Laporan Modul 3: Controller

Mata Kuliah: Workshop Web Lanjut
Nama: Muhammad Dhiyaul Atha
NIM: 2024573010075
Kelas: TI 2B

Abstrak

Laporan ini membahas konsep dasar *Controller* pada framework **Laravel 12** yang merupakan bagian penting dari pola arsitektur **Model-View-Controller (MVC)**. *Controller* berperan dalam menangani logika aplikasi dengan menerima permintaan dari rute (*route*), memproses input pengguna, dan mengembalikan respons yang sesuai. Modul ini menjelaskan jenis-jenis *controller*, cara pembuatannya menggunakan perintah Artisan, serta praktik terbaik dalam pengorganisasian logika aplikasi agar kode tetap terstruktur, efisien, dan mudah dipelihara.

BAB I – Dasar Teori

1.1 Pengertian Controller

Dalam arsitektur **MVC (Model-View-Controller)**, *controller* bertindak sebagai jembatan antara *model* dan *view*. *Controller* menerima input dari pengguna melalui *route*, berinteraksi dengan *model* untuk mengambil atau memproses data, lalu mengembalikan *response* yang sesuai ke *view*.
Dengan adanya *controller*, logika aplikasi dapat dipisahkan dari tampilan sehingga pengembangan menjadi lebih terstruktur dan mudah dikelola.

Untuk membuat sebuah *controller*, digunakan perintah Artisan:

```
php artisan make:controller NamaController
```

Perintah ini akan membuat file controller di direktori:

```
app/Http/Controllers/
```

1.2 Jenis-Jenis Controller pada Laravel

Laravel mendukung beberapa jenis *controller* yang dirancang untuk berbagai pola pengembangan aplikasi, yaitu:

1. Basic Controller (Controller Dasar)

Jenis *controller* ini adalah kelas standar yang berisi beberapa metode untuk menangani berbagai aksi atau endpoint dalam satu kelas. Dibuat dengan perintah:

```
php artisan make:controller PageController
```

---

2. Resource Controller (Controller Sumber Daya)

Digunakan untuk menangani operasi CRUD (*Create, Read, Update, Delete*) secara otomatis berdasarkan konvensi RESTful. Dibuat dengan perintah:

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

Perintah ini menghasilkan metode bawaan seperti: `index()`, `create()`, `store()`, `show()`, `edit()`, `update()`, dan `destroy()`.

---

3. Invokable Controller (Controller Satu Aksi)

Jenis *controller* ini hanya memiliki satu metode utama dan biasanya digunakan untuk aksi tunggal seperti menampilkan halaman atau menjalankan proses tertentu. Dibuat dengan perintah:

```
php artisan make:controller ContactController --invokable
```

---

4. Pengelompokan Rute dengan Controller

Laravel memungkinkan pengelompokan rute berdasarkan *controller* untuk menjaga struktur aplikasi tetap rapi. Pendekatan ini membantu pengorganisasian kode ketika terdapat banyak endpoint yang masih berada dalam konteks yang sama.

---

5. Injeksi Permintaan dan Ketergantungan

Laravel mendukung *dependency injection*, di mana objek seperti `Request` atau layanan (*service class*) dapat disuntikkan langsung ke dalam metode *controller*. Fitur ini membantu pengembang menulis kode yang modular dan mudah diuji. Sebagai contoh konsep, data dapat diproses dengan cara:

```
php artisan make:request StoreUserRequest
```

Perintah ini digunakan untuk membuat kelas validasi yang terpisah agar kode lebih bersih.

---

6. Validasi Permintaan (Request Validation)

Laravel menyediakan dua cara utama untuk melakukan validasi:

- **Langsung di dalam metode controller**, menggunakan fungsi `validate()`.
- **Menggunakan Form Request**, di mana validasi ditempatkan pada kelas khusus untuk menjaga kode tetap bersih dan terstruktur.

Pendekatan kedua direkomendasikan karena memisahkan logika validasi dari *controller* utama.

---

7. Mengembalikan Respons dari Controller

Controller di Laravel dapat mengembalikan berbagai jenis respons tergantung kebutuhan aplikasi, antara lain:

- **View:** menampilkan halaman tampilan kepada pengguna.
- **JSON:** mengirimkan data dalam format JSON, biasanya untuk API.
- **Redirect:** mengarahkan pengguna ke rute lain setelah proses selesai.
- **Response Kustom:** mengembalikan data dengan format dan status tertentu sesuai kebutuhan.

BAB II – Hasil dan Pembahasan

2.1 Praktikum 1 – Menangani Request dan Response View di Laravel 12

1. Langkah-Langkah

a. Membuat Project Baru

Untuk membuat project baru dengan perintah:

```
laravel new lab-view
```

Perintah tersebut akan membuat proyek baru dengan folder bernama *lab-view*.

b. Membuat Sebuah Controller

Untuk membuat controller ketik perintah:

```
php artisan make:controller DemoController
```

Perintah tersebut akan membuat file *DemoController* di:

```
app/Http/Controllers/DemoController.php
```

Berikut isi dari *DemoController* tersebut:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class DemoController extends Controller
{
    public function hello(){
        $name = "Laravel Learner";
        return view('hello', ['name' => $name]);
    }

    public function greet($name){
        return view('greet', ['name' => ucfirst($name)]);
    }

    public function search(Request $request){
        $keyword = $request->query('q','none');
        return view('search', ['keyword' => $keyword]);
    }
}
```

c. Mendefinisikan Route

Setelah *controller* dibuat, langkah selanjutnya adalah membuat *route* untuk controller tersebut.

```php
<?php

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

Route::get('/hello', [DemoController::class, 'hello']);
Route::get('/greet/{name}', [DemoController::class, 'greet']);
Route::get('/search', [DemoController::class, 'search']);
```

Ketiga *route* tersebut digunakan untuk memanggil metode-metode yang ada pada *DemoController.php*.

d. Membuat View

Selanjutnya membuat *view* untuk ketiga *route* yang telah dibuat. Berikut isi dari masing-masing *view*:

- **resources/view/greet.blade.php**

```blade
<!DOCTYPE html>
<html>

<head>
    <title>Hello</title>
</head>

<body>
    <h1>Nice to meet you, {{ $name }}</h1>
</body>
</html>
```

- **resources/view/hello.blade.php**

```blade
<!DOCTYPE html>
<html>

<head>
    <title>Hello</title>
</head>

<body>

    <h1>Hello, {{ $name }}</h1>

</body>
</html>
```

- **resources/view/search.blade.php**

```blade
<!DOCTYPE html>
<html>

<head>
    <title>Hello</title>
</head>

<body>

    <h1>You searched for : <strong> {{ $keyword }} </strong> </h1>

</body>
</html>
```

e. Menjalankan Program

Terakhir jalankan dengan perintah:

```
php artisan serve
```

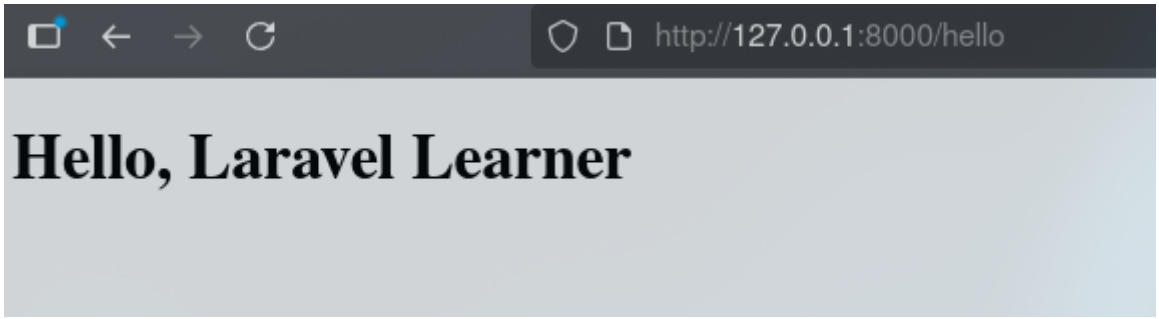
Maka server akan berjalan di:

```
http://127.0.0.1:8000
```

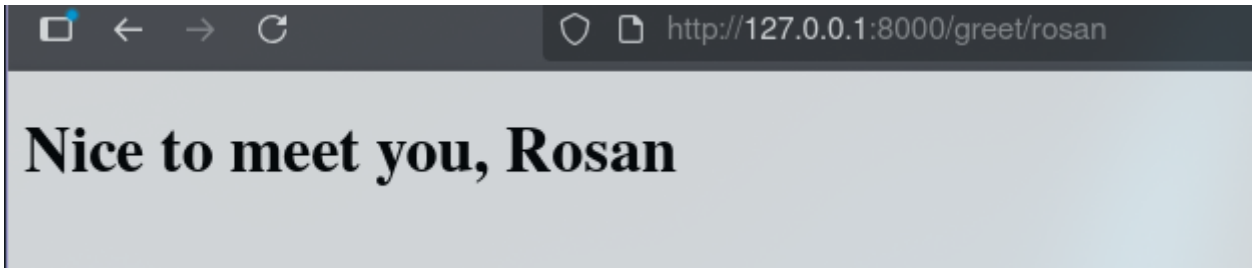
2. Hasil

Program berjalan dengan baik dan menampilkan data yang dikirim oleh *controller* ke *view*. Berikut tampilan dari hasil praktikum 1:

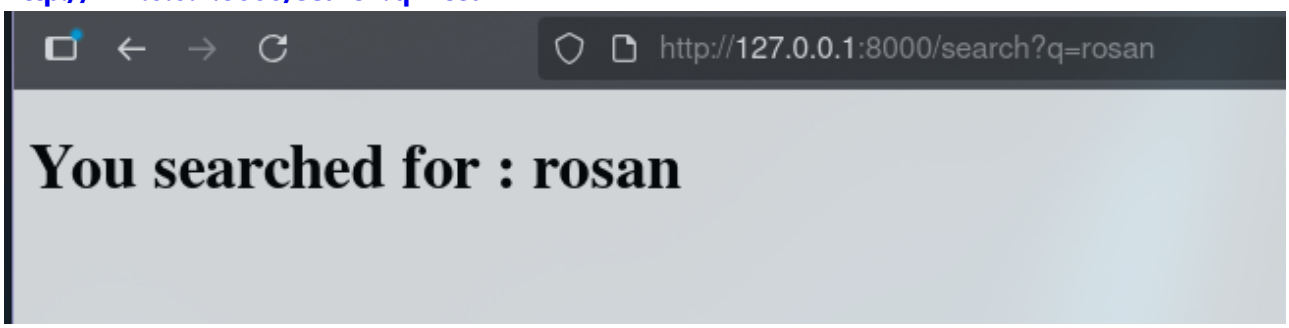
- **http://127.0.0.1:8000/hello**



- **http://127.0.0.1:8000/greet/rosan**



- **http://127.0.0.1:8000/search?q=rosan**



2.2 Praktikum 2 – Group Route

1. Langkah-Langkah

a. Membuat Project Baru

```
laravel new lab-group
```

Perintah tersebut akan membuat proyek baru dengan folder bernama *lab-group*.

b. Membuat PageController

```
php artisan make:controller PageController
```

Berikut isi dari *PageController.php*:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PageController extends Controller
{
    public function home(){
        $message = "Welcome to Homepage.";
        return view('pages.home', compact('message'));
    }

    public function about(){
        $message = "This is the about page.";
        return view('pages.about', compact('message'));
    }

    public function contact(){
        $message = "Reach us throught the contact page.";
        return view('pages.contact', compact('message'));
    }
}
```

Controller ini memiliki tiga metode:

- home()
- about()
- contact()

c. Mendefinisikan Route

```php
<?php

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

Route::controller(PageController::class)->group(function (){
    Route::get('/', 'home')->name('home');
    Route::get('/about', 'about')->name('about');
    Route::get('/contact', 'contact')->name('contact');
});
```

Route ini mengarah ke metode pada *PageController.php*. Dengan menggunakan *group routing*, kode menjadi lebih rapi tanpa penulisan ulang *PageController* untuk setiap metode karena sudah ditangani oleh *group* tersebut.

d. Membuat View

- **resources/views/pages/home.blade.php**

```blade
<!DOCTYPE html>
<html>

<head>
    <title>Home</title>
</head>

<body>
    <h1>{{ $message }}</h1>
</body>
</html>
```

- **resources/views/pages/about.blade.php**

```blade
<!DOCTYPE html>
<html>

<head>
    <title>About</title>
</head>

<body>
    <h1>{{ $message }}</h1>
</body>
</html>
```

- **resources/views/pages/contact.blade.php**

```blade
<!DOCTYPE html>
<html>

<head>
    <title>Contact</title>
</head>

<body>
    <h1>{{ $message }}</h1>
</body>
</html>
```

e. Menjalankan Program

```
php artisan serve
```

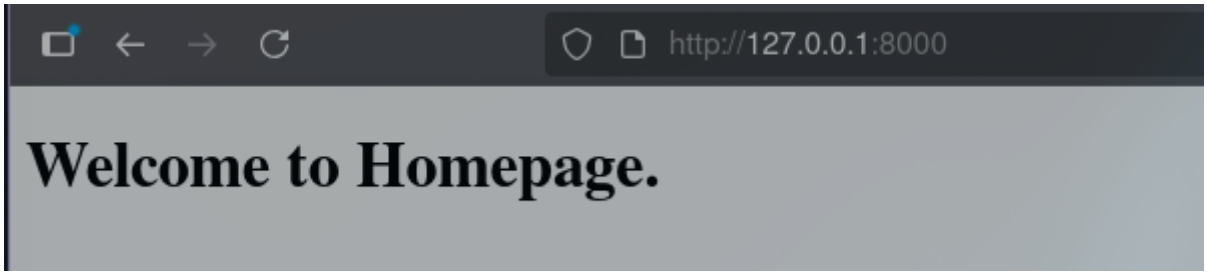
Server akan berjalan di:

```
http://127.0.0.1:8000
```

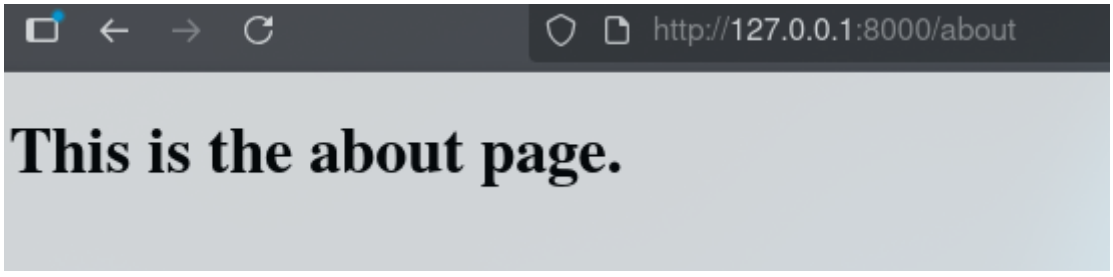
2. Hasil

Program berhasil dijalankan dan menampilkan halaman *home*, *about*, serta *contact* menggunakan *route group*. Berikut tampilannya:

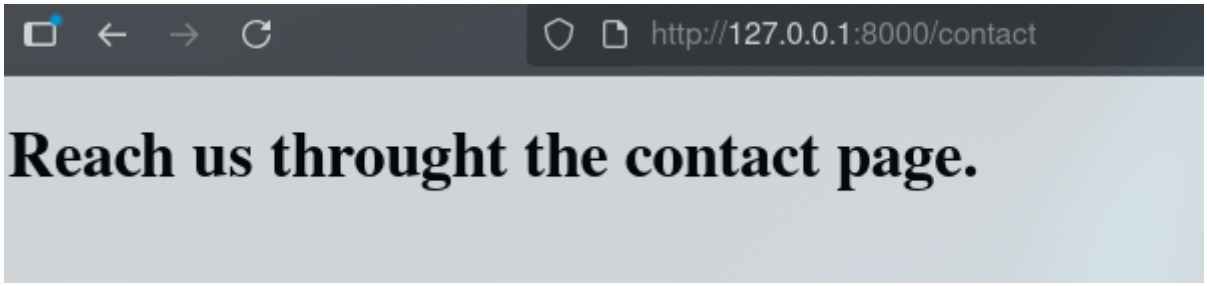
- **http://127.0.0.1:8000/** *(Home)*



- **http://127.0.0.1:8000/about** *(About)*



- **http://127.0.0.1:8000/contact** *(Contact)*



2.3 Praktikum 3 – Pengelompokan Prefix dengan Namespace Rute di Laravel 12

1. Langkah-Langkah

a. Membuat Project Baru

```
laravel new lab-prefix
```

Perintah tersebut akan membuat proyek baru dengan folder bernama *lab-prefix*.

b. Membuat Dua Controller dengan Namespace Admin

```
php artisan make:controller Admin/DashboardController
php artisan make:controller Admin/UserController
```

Berikut isi dari kedua controller tersebut:

- **App/Http/Controllers/Admin/DashboardController.php**

```php
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class DashboardController extends Controller
{
    public function index(){
        return view('admin.dashboard', ['message' => 'Welcome to Admin Dashboard']);
    }
}
```

Controller ini memiliki satu metode yang mengembalikan *message* ke *view*.

- **App/Http/Controllers/Admin/UserController.php**

```php
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function index(){
        $users = ['Ria', 'Lie', 'Jon'];
        return view('admin.users.index', compact('users'));
    }

    public function show($id){
        $user = "User #". $id;
        return view('admin.users.show', compact('user'));
    }
}
```

Controller ini memiliki dua metode:

  - `index()` : mengembalikan daftar nama pengguna.
  - `show($id)` : menampilkan detail pengguna berdasarkan ID.

c. Mendefinisikan Route

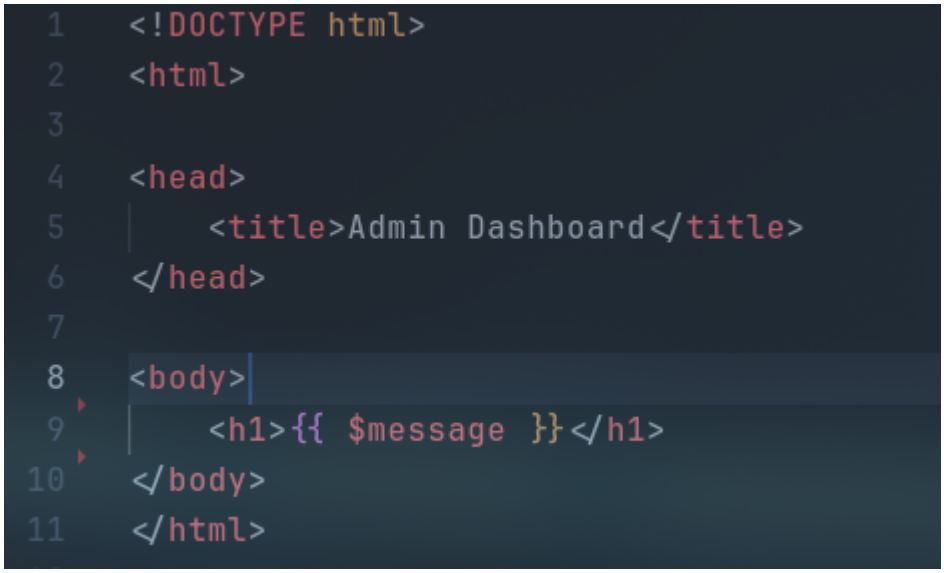
Setelah membuat controller, langkah selanjutnya adalah membuat *route* dengan prefix `admin` dan namespace `Admin`.

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Admin\DashboardController;
use App\Http\Controllers\Admin\UserController;

Route::prefix('admin')->group(function () {
    Route::get('/dashboard', [DashboardController::class, 'index'])->name('admin.dashboard');
    Route::get('/users', [UserController::class, 'index'])->name('admin.users');
    Route::get('/users/{id}', [UserController::class, 'show'])->name('admin.users.show');
});
```

Dengan menggunakan *prefix* dan *group*, beberapa *route* dengan pola URL serupa dapat dikelompokkan dalam satu struktur yang lebih rapi dan efisien.

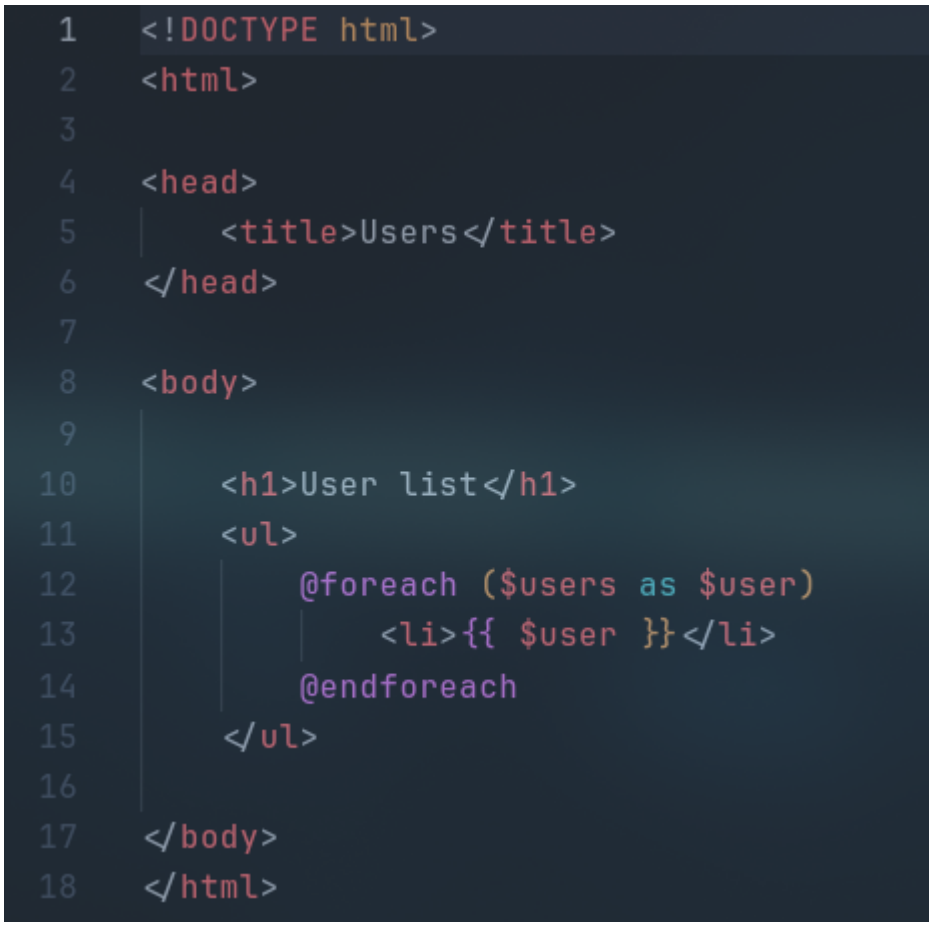
d. Membuat View

- **resources/views/dashboard.blade.php**



```
<!DOCTYPE html>
<html>

<head>
    <title>Admin Dashboard</title>
</head>

<body>
    <h1>{{ $message }}</h1>
</body>
</html>
```

- **resources/views/admin/users/index.blade.php**



```
<!DOCTYPE html>
<html>

<head>
    <title>Users</title>
</head>

<body>

    <h1>User list</h1>
    <ul>
        @foreach ($users as $user)
            <li>{{ $user }}</li>
        @endforeach
    </ul>

</body>
</html>
```

- **resources/views/admin/users/show.blade.php**

```
<!DOCTYPE html>
<html>

<head>
    <title>Contact</title>
</head>

<body>
    <h1>Detail For : {{ $user }}</h1>
</body>
</html>
```

e. Menjalankan Program

```
php artisan serve
```

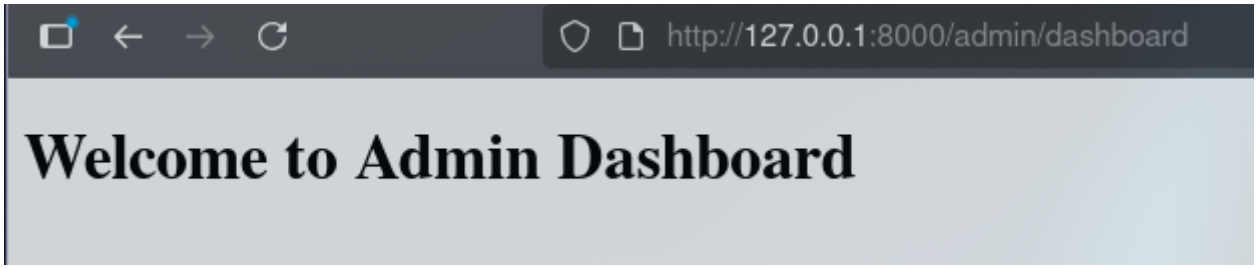
Server akan berjalan di:

```
http://127.0.0.1:8000
```

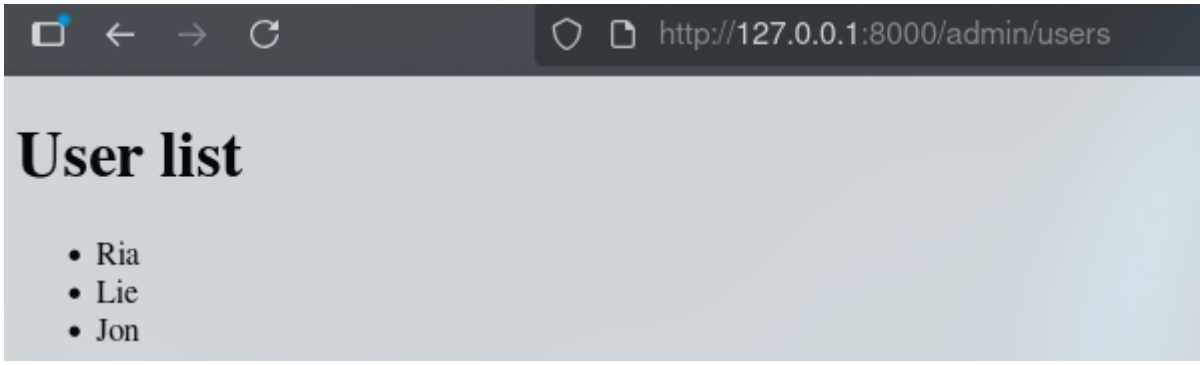
2. Hasil

Program berhasil dijalankan pada ketiga rute dengan *prefix* yang telah ditentukan, dan menampilkan hasil sesuai dengan tampilan pada halaman *dashboard*, *admin/index*, serta *admin/show*.

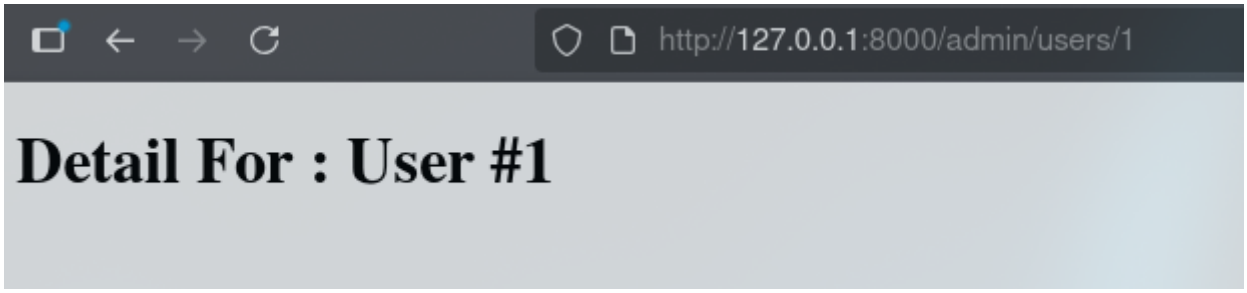
- **http://127.0.0.1:8000/admin/dashboard**



- **http://127.0.0.1:8000/admin/users**



- **http://127.0.0.1:8000/admin/users/1**



BAB III – Kesimpulan

Controller merupakan komponen utama dalam arsitektur **Model-View-Controller (MVC)** yang berfungsi sebagai pengatur alur logika aplikasi di Laravel. Melalui *controller*, data dari pengguna dapat diproses secara terstruktur sebelum dikirimkan ke tampilan (*view*).

Dari ketiga praktikum yang dilakukan — mulai dari penerapan *Basic Controller*, *Group Route*, hingga penggunaan *Prefix* dan *Namespace* — dapat disimpulkan bahwa *controller* berperan penting dalam menjaga keteraturan kode, memisahkan logika bisnis dari tampilan, serta mempermudah pengelolaan rute dalam aplikasi.

Dengan memahami berbagai jenis *controller* dan penerapan konsep *route grouping* maupun *prefixing*, pengembang dapat membangun aplikasi Laravel yang lebih efisien, terorganisir, dan mudah dikembangkan sesuai standar arsitektur MVC yang baik.

Daftar Pustaka

- **Modul 3 – Controller** — HackMD https://hackmd.io/@mohdrzu/H1sB73dnxg
- **Laravel Official Documentation — Controllers** https://laravel.com/docs/12.x/controllers
- **Laravel Resource Controllers — DigitalOcean** https://www.digitalocean.com/community/tutorials/laravel-resource-controllers