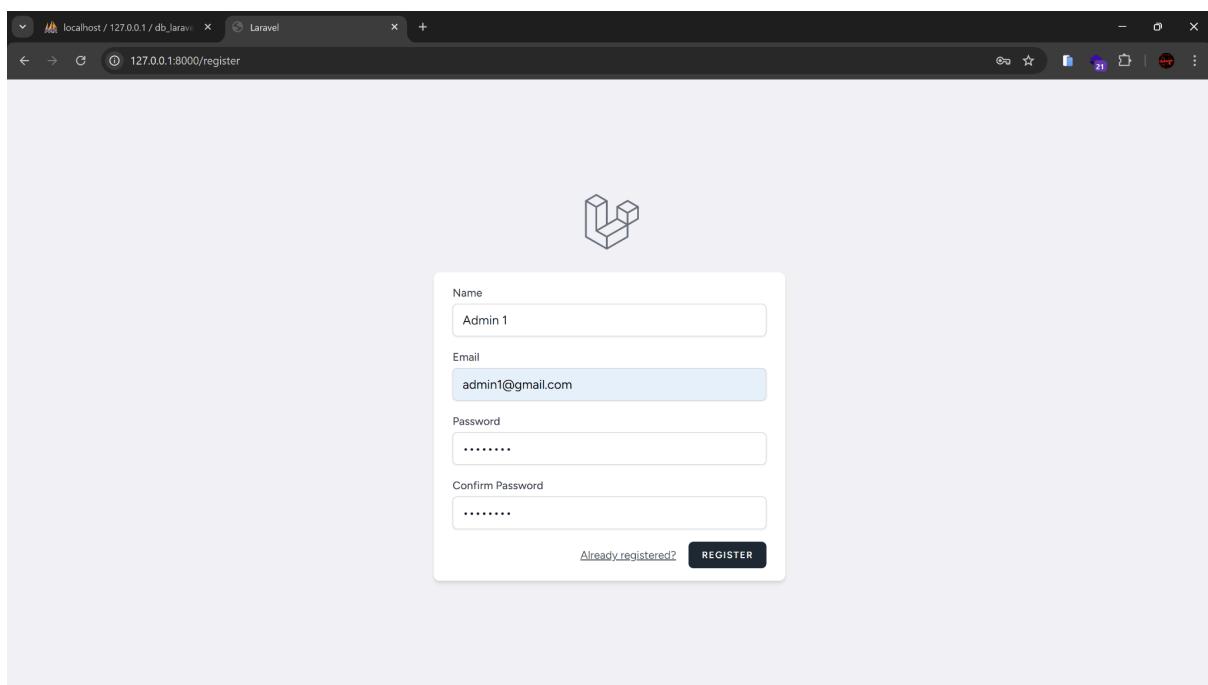
Berikutnya silahkan klik tombol “Register” pada ‘welcome page’ Laravel. Lalu kita akan dialihkan ke halaman berikut:



Silahkan mengisi field untuk mendaftarkan akun pertama dengan role “user”. Setelah menekan tombol “REGISTER”, kita akan langsung diarahkan ke halaman dashboard seperti berikut:



Langsung saja log out lalu berikutnya daftarkan untuk akun admin.



Setelah berhasil registrasi, buka kembali phpMyAdmin pada database “db_laravel_auth” dalam tabel “users”.

Showing rows 0 - 1 (Total: 2) Query took 0.0002 seconds.

	Edit	Copy	Delete	1 User 1	user1@gmail.com	user	NULL	\$2y\$12\$pbMTMw0XxCzbDtE5GQSA.IuXXe3yiFuYzr3Z7xGWz...	NULL	2024-08-29	2024-08-29	06:43:56
	Edit	Copy	Delete	2 Admin	admin1@gmail.com	user	NULL	\$2y\$12\$Lwp1Rjp87iEHDI/uAguoa2TYfQx/hkXP2aA78xsj...	NULL	2024-08-29	2024-08-29	06:48:06
With selected:	Edit	Copy	Delete									

Berikutnya value akun admin yang awalnya masih “user”, kita ubah menjadi “admin” seperti pada gambar di bawah.

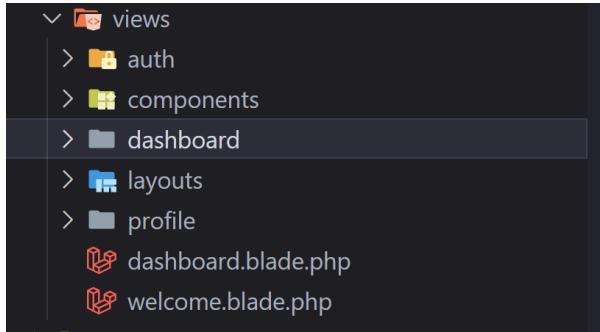
Showing rows 0 - 1 (Total: 2) Query took 0.0002 seconds.

UPDATE `users` SET `role` = 'admin' WHERE `user` = 'User 1' ✓ 1 row affected.

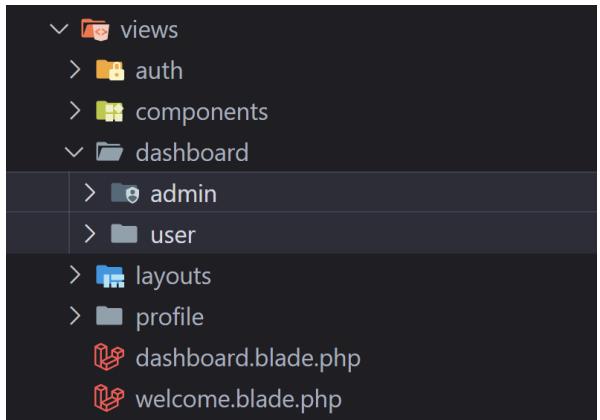
	Edit	Copy	Delete	1 User 1	user1@gmail.com	user	NULL	\$2y\$12\$pbMTMw0XxCzbDtE5GQSA.IuXXe3yiFuYzr3Z7xGWz...	NULL	2024-08-29	2024-08-29	06:43:56
	Edit	Copy	Delete	2 Admin	admin1@gmail.com	admin	NULL	\$2y\$12\$Lwp1Rjp87iEHDI/uAguoa2TYfQx/hkXP2aA78xsj...	NULL	2024-08-29	2024-08-29	06:48:06
With selected:	Edit	Copy	Delete									

10.3.6 Membuat Masing-Masing Halaman Dashboard untuk User dan Admin

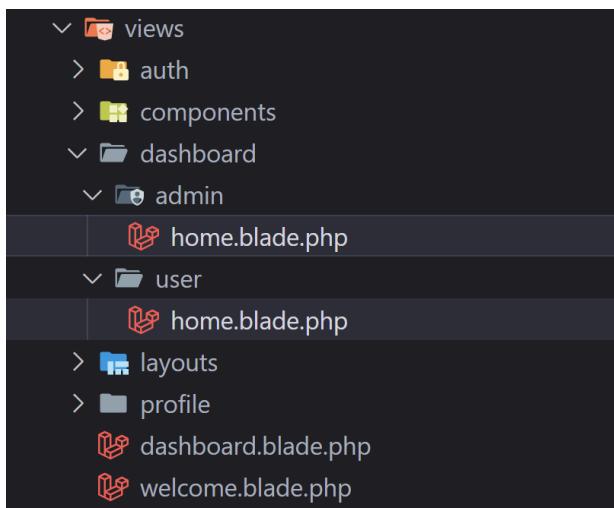
Selanjutnya kita kembali ke VS Code. Silahkan buat folder “dashboard” di dalam folder “views”.



Lalu di dalam folder “dashboard” tersebut, buat lagi dua folder untuk masing-masing role (“user” dan “admin”)



Setelah itu, buat file blade.php untuk tampilan halaman dashboard masing-masing user. Misalkan kita buat dengan nama “home.blade.php”.



Berikutnya buka file “dashboard.blade.php” yang terdapat di bawah. Copy keseluruhan isi file-nya lalu paste ke file “home.blade.php” yang terletak di dalam folder “dashboard/user”.

The screenshot shows the VS Code interface with the 'laravel-auth' project open. The Explorer sidebar on the left shows the project structure, including 'LARAVEL-AUTH' with subfolders like app, bootstrap, config, database, node_modules, public, resources, views, auth, components, dashboard, admin, and user. Inside the user folder, there are files named home.blade.php, layouts, profile, dashboard.blade.php, and welcome.blade.php. The 'home.blade.php' file is selected in the Explorer and is displayed in the main editor area. The code in the editor is:

```
<x-app-layout>
  <x-slot name="header">
    <h2 class="font-semibold text-xl text-gray-800 leading-tight">
      {{ __('Dashboard') }}
    </h2>
  </x-slot>

  <div class="py-12">
    <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
      <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
        <div class="p-6 text-gray-900">
          {{ __("You're logged in!") }}
        </div>
      </div>
    </div>
  </div>
</x-app-layout>
```

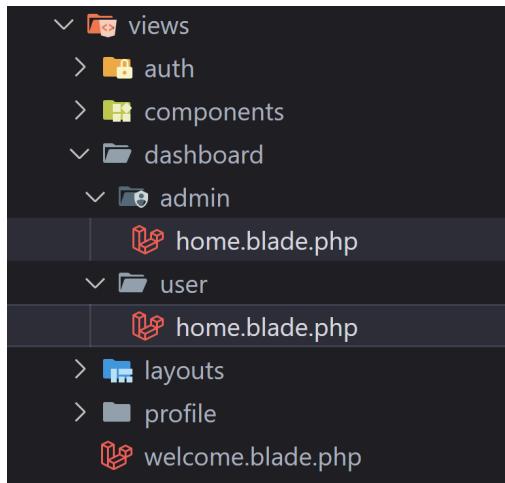
Berikutnya paste juga ke file “home.blade.php” yang terletak di dalam folder “dashboard/admin”. Jangan lupa untuk memberi keterangan berupa halaman ini merupakan halaman admin. Sebagai contoh, di sini kita text “Dashboard Admin” pada header.

The screenshot shows the VS Code interface with the 'laravel-auth' project open. The Explorer sidebar on the left shows the project structure, including 'LARAVEL-AUTH' with subfolders like app, bootstrap, config, database, node_modules, public, resources, views, auth, components, dashboard, admin, and user. Inside the admin folder, there are files named home.blade.php, user, layouts, profile, dashboard.blade.php, and welcome.blade.php. The 'home.blade.php' file is selected in the Explorer and is displayed in the main editor area. The code in the editor is:

```
<x-app-layout>
  <x-slot name="header">
    <h2 class="font-semibold text-xl text-gray-800 leading-tight">
      {{ __('Dashboard Admin') }}
    </h2>
  </x-slot>

  <div class="py-12">
    <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
      <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
        <div class="p-6 text-gray-900">
          {{ __("You're logged in as Admin!") }}
        </div>
      </div>
    </div>
  </div>
</x-app-layout>
```

Berikutnya kita dapat menghapus file “dashboard.blade.php” yang terletak di luar.



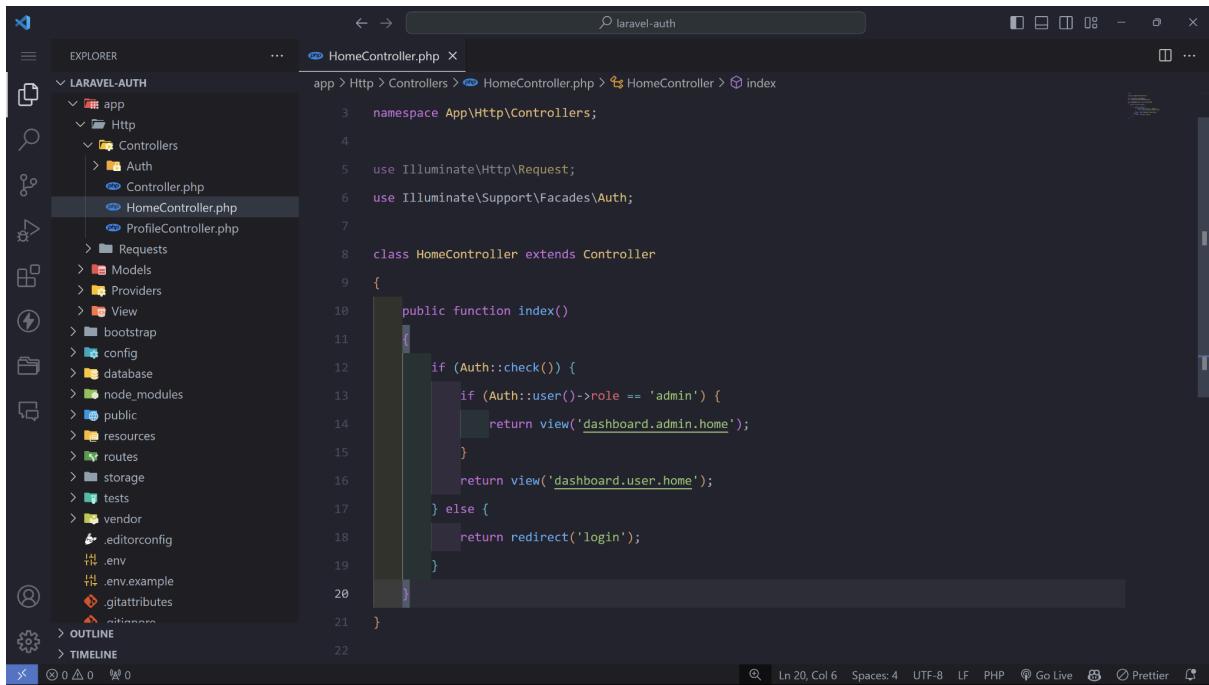
10.3.7 Memasukkan Halaman Dashboard User dan Admin ke Route

Selanjutnya kita akan mendaftarkan route untuk halaman dashboard user dan admin yang dihandle menggunakan controller agar kita bisa mengarahkan jika yang login adalah user, maka arahkan ke dashboard khusus untuk user. Lalu jika yang login adalah admin, maka akan diarahkan ke dashboard khusus untuk admin.

Maka kita perlu untuk membuat dashboard/home controller terlebih dahulu. Silahkan kembali buka Terminal lalu masukkan sintaks berikut:

```
php artisan make:controller HomeController
```

Setelah controller berhasil dibuat, buka file "HomeController.php" tersebut yang terletak di folder "app/Http/Controllers". Pada function index(), masukkan kode seperti pada gambar di bawah.

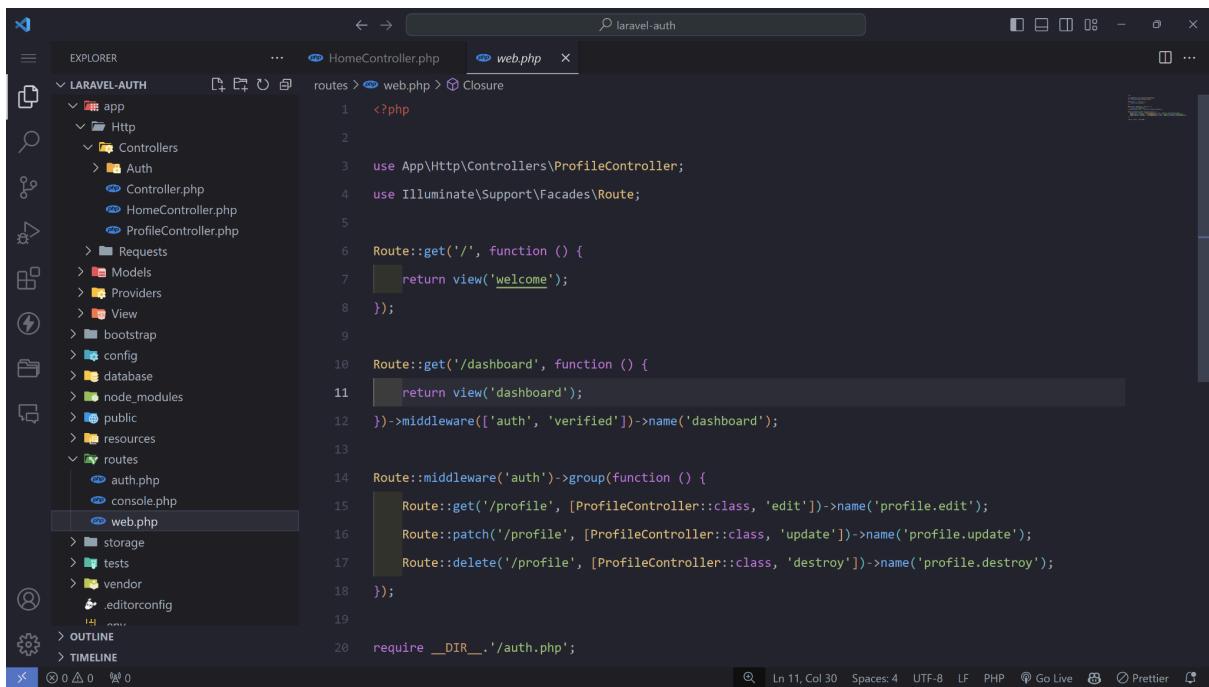


The screenshot shows the VS Code interface with the file `HomeController.php` open in the editor. The code handles the `index` method, which checks if the user is authenticated. If the user is an admin, it returns the `dashboard.admin.home` view; otherwise, it returns the `dashboard.user.home` view. If the user is not authenticated, it redirects to the `login` page.

```
namespaces App\Http\Controllers;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class HomeController extends Controller
{
    public function index()
    {
        if (Auth::check()) {
            if (Auth::user()->role == 'admin') {
                return view('dashboard.admin.home');
            }
            return view('dashboard.user.home');
        } else {
            return redirect('login');
        }
    }
}
```

Berikutnya, buka file “web.php” yang terletak pada folder “routes/web.php”.



The screenshot shows the VS Code interface with the file `web.php` open in the editor. It defines routes for the application. It starts with a closure route for the root path (`/`) that returns the `welcome` view. It then defines a middleware group for the `auth` and `verified` routes, applying it to the `/dashboard` route, which returns the `dashboard` view. Finally, it defines three specific routes for profile management: `/profile` (GET), `/profile` (PATCH), and `/profile` (DELETE), each associated with the `ProfileController`.

```
<?php

use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
});

Route::get('/dashboard', function () {
    return view('dashboard');
})->middleware(['auth', 'verified'])->name('dashboard');

Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
    Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
    Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
});

require __DIR__.'/auth.php';
```

Silahkan hapus atau comment kode pada baris 10-12, lalu gantikan dengan kode seperti pada gambar di bawah.

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure under "LARAVEL-AUTH". The "routes" folder contains "web.php". Other files like "HomeController.php", "ProfileController.php", and "AuthController.php" are also listed.
- routes/web.php** file content:

```
<?php

use App\Http\Controllers\HomeController;
use App\Http\Controllers\ProfileController;
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('welcome');
});

Route::get('/dashboard', [HomeController::class, 'index'])->middleware(['auth', 'verified'])->name('dashboard');

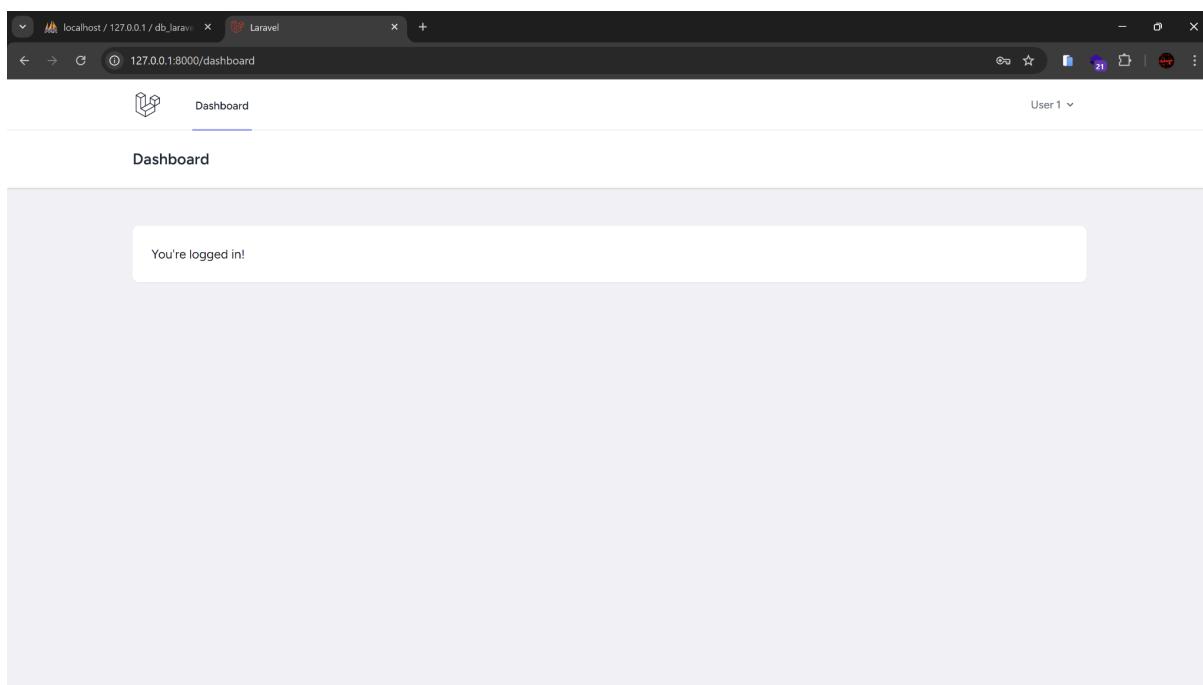
// Route::get('/dashboard', function () {
//     return view('dashboard');
// })->middleware(['auth', 'verified'])->name('dashboard');

Route::middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'edit'])->name('profile.edit');
    Route::patch('/profile', [ProfileController::class, 'update'])->name('profile.update');
    Route::delete('/profile', [ProfileController::class, 'destroy'])->name('profile.destroy');
});

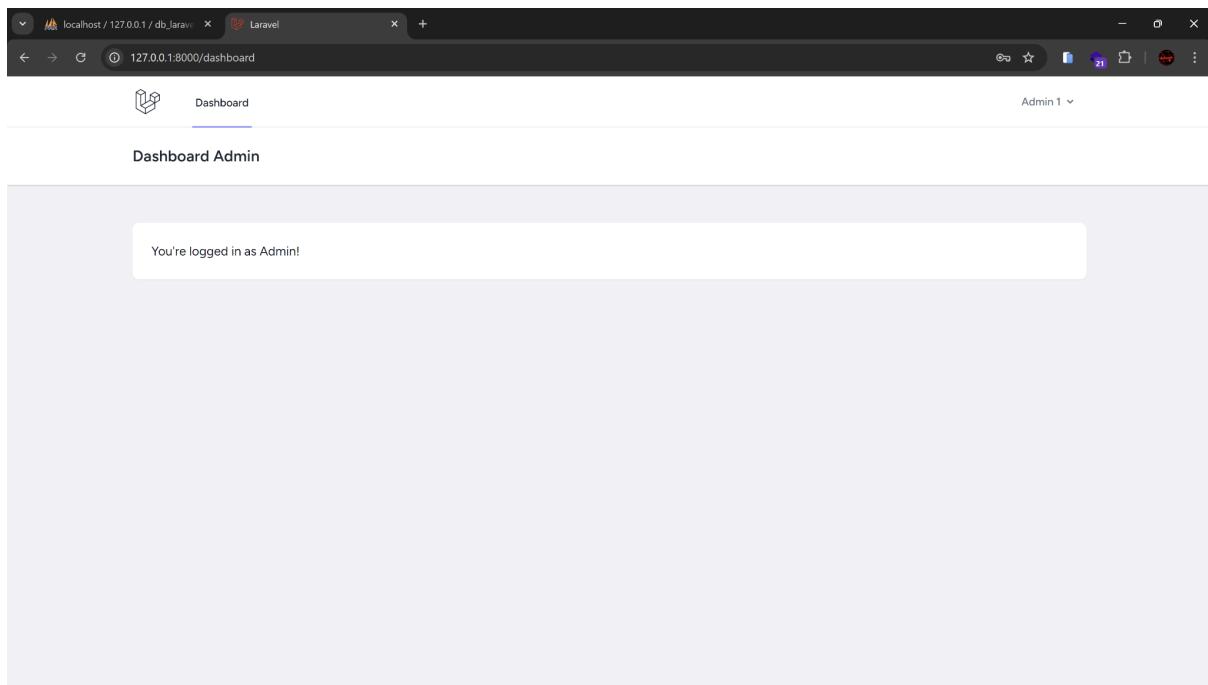
require __DIR__ . '/auth.php';
```

- Bottom status bar: Ln 11, Col 113, Spaces: 4, UTF-8, LF, PHP, Go Live, Prettier.

Berikutnya, silahkan buka kembali web proyek Laravel lalu login dengan masing-masing akun.



(Akun User)



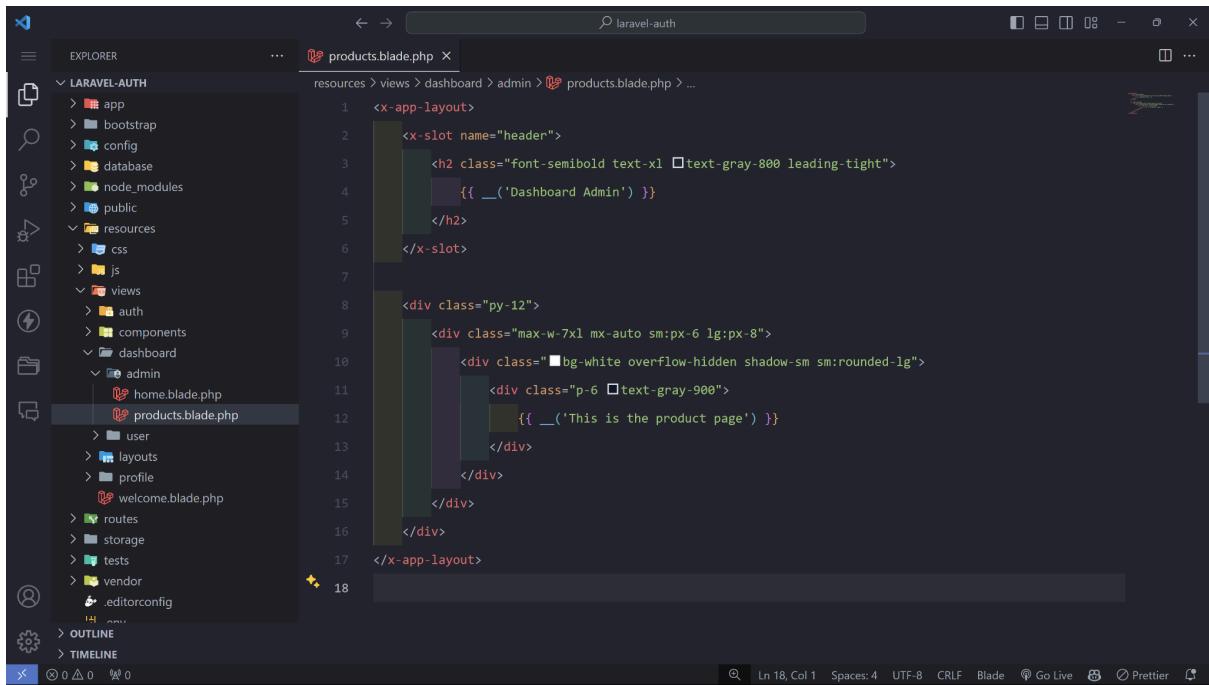
(Akun Admin)

Dengan demikian, kita sudah berhasil menghandle jika yang login adalah admin, maka akan diarahkan ke dashboard admin, begitupun untuk role user.

10.3.8 Membuat Middleware dan Otorisasi

Berikutnya kita buat sebuah halaman baru yang hanya bisa diakses oleh admin. Jika yang login adalah user biasa, maka akses ke halaman tersebut akan ditolak. Kita ambil contoh akan membuat halaman “/products”.

Silahkan membuat file baru dengan nama “product.blade.php” di dalam folder “resources/views/dashboard/admin”. Lalu copy code pada “home.blade.php” yang ada di folder “dashboard/admin”. Ubah text di bawah sehingga memberikan keterangan kalau ini adalah halaman produk.



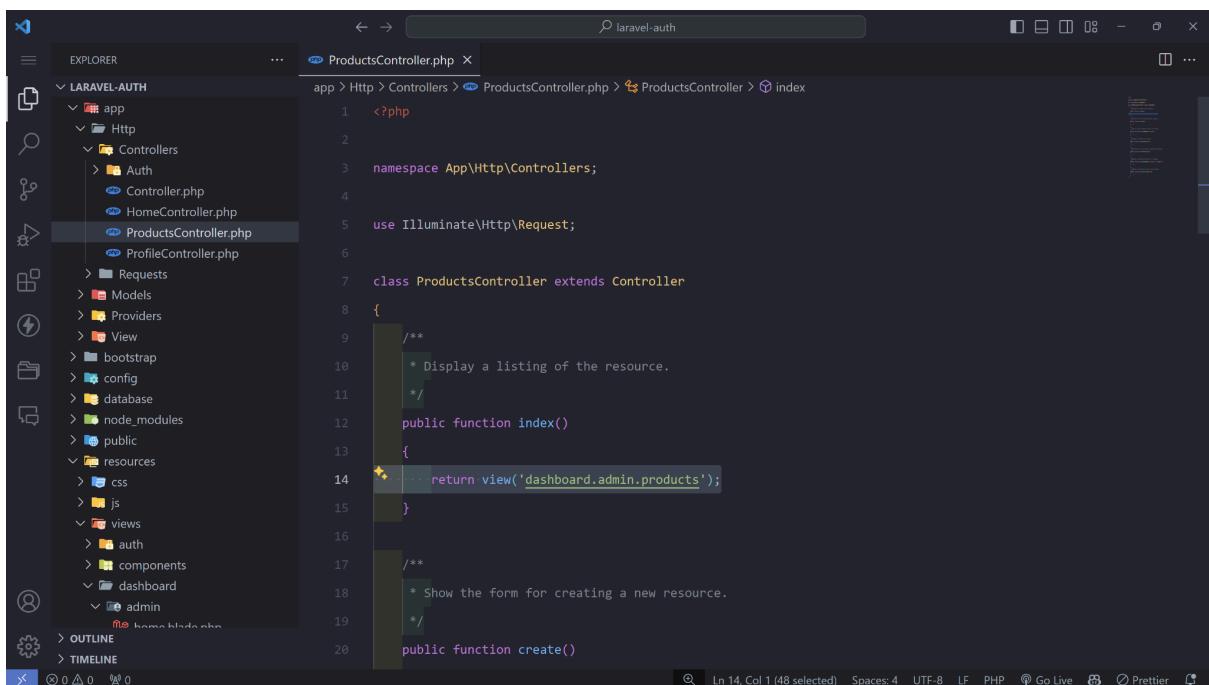
```
<x-app-layout>
    <x-slot name="header">
        <h2 class="font-semibold text-xl text-gray-800 leading-tight">
            {{ __('Dashboard Admin') }}
        </h2>
    </x-slot>

    <div class="py-12">
        <div class="max-w-7xl mx-auto sm:px-6 lg:px-8">
            <div class="bg-white overflow-hidden shadow-sm sm:rounded-lg">
                <div class="p-6 text-gray-900">
                    {{ __('This is the product page') }}
                </div>
            </div>
        </div>
    </div>
</x-app-layout>
```

Berikutnya kita buat controller dengan nama “ProductsController” untuk handle menampilkan halaman produk.

```
php artisan make:controller ProductsController --resource
```

Lalu pada file ProductsController.php , kita menampilkan halaman produk yang diberi nama “products.blade.php”.



```
<?php

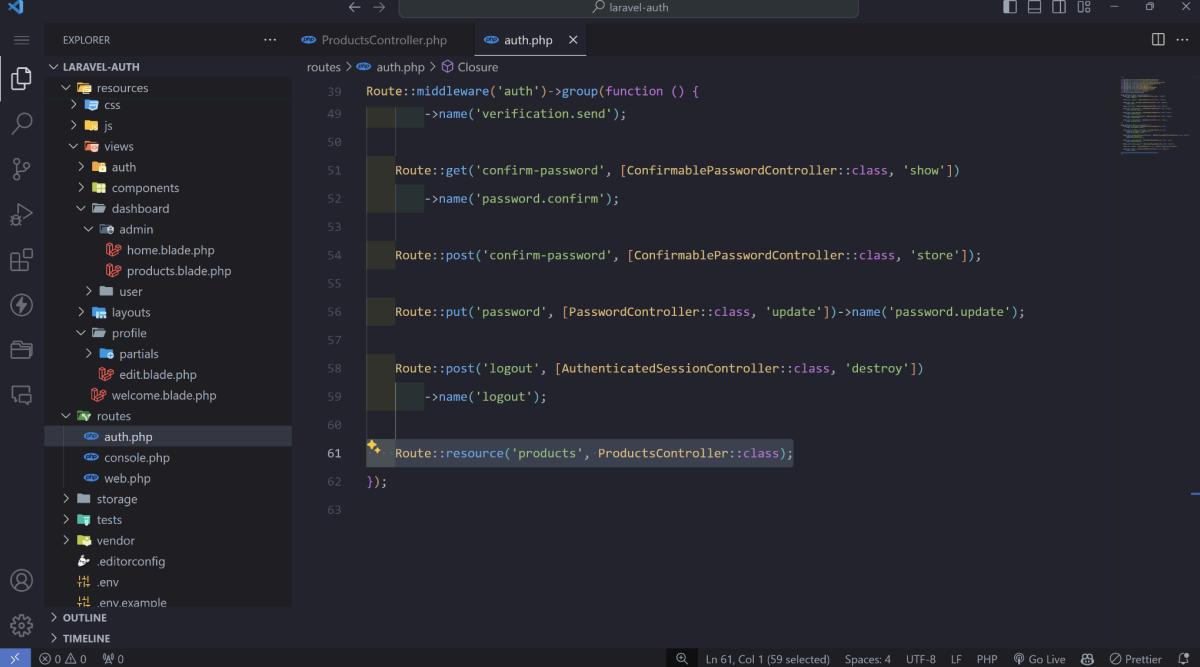
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ProductsController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        return view('dashboard.admin.products');
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
```

Berikutnya kita daftarkan file “products.blade.php” menggunakan “ProductsController” ke routes. Kita buka file “auth.php” yang terletak di folder “routes”.



```
Route::middleware('auth')->group(function () {
    // ...
    Route::get('confirm-password', [ConfirmablePasswordController::class, 'show'])
        ->name('password.confirm');

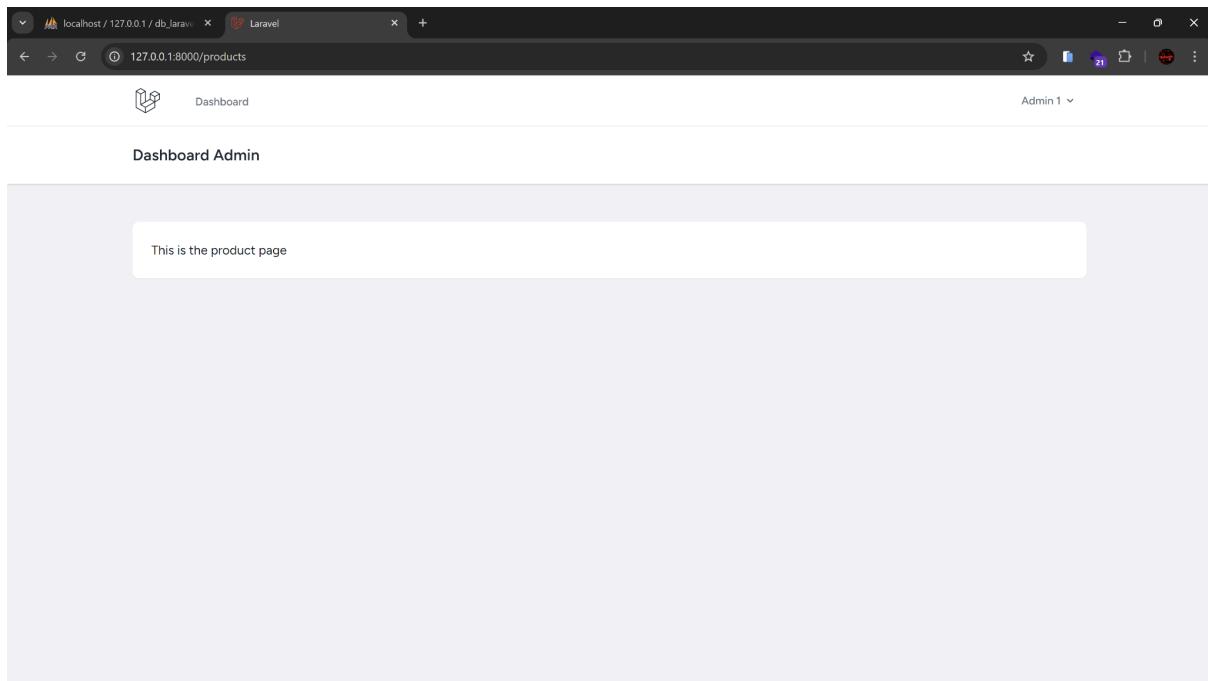
    Route::post('confirm-password', [ConfirmablePasswordController::class, 'store']);
    Route::put('password', [PasswordController::class, 'update'])
        ->name('password.update');

    Route::post('logout', [AuthenticatedSessionController::class, 'destroy'])
        ->name('logout');
});

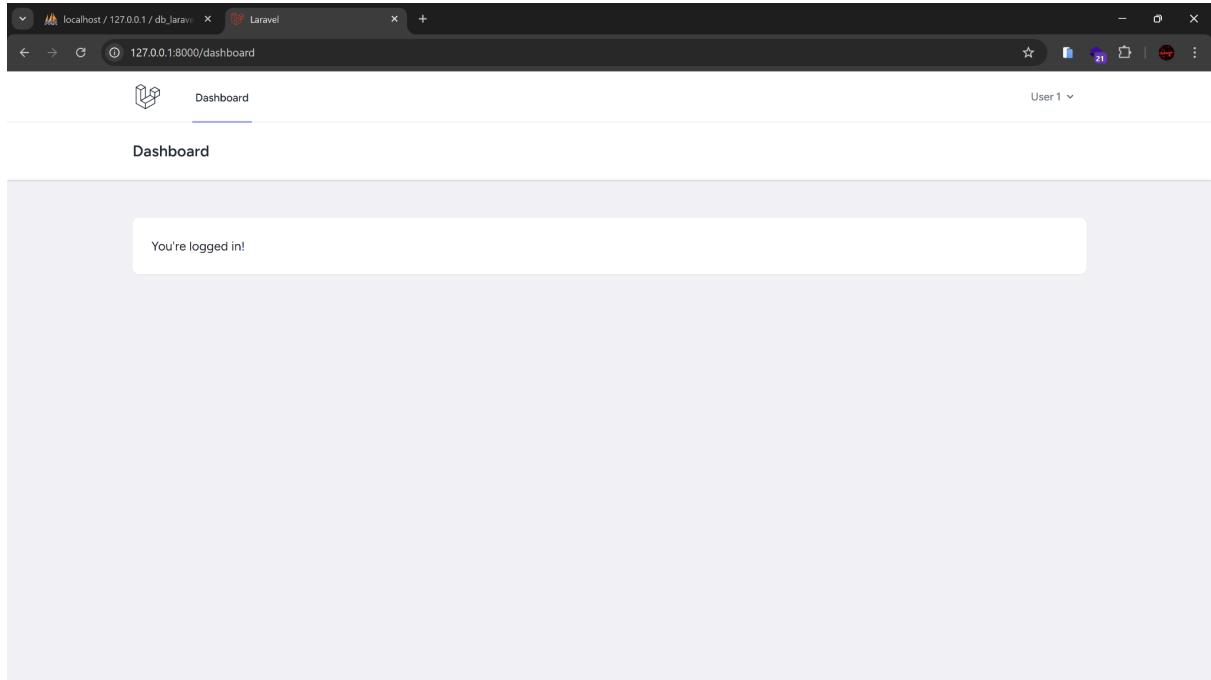
Route::resource('products', ProductsController::class);
```

Kenapa bukan di file “web.php” ? Kita menambahkan route tersebut melalui file “auth.php” agar lebih rapi, sekaligus kita memasukkannya ke daftar route yang hanya dapat diakses jika sudah login (dibungkus dengan middleware “auth”).

Jika kita mengakses halaman tersebut menggunakan halaman admin, maka kita bisa masuk ke dalamnya.



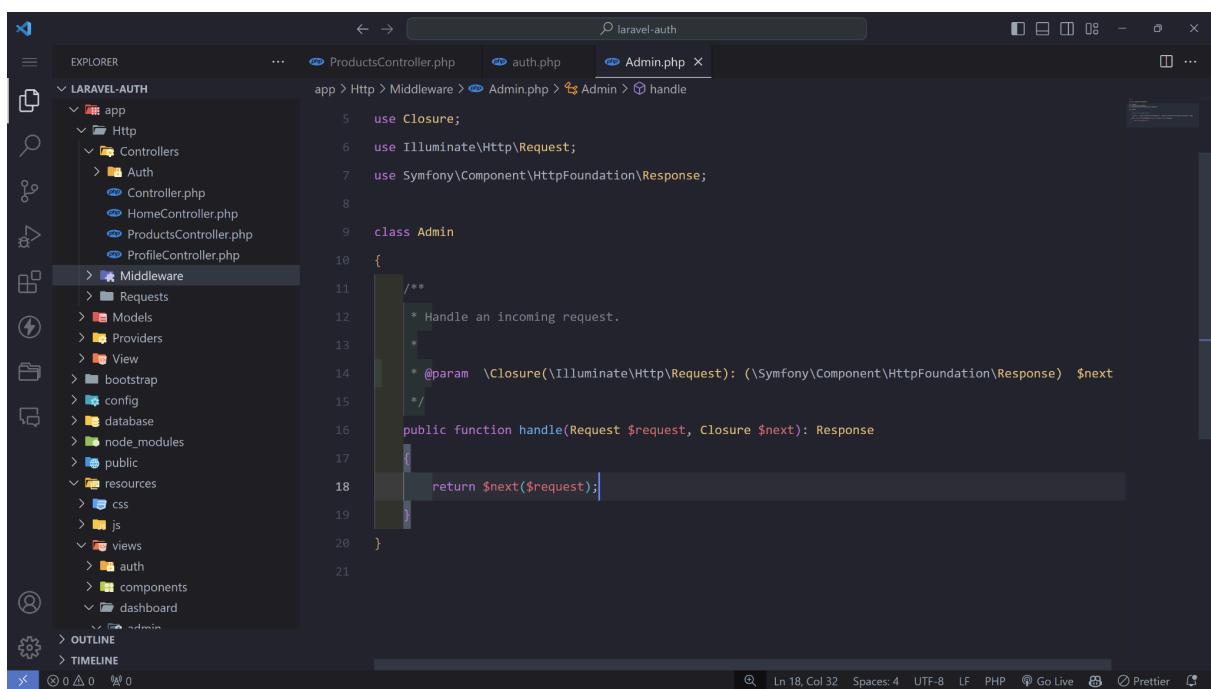
Tetapi jika kita mengaksesnya menggunakan akun user, maka kita tetap bisa masuk ke dalamnya.



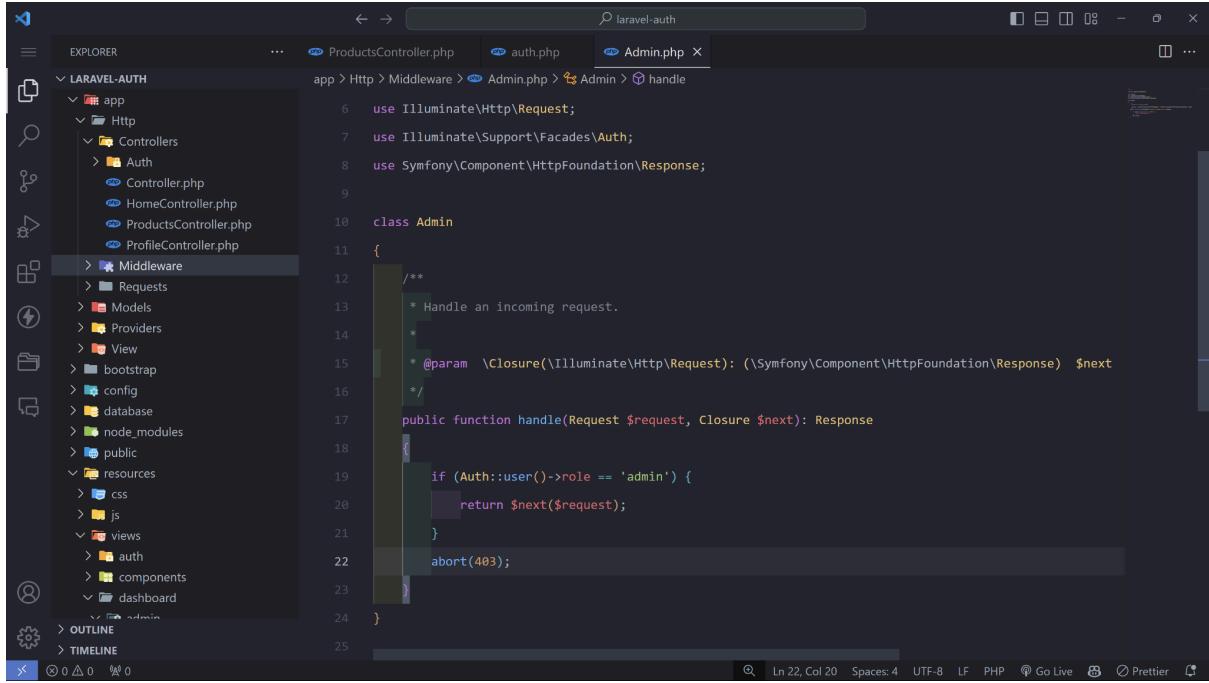
Kita akan menghandle masalah tersebut dengan cara membuat middleware “admin”.

```
php artisan make:middleware Admin
```

Setelah itu, buka file middleware “Admin.php” yang terletak di folder “app/Http/Middleware”.



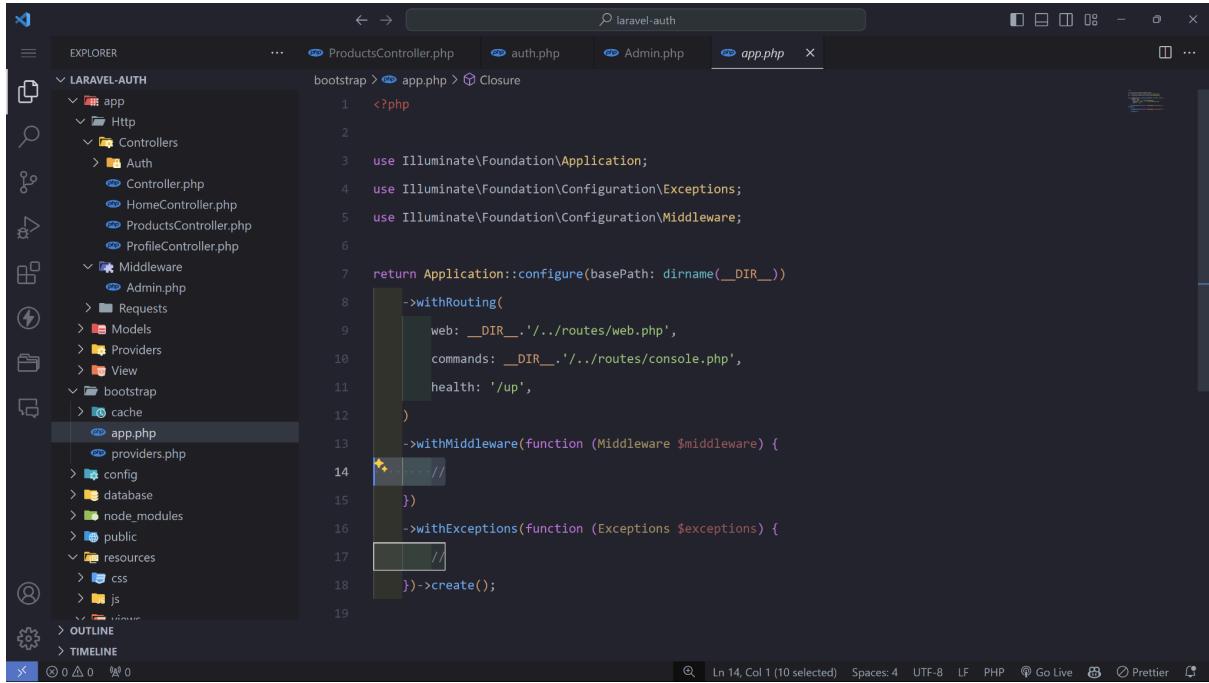
Lalu ubah kode yang ada di dalam function handle(), menjadi seperti pada gambar di bawah.



```
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Symfony\Component\HttpFoundation\Response;

class Admin
{
    /**
     * Handle an incoming request.
     *
     * @param \Closure(\Illuminate\Http\Request) $next
     */
    public function handle(Request $request, Closure $next)
    {
        if (Auth::user()->role == 'admin') {
            return $next($request);
        }
        abort(403);
    }
}
```

Kita sudah membuat middleware “admin”, selanjutnya kita harus mendaftarkannya ke file “app.php” yang terletak di folder “bootstrap”.

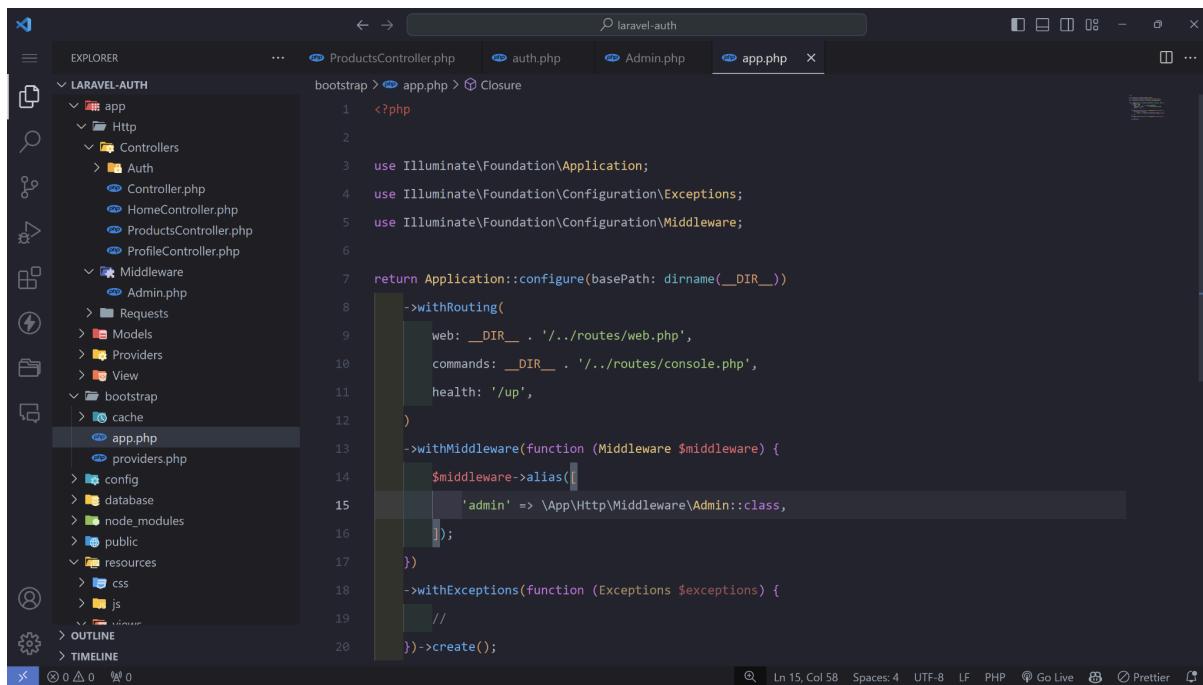


```
<?php

use Illuminate\Foundation\Application;
use Illuminate\Foundation\Configuration\Exceptions;
use Illuminate\Foundation\Configuration\Middleware;

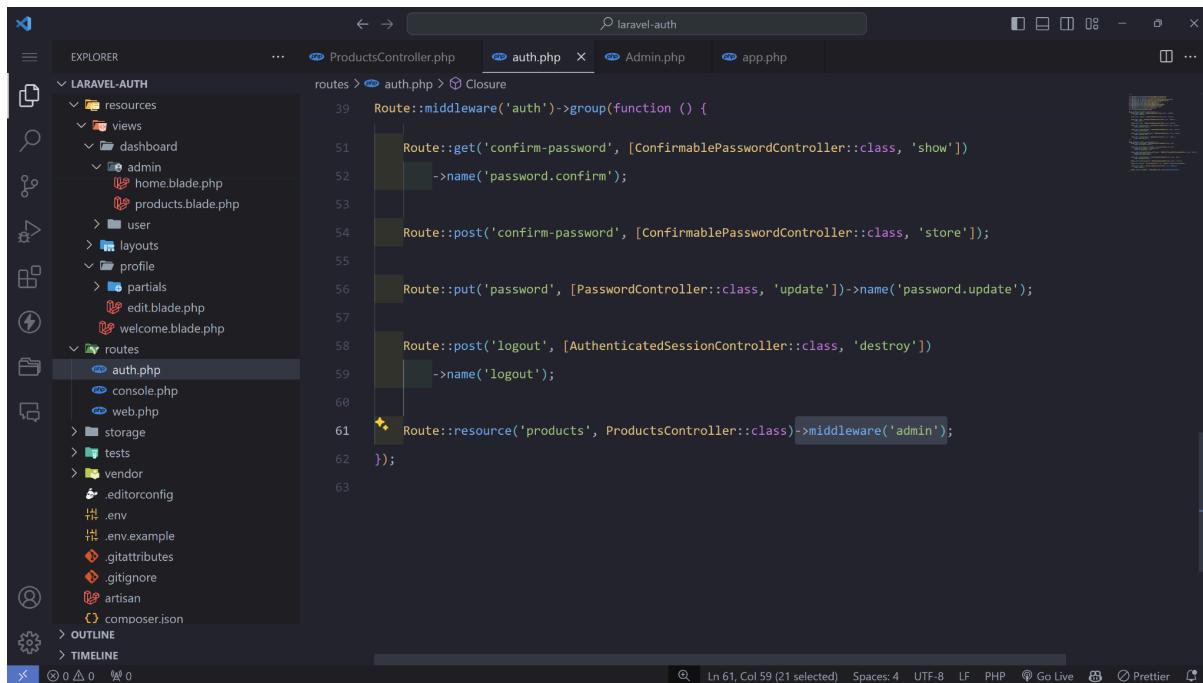
return Application::configure(basePath: dirname(__DIR__))
    ->withRouting(
        web: __DIR__.'/../routes/web.php',
        commands: __DIR__.'/../routes/console.php',
        health: '/up',
    )
    ->withMiddleware(function (Middleware $middleware) {
        ...
    })
    ->withExceptions(function (Exceptions $exceptions) {
        ...
    })
->create();
```

Kita akan mendaftarkan middleware “admin” di dalam function withMiddleware() pada line 14 tersebut.



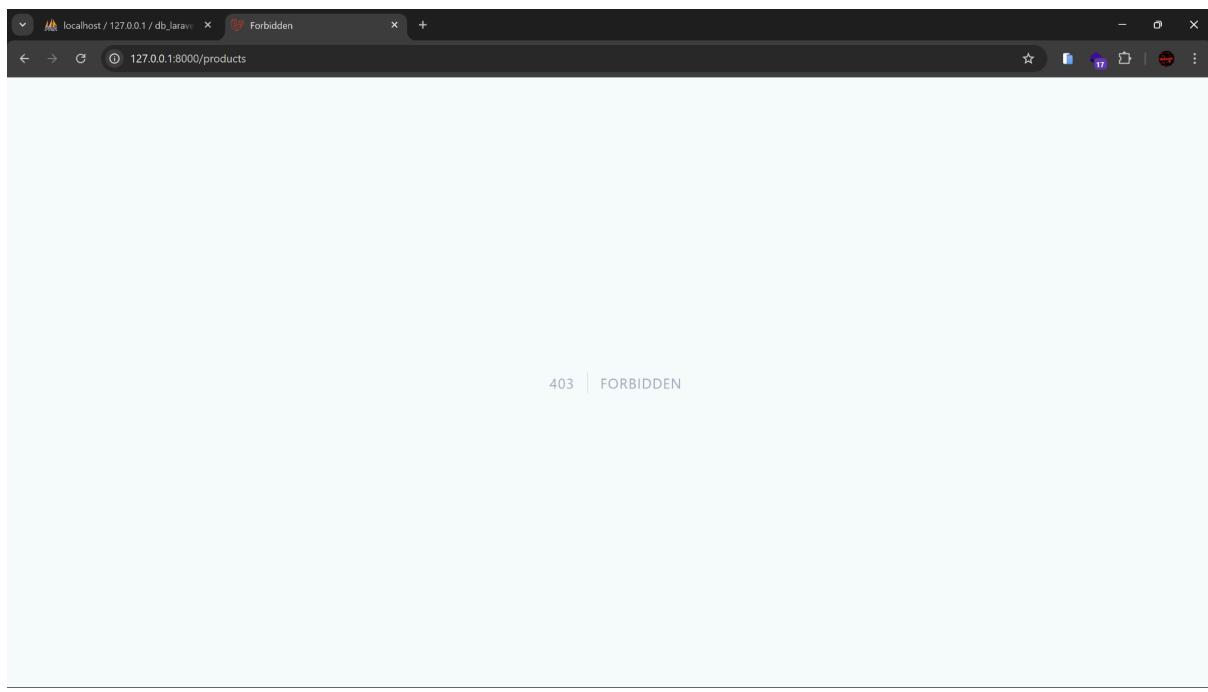
```
bootstrap > app.php > Closure
1 <?php
2
3 use Illuminate\Foundation\Application;
4 use Illuminate\Foundation\Configuration\Exceptions;
5 use Illuminate\Foundation\Configuration\Middleware;
6
7 return Application::configure(basePath: dirname(__DIR__))
8 ->withRouting(
9     web: __DIR__ . '/../routes/web.php',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

Kita sudah mendaftarkan middleware “admin” dalam proyek Laravel kita. Selanjutnya kita bisa gunakan middleware tersebut di file konfigurasi route untuk halaman produk yang sebelumnya kita sudah buat.

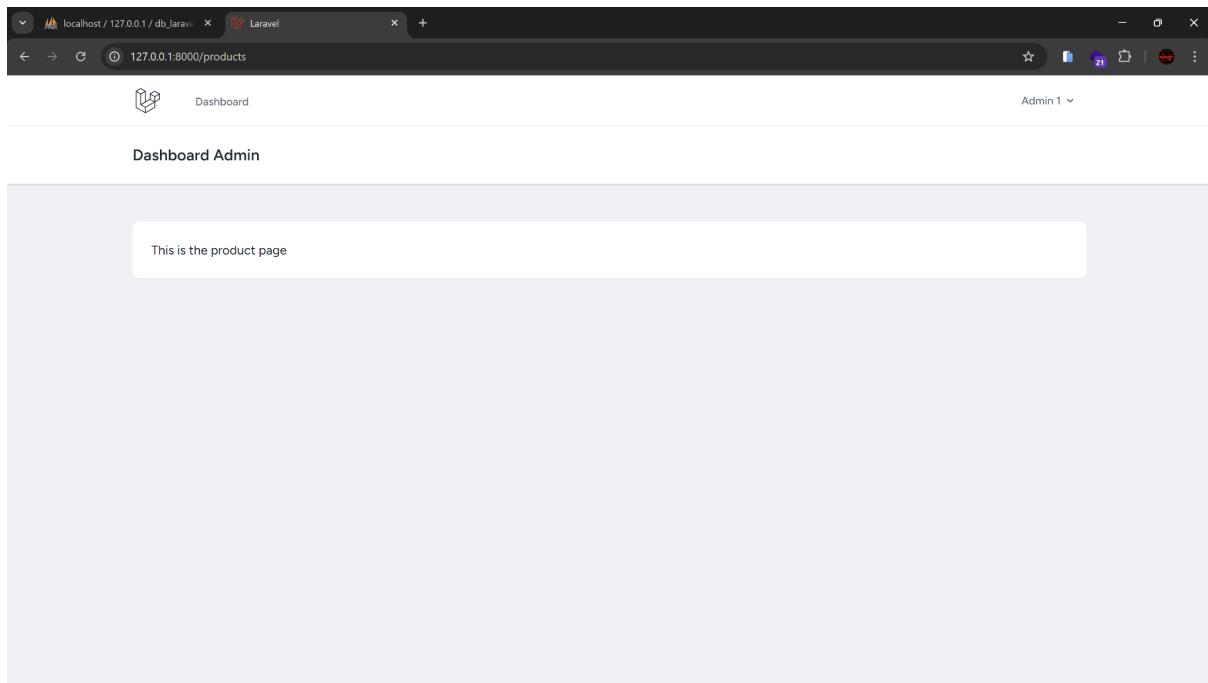


```
routes > auth.php > Closure
39 Route::middleware('auth')->group(function () {
40
41     Route::get('confirm-password', [ConfirmablePasswordController::class, 'show'])
42         ->name('password.confirm');
43
44     Route::post('confirm-password', [ConfirmablePasswordController::class, 'store']);
45
46     Route::put('password', [PasswordController::class, 'update'])->name('password.update');
47
48     Route::post('logout', [AuthenticatedSessionController::class, 'destroy'])
49         ->name('logout');
50
51     Route::resource('products', ProductsController::class)->middleware('admin');
52 });
53
54
55
56
57
58
59
60
61
62
63
```

Penamaan middleware yang ada di file “app.php” sebagai key yang akan kita gunakan untuk menggunakan middleware tersebut. Karena sebelumnya di file “bootstrap/app.php” kita mendaftarkan dengan key “admin”, maka ketika kita ingin menggunakannya di file konfigurasi routes (“web.php” maupun “auth.php”), kita menambahkan `->middleware('admin')`



Sekarang jika kita login sebagai user dan masuk ke halaman “/products” dengan cara ubah di URL web, maka akses akan diblokir dengan menampilkan halaman status 403.



Sedangkan jika kita login sebagai admin dan mengakses halaman “/products”, maka akses akan tetap diberikan.