

LAPORAN PRAKTIKUM
JOBSHEET 02
ROUTING, CONTROLLER, DAN VIEW

Disusun untuk memenuhi nilai tugas
Mata Kuliah : Pemrograman Web Lanjut



Oleh :
Aqueena Regita Hapsari
2341760096
SIB 2B
03

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS
POLITEKNIK NEGERI MALANG
TAHUN AJARAN 2024/2025

1. MVC pada Laravel

MVC merupakan singkatan dari Model View Controller. Laravel menggunakan model MVC, oleh karena itu ada tiga bagian inti dari framework yang bekerja bersama: model, view, dan controller. Controller adalah bagian utama di mana sebagian besar pekerjaan dilakukan. Controller terhubung ke Model untuk mendapatkan, membuat, atau memperbarui data dan menampilkan hasilnya pada View, yang berisi struktur HTML aktual dari aplikasi.

a. Model

Dalam Laravel, kelas Model berisi semua metode dan atribut yang diperlukan untuk berinteraksi dengan skema database yang ditentukan.

b. View

View mewakili bagaimana informasi ditampilkan, digunakan untuk semua logika antarmuka pengguna perangkat lunak. View mewakili Antarmuka Pengguna (Frontend) dari halaman web.

c. Controller

Controller berperan sebagai perantara antara Model dan View, memproses semua masukan yang dikirim oleh pengguna dari View. Controller memproses semua logika bisnis, memanipulasi data menggunakan komponen Model, dan berinteraksi dengan View untuk merender output akhir.

2. Routing

Pada Laravel terdapat fitur yang bernama *route*. Route ini digunakan sebagai penghubung antara user dengan aplikasi. Dengan kata lain, URL yang kita tulis di dalam browser akan melewati route. Dan pada route tersebut akan ditentukan kemana selanjutnya, bisa ke Controller atau ke View.

Routing sendiri adalah proses pengiriman data maupun informasi ke pengguna melalui sebuah permintaan yang dilakukan kepada alamat yang sudah terdaftar, lalu alamat tersebut akan memproses dari permintaan kita tadi. Setelah proses selesai maka akan mengembalikan sebuah output atau hasil dari proses tersebut.

Untuk membuat route digunakan **Facade Route** diikuti dengan verb yang merupakan **HTTP verb**, umumnya terdiri dari **get, post, put, delete, options, patch**. Selain itu dibutuhkan **path** yang berupa URL setelah nama domain aplikasi yang diakses oleh pengguna. Dan pada bagian akhir terdapat **callback** yang dapat berupa **callback function** atau **controller action** yang menjalankan logika ketika path diakses oleh pengguna.

Berikut contoh sederhana penulisan route di Laravel 10.

```
use Illuminate\Support\Facades\Route;

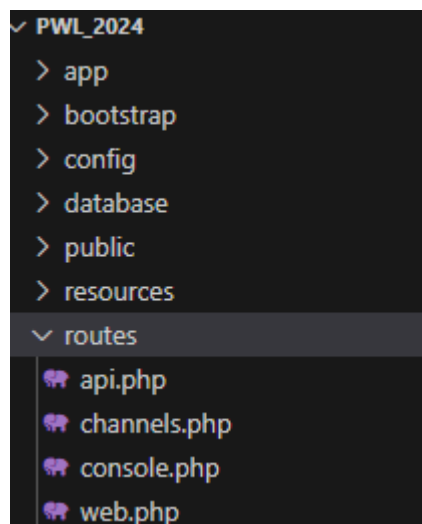
Route::get('/hello', function () {
    return 'Hello World';
});
```

Route di atas dapat diterjemahkan ketika pengguna mengakses URL pada **/hello** akan mengeksekusi callback function yang menampilkan pesan 'Hello World'.

Akan tetapi penggunaan callback function jarang sekali dipakai dalam pembuatan aplikasi sesungguhnya, karena untuk logika yang kompleks menjadikan kode susah di-maintenance. Sebagai solusi diperkenalkan konsep Controller. Jika route di atas dikonversi ke controller menjadi sebagai berikut:

```
use Illuminate\Support\Facades\Route;

Route::get('/hello', [WelcomeController::class, 'hello']);
```



Di dalam project Laravel, terdapat folder **routes**. Secara umum laravel membagi menjadi empat tempat, yaitu:

- routes/web.php digunakan untuk web standard
- routes/api.php digunakan untuk web service/API
- routes/console.php digunakan untuk command line
- routes/channel.php digunakan untuk broadcast channel melalui websocket

Secara umum aplikasi yang dibuat cukup dengan routes/web.php dan routes/api.php. Bahkan jika aplikasi tidak perlu menyediakan API hanya menggunakan routes/web.php saja.

Pada Laravel kita dapat menggunakan semua http verb untuk dipasang sebagai method router yang ingin digunakan, sudah dijelaskan sebelumnya bahwa semua http verb dapat dilayani dengan Router pada laravel. Endpoint / url router sebaiknya mengikuti best practice berikut ini dimana sebuah resource dapat dilayani dengan fungsi berbeda pada setiap http verb nya.

Resource	POST	GET	PUT	DELETE
/mahasiswa	Membuat record mahasiswa baru	Mengambil Daftar Mahasiswa	Update banyak data mahasiswa	Delete banyak data mahasiswa
/mahasiswa/{id}	Error	Tampilkan Data Satu Mahasiswa	Update data mahasiswa jika ada data dengan id yang dikirim	Delete satu data mahasiswa

Perlu diketahui laravel dapat mendukung satu route yang memiliki lebih dari satu http verb atau memiliki semua http verb. Berikut ini kode program routing untuk tabel di atas

```
Route::get('mahasiswa', function ($id) {
});
Route::post('mahasiswa', function ($id) {
});
Route::put('mahasiswa', function ($id) {
});
Route::delete('mahasiswa', function ($id) {
});
Route::get('mahasiswa/{id}', function ($id) {
});
Route::put('mahasiswa/{id}', function ($id) {
});
```

Untuk memeriksa dan memvalidasi apakah route yang dibuat sudah benar dengan menggunakan perintah berikut

```
php artisan route:list
```

Output dari perintah tersebut jika routing yang anda buat benar akan menjadi seperti ini

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	web
	GET HEAD	api/user		Closure	api auth:api
	GET HEAD	mahasiswa		Closure	web
	POST	mahasiswa		Closure	web
	PUT	mahasiswa		Closure	web
	DELETE	mahasiswa		Closure	web
	GET HEAD	mahasiswa/{id}		Closure	web
	PUT	mahasiswa/{id}		Closure	web
	DELETE	mahasiswa/{id}		Closure	web

Jika anda membutuhkan route yang dapat memiliki lebih dari satu http method routing nya dapat dibuat dengan cara seperti ini.

```
Route::match(['get', 'post'], '/specialUrl', function () {
    });

Route::any('/specialMahasiswa', function ($id) {
    });
```

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	web
	GET HEAD	api/user		Closure	api auth:api
	GET HEAD POST PUT PATCH DELETE OPTIONS	specialMahasiswa		Closure	web
	GET POST HEAD	specialUrl		Closure	web

- Basic Routing

Pada dasarnya Routing di Laravel membutuhkan informasi mengenai http verb kemudian input berupa url dan apa yang harus dilakukan ketika menerima url tersebut. Untuk membuat sebuah route anda dapat menggunakan callback function atau menggunakan sebuah controller.

Langkah-langkah Praktikum:

- Pada bagian ini, kita akan membuat dua buah route dengan ketentuan sebagai berikut.

No	Http Verb	Url	Fungsi
1	get	/hello	Tampilkan String Hello ke browser.
2	get	/world	Tampilkan String World ke browser

Kita akan menggunakan project minggu sebelumnya yaitu PWL_2024.

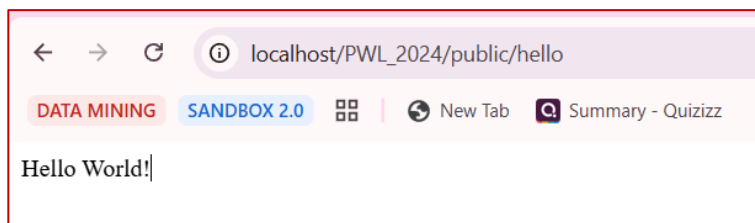
- b. Buka file `routes/web.php`. Tambahkan sebuah route untuk nomor 1 seperti di bawah ini:

```
use Illuminate\Support\Facades\Route;

Route::get('/hello', function () {
    return 'Hello World';
});
```

- c. Buka browser, tuliskan URL untuk memanggil route tersebut: `localhost/PWL_2024/public/hello`. Perhatikan halaman yang muncul apakah sudah sesuai dan jelaskan pengamatan Anda.

Hasil : Ketika pengguna mengakses `http://localhost/PWL_2024/public/hello`, Laravel akan menjalankan kode di dalam fungsi `return 'Hello World'`;, sehingga halaman akan menampilkan teks "Hello World".



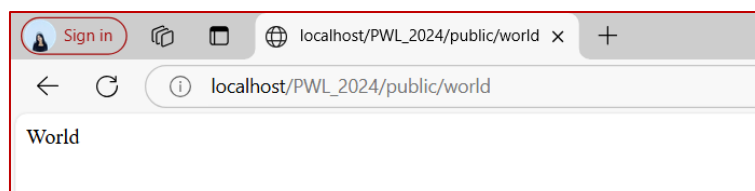
- d. Untuk membuat route kedua, tambahkan route `/world` seperti di bawah ini:

```
use Illuminate\Support\Facades\Route;

Route::get('/world', function () {
    return 'World';
});
```

- e. Bukalah pada browser, tuliskan URL untuk memanggil route tersebut: `localhost/PWL_2024/public/world`. Perhatikan halaman yang muncul apakah sudah sesuai dan jelaskan pengamatan Anda.

Hasil : mengakses `http://localhost/PWL_2024/public/world`, maka browser akan menampilkan teks "World".

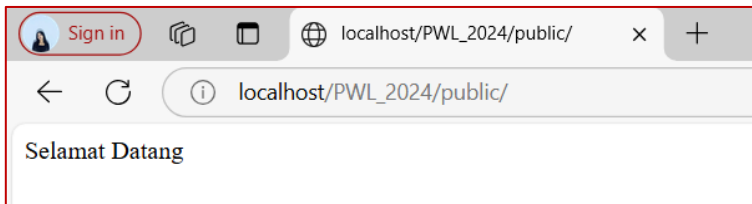


- f. Selanjutnya, cobalah membuat route '/' yang menampilkan pesan 'Selamat Datang'.

- **Hasil :**

```
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return 'Selamat Datang';
});
```



- a. Kemudian buatlah route '/about' yang akan menampilkan NIM dan nama Anda.

Hasil : Route ini akan menampilkan informasi nama dan NIM di browser.

```
<?php

use Illuminate\Support\Facades\Route;

Route::get('/about', function () {
    return 'Halo Semuanya, Nama Saya Aqueena Regita
    Hapsari. Saya adalah Mahasiswa Jurusan Teknologi
    Informasi Politeknik Negeri Malang, NIM saya
    2341760096';
});
```



- Route Parameters

Terkadang saat membuat sebuah URL, kita perlu mengambil sebuah parameter yang merupakan bagian dari segmen URL dalam route kita. Misalnya, kita membutuhkan nama user yang dikirim melalui sebuah URL.

Langkah-langkah Praktikum:

Untuk membuat routing dengan parameter dapat dilakukan dengan cara berikut ini.

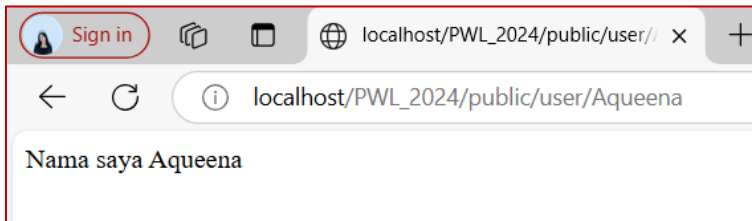
- a. Kita akan memanggil route `/user/{name}` sekaligus mengirimkan parameter berupa nama user `$name` seperti kode di bawah ini.

```
Route::get('/user/{name}', function ($name) {
    return 'Nama saya '.$name;
});
```

- b. Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: **localhost/PWL_2024/public/user>NamaAnda**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Hasil : {name} adalah variabel yang akan diisi dengan nilai dari URL.

Jika mengakses http://localhost/PWL_2024/public/user/Aqueena, browser akan menampilkan:



- c. Selanjutnya, coba tuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

Hasil : **ERROR 404**, karena parameter {name} tidak diisi

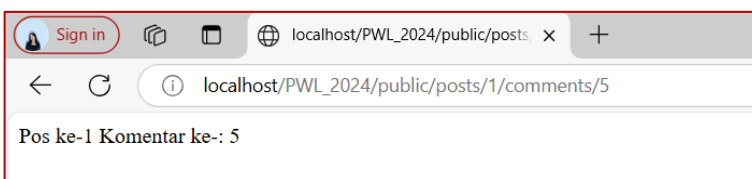
- d. Suatu route, juga bisa menerima lebih dari 1 parameter seperti kode berikut ini. Route menerima parameter \$postId dan juga \$comment.

```
Route::get('/posts/{post}/comments/{comment}', function ($postId, $commentId) {  
    return 'Pos ke-' . $postId . " Komentar ke-: " . $commentId;  
});
```

- e. Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: **localhost/PWL_2024/public/posts/1/comments/5**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Hasil : Route ini membutuhkan dua parameter dari URL.

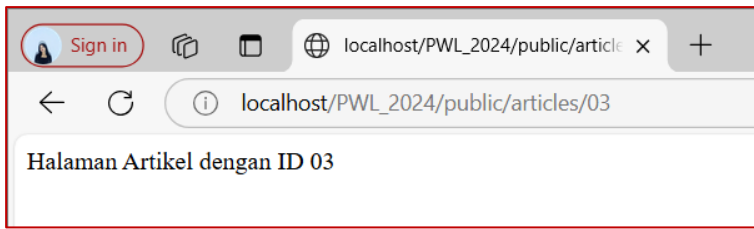
Jika kita mengakses http://localhost/PWL_2024/public/posts/1/comments/5, browser akan menampilkan:



- f. Kemudian buatlah route `/articles/{id}` yang akan menampilkan output “Halaman Artikel dengan ID {id}”, ganti id sesuai dengan input dari url.

Hasil :

```
use Illuminate\Support\Facades\Route;  
  
Route::get('/articles/{id}', function ($id) {  
    return 'Halaman Artikel dengan ID ' . $id;  
});
```

- Optional Parameters

Kita dapat menentukan nilai parameter route, tetapi menjadikan nilai parameter route tersebut opsional. Pastikan untuk memberikan variabel yang sesuai pada route sebagai nilai default. Parameter opsional diberikan tanda “?”.

Langkah-langkah Praktikum:

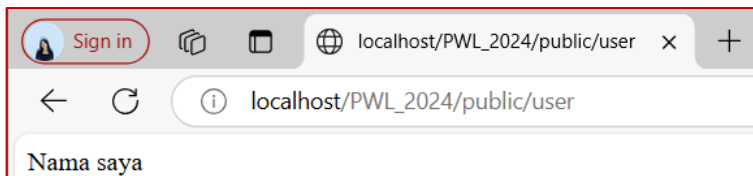
Untuk membuat routing dengan optional parameter dapat dilakukan dengan cara berikut ini.

- Kita akan memanggil route `/user` sekaligus mengirimkan parameter berupa nama user `$name` dimana parameternya bersifat opsional.

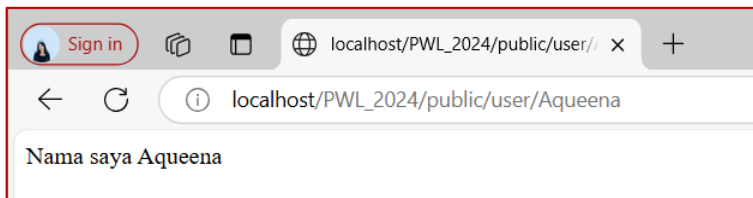
```
Route::get('/user/{name?}', function ($name=null) {  
    return 'Nama saya '.$name;  
});
```

- Jalankan kode dengan menuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Hasil :



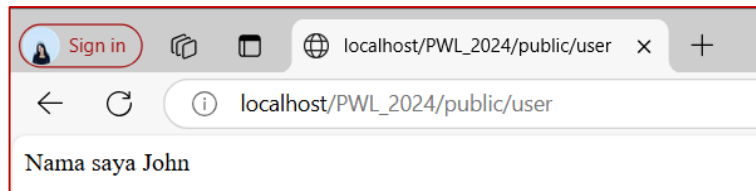
- Selanjutnya tuliskan URL: **localhost/PWL_2024/public/user>NamaAnda**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda



- Ubah kode pada route `/user` menjadi seperti di bawah ini.

```
Route::get('/user/{name?}', function ($name='John') {  
    return 'Nama saya '.$name;  
});
```

- Jalankan kode dengan menuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



- Route Name

Route name biasanya digunakan untuk mempermudah kita dalam pemanggilan route saat membangun aplikasi. Kita cukup memanggil name dari route tersebut.

```
Route::get('/user/profile', function () {
    //
})->name('profile');

Route::get(
    '/user/profile',
    [UserProfileController::class, 'show']
)->name('profile');

// Generating URLs...
$url = route('profile');

// Generating Redirects...
return redirect()->route('profile');
```

- Route Group dan Route Prefixes

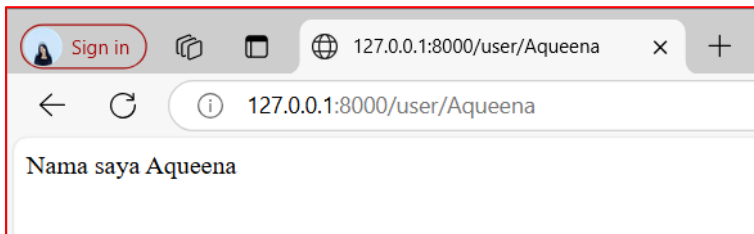
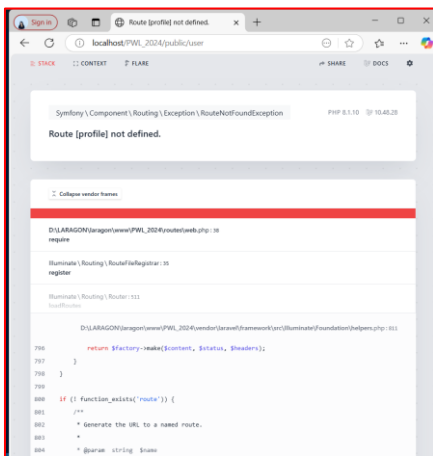
Beberapa route yang memiliki atribut yang sama seperti middleware yang sama dapat dikelompokkan menjadi satu kelompok untuk mempermudah penulisan route selain

- a. Jalankan kode dengan menuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Hasil : error karena sebelumnya tidak ada route untuk UserProfileController

- b. Selanjutnya tuliskan URL: **localhost/PWL_2024/public/user>NamaAnda**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

Hasil : error karena sebelumnya tidak ada route untuk UserProfileController, kemudian saya tambahkan controller baru yakni UserProfileController kemudian berhasil di defined

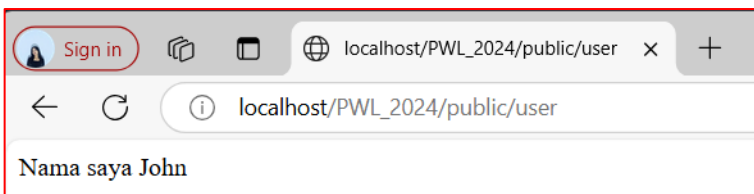


- c. Ubah kode pada route /user menjadi seperti di bawah ini.

```
Route::get('/user/{name?}', function ($name='John') {  
    return 'Nama saya '.$name;  
});
```

- d. Jalankan kode dengan menuliskan URL: **localhost/PWL_2024/public/user/**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Hasil : Setelah saya hilangkan generate URL dan generate redirect, bisa diakses seperti dibawah ini



- Route Group dan Route Prefixes

digunakan untuk middleware masih ada lagi penggunaan route group untuk route yang berada dibawah satu subdomain. Contoh penggunaan route group adalah sebagai berikut:

```
Route::middleware(['first', 'second'])->group(function () {
    Route::get('/', function () {
        // Uses first & second middleware...
    });

    Route::get('/user/profile', function () {
        // Uses first & second middleware...
    });
});

Route::domain('{account}.example.com')->group(function () {
    Route::get('user/{id}', function ($account, $id) {
        //
    });
});

Route::middleware('auth')->group(function () {
    Route::get('/user', [UserController::class, 'index']);
    Route::get('/post', [PostController::class, 'index']);
    Route::get('/event', [EventController::class, 'index']);
});
```

Route Prefixes

Pengelompokan route juga dapat dilakukan untuk route yang memiliki prefix (awalan) yang sama. Untuk pembuatan route dengan prefix dapat dilihat kode seperti di bawah ini

```
Route::prefix('admin')->group(function () {
    Route::get('/user', [UserController::class, 'index']);
    Route::get('/post', [PostController::class, 'index']);
    Route::get('/event', [EventController::class, 'index']);
});
```

- Redirect Routes

Untuk melakukan redirect pada laravel dapat dilakukan dengan menggunakan Route::redirect cara penggunaannya dapat dilihat pada kode program dibawah ini.

```
Route::redirect('/here', '/there');
```

Redirect ini akan sering digunakan pada kasus kasus CRUD atau kasus lain yang

- View Routes

Laravel juga menyediakan sebuah route khusus yang memudahkan dalam membuat sebuah routes tanpa menggunakan controller atau callback function. Routes ini langsung menerima input berupa url dan mengembalikan view / tampilan. Berikut ini cara membuat view routes.

```
Route::view('/welcome', 'welcome');  
Route::view('/welcome', 'welcome', ['name' => 'Taylor']);
```

Pada view routes diatas /welcome akan menampilkan view welcome dan pada route kedua /welcome akan menampilkan view welcome dengan tambahan data berupa variabel name.

Simpan perubahan yang telah dilakukan pada Git.

3. Controller

Controller digunakan untuk mengorganisasi logika aplikasi menjadi lebih terstruktur. Logika action aplikasi yang masih ada kaitan dapat dikumpulkan dalam satu kelas Controller. Atau sebuah Controller dapat juga hanya berisi satu buah action. Controller pada Laravel disimpan dalam folder `app/Http/Controllers`.

- Membuat Controller

Langkah-langkah Praktikum:

- Untuk membuat controller pada Laravel telah disediakan perintah untuk menggenerate struktur dasarnya. Kita dapat menggunakan perintah artisan diikuti dengan definisi nama controller yang akan dibuat.

```
php artisan make:controller WelcomeController
```

- Buka file pada `app/Http/Controllers/WelcomeController.php`. Struktur pada controller dapat digambarkan sebagai berikut:

```
<?php  
  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;
```

```
class WelcomeController extends Controller
{
    //
}
```

- c. Untuk mendefinisikan action, silahkan tambahkan function dengan access public. Sehingga controller di atas menjadi sebagai berikut:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class WelcomeController extends Controller
{
    public function hello() {
        return 'Hello World';
    }
}
```

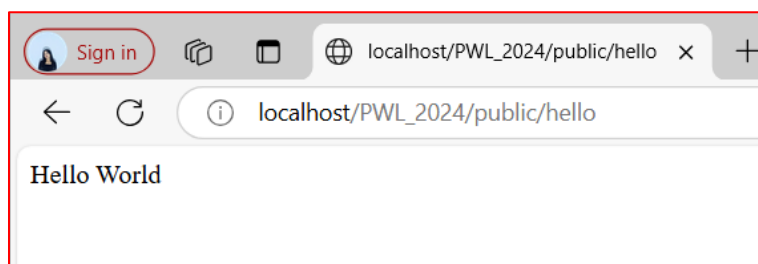
- d. Setelah sebuah controller telah didefinisikan action, kita dapat menambahkan controller tersebut pada route. Ubah route /hello menjadi seperti berikut:

```
Route::get('/hello', [WelcomeController::class, 'hello']);
```

- e. Buka browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL_2024/public/hello. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Hasil : use Illuminate\Http\Request mengimpor class request digunakan jika ingin menangani input dari pengguna(tapi disini tidak dipakai). Kemudian membuat controller bernama WelcomeController yang tujuannya menangani permintaan (request) dari pengguna. Method hello() mengembalikan teks “Hello World” sebagai respon pada saat route /hello diakses.

use App\Http\Controllers\WelcomeController; buat mengimpor controller, kemudian memanggil route /hello dan menjalankan method hello().

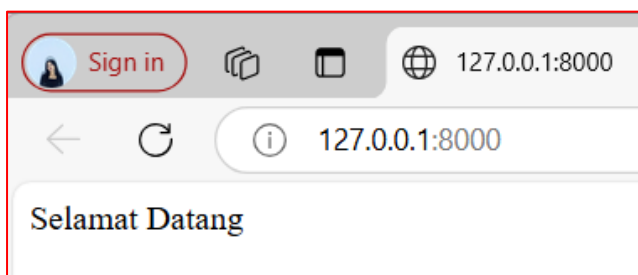


- f. Modifikasi hasil pada praktikum poin 2 (Routing) dengan konsep controller. Pindahkan logika eksekusi ke dalam controller dengan nama PageController.

Resource	POST	GET	PUT	DELETE
/		Tampilkan Pesan 'Selamat Datang' PageController : index		
/about		Tampilkan Nama dan NIM PageController : about		
/articles/{id}		Tampilkan halaman dinamis 'Halaman Artikel dengan Id {id}' id diganti sesuai input dari url		

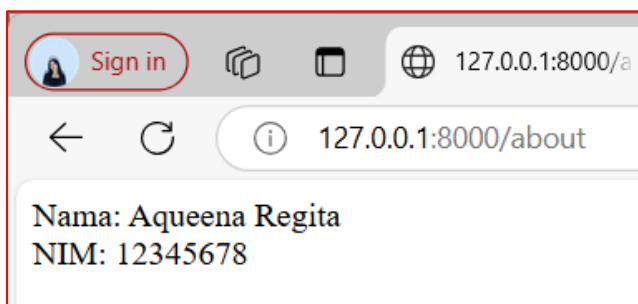
Hasil :

Resource '/'



```
public function index()
{
    return 'Selamat Datang';
}
```

Resource /about



```
public function about()
{
    return 'Nama: Aqueena Regita <br> NIM: 12345678';
}
```

Resource /articles/(id)



```
public function articles($id)
{
    return "Halaman Artikel dengan ID: " . $id;
}
```

		PageController : articles		
--	--	---------------------------	--	--

- g. Modifikasi kembali implementasi sebelumnya dengan konsep Single Action Controller. Sehingga untuk hasil akhir yang didapatkan akan ada HomeController, AboutController dan ArticleController. Modifikasi juga route yang digunakan.

- Resource Controller

Khusus untuk controller yang terhubung dengan Eloquent model dan dapat dilakukan operasi CRUD terhadap model Eloquent tersebut, kita dapat membuat sebuah controller yang bertipe Resource Controller. Dengan membuat sebuah resource controller, maka controller tersebut telah dilengkapi dengan method-method yang mendukung proses CRUD, serta terdapat sebuah route resource yang menampung route untuk controller tersebut.

Langkah-langkah Praktikum:

- a. Untuk membuatnya dilakukan dengan menjalankan perintah berikut ini di terminal.

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

Perintah ini akan generate sebuah controller dengan nama PhotoController yang berisi method method standar untuk proses CRUD.

- b. Setelah controller berhasil degenerate, selanjutnya harus dibuatkan route agar dapat terhubung dengan frontend. Tambahkan kode program berikut pada file web.php.

```
use App\Http\Controllers\PhotoController;

Route::resource('photos', PhotoController::class);
```

- c. Jalankan cek list route (php artisan route:list) akan dihasilkan route berikut ini.

Domain	Method	URI	Name	Action	Middleware
	GET HEAD	/		Closure	web
	GET HEAD	api/user		Closure	api
	POST	photos	photos.index	App\Http\Controllers\PhotoController@index	web
	GET HEAD	photos/create	photos.store	App\Http\Controllers\PhotoController@store	web
	GET HEAD	photos/{photo}	photos.create	App\Http\Controllers\PhotoController@create	web
	PUT PATCH	photos/{photo}	photos.show	App\Http\Controllers\PhotoController@show	web
	DELETE	photos/{photo}	photos.update	App\Http\Controllers\PhotoController@update	web
	GET HEAD	photos/{photo}/edit	photos.destroy	App\Http\Controllers\PhotoController@destroy	web
	GET HEAD POST PUT PATCH DELETE OPTIONS	specialMahasiswa	photos.edit	App\Http\Controllers\PhotoController@edit	web
	GET POST HEAD	specialUrl		Closure	web
				Closure	web

Hasil :


```
PS D:\php artisan route:list

GET|HEAD / ..... PageController@index
POST ignition/execute-solution ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSo...
GET|HEAD ignition/health-check ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckContro...
POST ignition/update-config ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigCon...
GET|HEAD about ..... PageController@about
GET|HEAD api/user .....
GET|HEAD articles/{id} ..... PageController@articles
GET|HEAD hello ..... WelcomeController@hello
GET|HEAD photos ..... photos.index > PhotoController@index
POST photos ..... photos.store > PhotoController@store
GET|HEAD photos/create ..... photos.create > PhotoController@create
GET|HEAD photos/{photo} ..... photos.show > PhotoController@show
PUT|PATCH photos/{photo} ..... photos.update > PhotoController@update
DELETE photos/{photo} ..... photos.destroy > PhotoController@destroy
GET|HEAD photos/{photo}/edit ..... photos.edit > PhotoController@edit
GET|HEAD posts/{post}/comments/{comment} .....
GET|HEAD sanctum/csrf-cookie ... sanctum.csrf-cookie > Laravel\Sanctum > CsrfCookieController@show
GET|HEAD user/profile ..... profile > UserProfileController@show
GET|HEAD user/{name?} .....
GET|HEAD user/{name} .....
GET|HEAD welcome .....
GET|HEAD world .....
```

Showing [22] routes

- d. Pada route list semua route yang berhubungan untuk crud photo sudah di generate oleh laravel. Jika tidak semua route pada resource controller dibutuhkan dapat dikurangi dengan

```
Route::resource('photos', PhotoController::class)->only([
    'index', 'show'
]);

Route::resource('photos', PhotoController::class)->except([
    'create', 'store', 'update', 'destroy'
]);
```

Hasil : pada tampilan ini saya menggunakan pengurangan resource controller menggunakan except create store update dan destroy

```
PS D:\php artisan route:list

GET|HEAD / ..... PageController@index
POST ignition/execute-solution ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSo...
GET|HEAD ignition/health-check ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckContro...
POST ignition/update-config ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigCon...
GET|HEAD about ..... PageController@about
GET|HEAD api/user .....
GET|HEAD articles/{id} ..... PageController@articles
GET|HEAD hello ..... WelcomeController@hello
GET|HEAD photos ..... photos.index > PhotoController@index
POST photos ..... photos.store > PhotoController@store
GET|HEAD photos/create ..... photos.create > PhotoController@create
GET|HEAD photos/{photo} ..... photos.show > PhotoController@show
PUT|PATCH photos/{photo} ..... photos.update > PhotoController@update
DELETE photos/{photo} ..... photos.destroy > PhotoController@destroy
GET|HEAD photos/{photo}/edit ..... photos.edit > PhotoController@edit
GET|HEAD posts/{post}/comments/{comment} .....
GET|HEAD sanctum/csrf-cookie ... sanctum.csrf-cookie > Laravel\Sanctum > CsrfCookieController@show
GET|HEAD user/profile profile > UserProfileController@show
GET|HEAD user/{name?} .....
GET|HEAD user/{name} .....
GET|HEAD welcome .....
GET|HEAD world .....
```

Simpan perubahan yang telah dilakukan pada Git.

4. View

Dalam kerangka kerja Laravel, View merujuk pada bagian dari aplikasi web yang bertanggung jawab untuk menampilkan antarmuka pengguna kepada pengguna akhir. View pada dasarnya adalah file template yang digunakan untuk menghasilkan HTML yang akan ditampilkan kepada pengguna.

Blade merupakan templating engine bawaan Laravel. Berguna untuk mempermudah dalam menulis kode tampilan. Dan juga memberikan fitur tambahan untuk memanipulasi data di view yang dilempar dari controller. Blade juga memungkinkan penggunaan plain PHP pada kode View. Karena Laravel menggunakan *templating engine* bawaan Blade, maka setiap *file* View diakhiri dengan `.blade.php`. Misal: `index.blade.php`, `home.blade.php`, `product.blade.php`.

- Membuat View

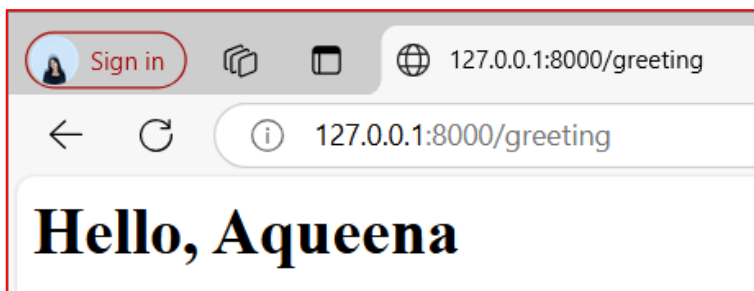
Langkah-langkah Praktikum:

- a. Pada direktori `app/resources/views`, buatlah file `hello.blade.php`.

```
<!-- View pada resources/views/hello.blade.php -->
<html>
  <body>
    <h1>Hello, {{ $name }}</h1>
  </body>
</html>
```

- b. View tersebut dapat dijalankan melalui Routing, dimana *route* akan memanggil View sesuai dengan nama *file* tanpa `'blade.php'`. (Catatan: Gantilah Andi dengan nama Anda)
- c. Jalankan code dengan membuka url `localhost/PWL_2024/public/greeting`. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Hasil : saat mengakses `/greeting` di Laravel, maka menjalankan route di `web.php` yang memanggil `view('hello')`. Kemudian dikirimkan data `name = 'Aqueena'` ke view `hello.blade.php`. nah, didalam `hello.blade.php` bagian `{{ $name }}` akan otomatis diganti menjadi Aqueena seperti dibawah ini.

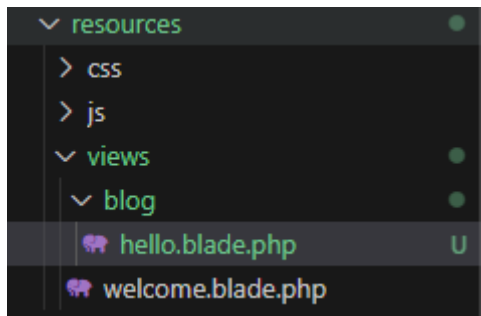


- View dalam direktori

Jika di dalam direktori `resources/views` terdapat direktori lagi untuk menyimpan *file* view, sebagai contoh `hello.blade.php` ada di dalam direktori `blog`, maka kita bisa menggunakan “dot” notation untuk mereferensikan direktori,

Langkah-langkah Praktikum:

- Buatlah direktori `blog` di dalam direktori `views`.
- Pindahkan file `hello.blade.php` ke dalam direktori `blog`.

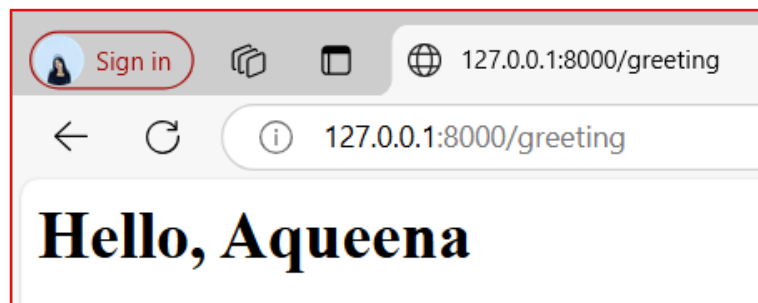


- Selanjutnya lakukan perubahan pada route.

```
Route::get('/greeting', function () {  
    return view('blog.hello', ['name' => 'Andi']);  
});
```

- Jalankan code dengan membuka url `localhost/PWL_2024/public/greeting`. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Hasil : Laravel mencari file dalam folder `blog` dan bukan langsung di `resources/views` karena menggunakan dot notation (`blog.hello`) yang berarti Laravel mencari file di `resources/views/blog/hello.blade.php`. jika sedikit view, tdk usah pake dot(.), kalau banyak view bisa dibikin struktur view berfolder-folder agar rapi dan mudah pengelolaan viewnya.



- Menampilkan View dari Controller

View dapat dipanggil melalui Controller. Sehingga Routing akan memanggil Controller terlebih dahulu, dan Controller akan me-*return* view yang dimaksud.

Langkah-langkah Praktikum:

- a. Buka WelcomeController.php dan tambahkan fungsi baru yaitu greeting.

```
class WelcomeController extends Controller
{
    public function hello(){
        return('Hello World');
    }

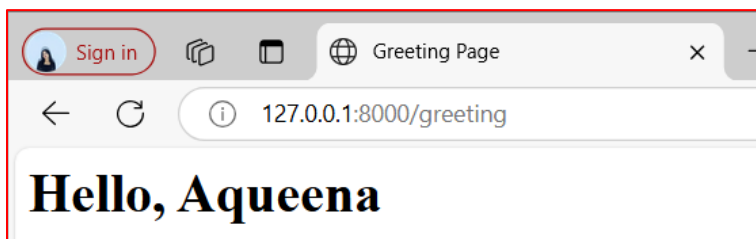
    public function greeting(){
        return view('blog.hello', ['name' => 'Andi']);
    }
}
```

- b. Ubah route /greeting dan arahkan ke WelcomeController pada fungsi greeting.

```
Route::get('/greeting', [WelcomeController::class,
'greeting']);
```

- c. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

Hasil : Laravel memanggil method greeting() di WelcomeController, kemudian method greeting() mengembalikan view blog.hello dengan variable name = 'Aqueena'. Nah, didalam hello.blade.php bagian {{ \$name }} akan digantikan dengan 'Aqueena'.



- Meneruskan data ke view

Pada contoh sebelumnya, kita dapat meneruskan data array ke view agar data tersebut tersedia untuk view:

```
return view('blog.hello', ['name' => 'Andi']);
```

Saat meneruskan informasi dengan cara ini, data harus berupa array dengan pasangan kunci / nilai. Setelah memberikan data ke view, kemudian kita dapat mengakses setiap nilai dalam view menggunakan kunci data seperti: `<?php echo $name; ?>` atau `{{ $name }}`. Sebagai alternatif untuk meneruskan array data lengkap ke fungsi view helper, kita dapat menggunakan metode **with** untuk menambahkan bagian data individual ke view. Metode **with**

mengembalikan instance view objek sehingga kita dapat melanjutkan rangkaian metode sebelum mengembalikan tampilan

notation untuk mereferensikan direktori,

Langkah-langkah Praktikum:

- a. Buka WelcomeController.php dan tambahkan ubah fungsi greeting.

```
class WelcomeController extends Controller
{
    public function hello(){
        return('Hello World');
    }

    public function greeting(){
        return('Hello World');
    }
}
```

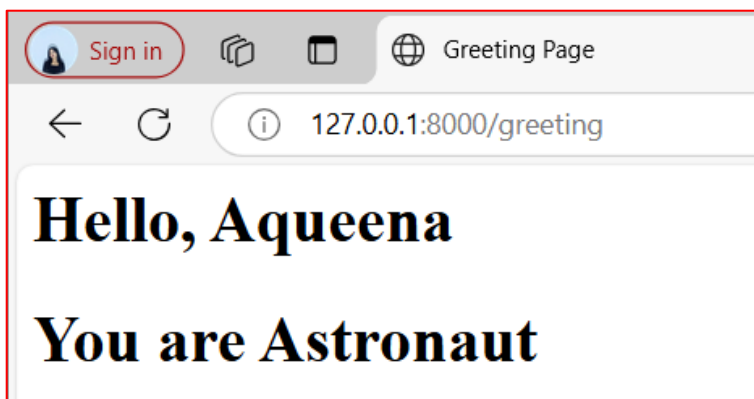
```
        ->with('name','Andi')
        ->with('occupation','Astronaut');
    }
}
```

- b. Ubah hello.blade.php agar dapat menampilkan dua parameter.

```
<html>
  <body>
    <h1>Hello, {{ $name }}</h1>
    <h1>You are {{ $occupation }}</h1>
  </body>
</html>
```

- c. Jalankan code dengan membuka url localhost/PWL_2024/public/greeting.
Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

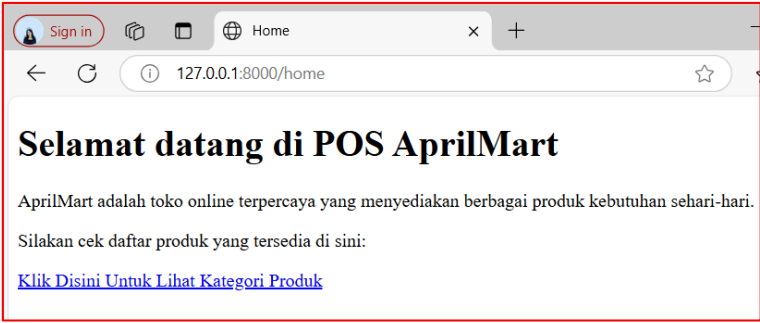
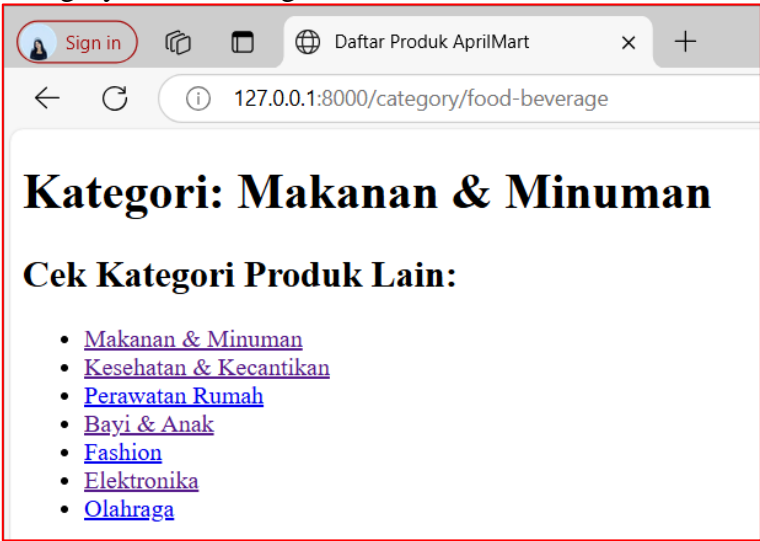
Hasil : saat mengakses /greeting, Laravel membaca route, kemudian menjalankan method greeting() di welcome controller yang kegiatannya disitu adalah mengirimkan nama Aqueena dan occupation Astronaut ke view. Kemudian mengembalikan tampilan ke blog.hello. Didalam hello.blade.php bagian {{ \$name }} dan {{ \$occupation }} akan digantikan dengan Aqueena dan Astronaut dan ditampilkan seperti dibawah ini.

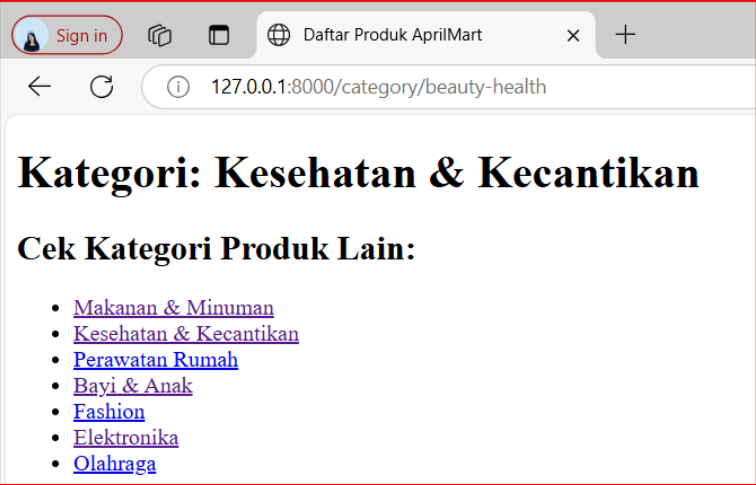
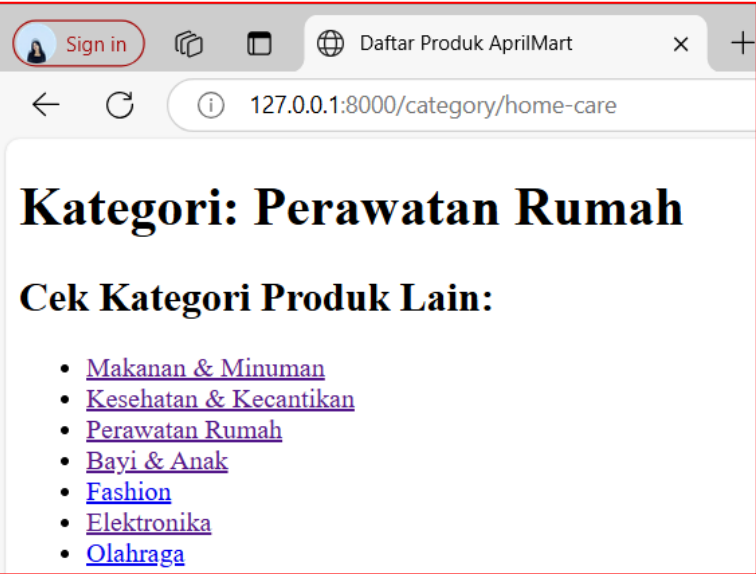
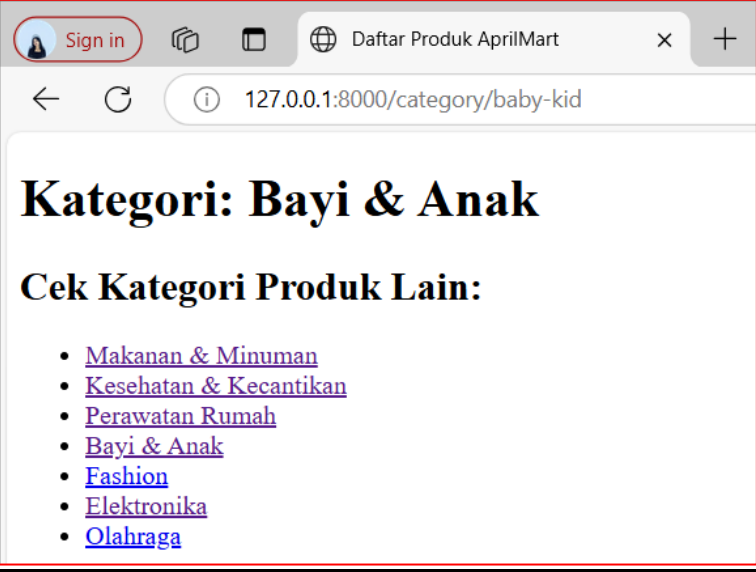


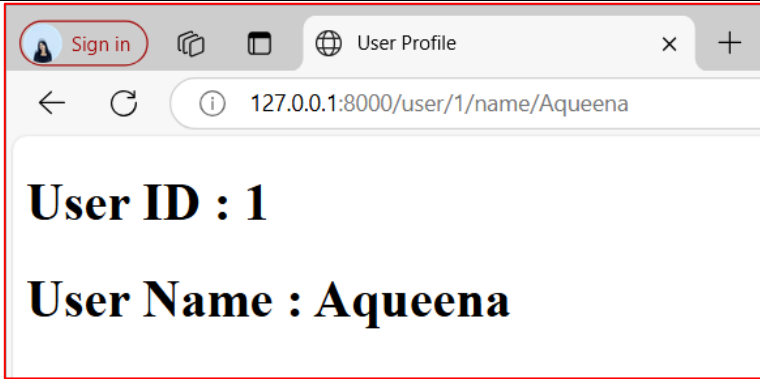
Simpan perubahan yang telah dilakukan pada Git.

SOAL PRAKTIKUM

1. Jalankan Langkah-langkah Praktikum pada jobsheet di atas. Lakukan sinkronisasi perubahan pada project PWL_2024 ke Github.
2. Buatlah project baru dengan nama POS. Project ini merupakan sebuah aplikasi Point of Sales yang digunakan untuk membantu penjualan.
3. Buatlah beberapa route, controller, dan view sesuai dengan ketentuan sebagai berikut.

1	<p>Halaman Home Menampilkan halaman awal website</p> 
2	<p>Halaman Products Menampilkan daftar product (route prefix) /category/food-beverage</p>  <p>/category/beauty-health</p>

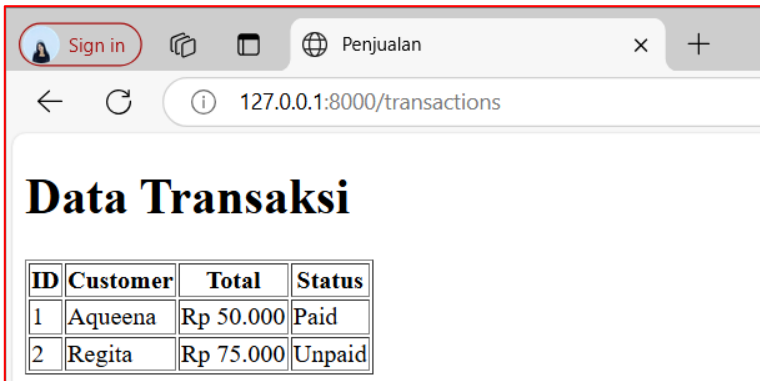
	 <p data-bbox="347 689 606 723">/category/home-care</p>  <p data-bbox="347 1303 584 1337">/category/baby-kid</p> 
3	<p data-bbox="339 1921 526 1955">Halaman User</p> <p data-bbox="339 1960 914 1993">Menampilkan profil pengguna (route param)</p> <p data-bbox="339 1998 649 2031">/user/{id}/name/{name}</p>



The screenshot shows a web browser window with the title 'User Profile'. The address bar displays '127.0.0.1:8000/user/1/name/Aqueena'. The page content features two lines of text: 'User ID : 1' and 'User Name : Aqueena'.

4 Halaman Penjualan

Menampilkan halaman transaksi POS



The screenshot shows a web browser window with the title 'Penjualan'. The address bar displays '127.0.0.1:8000/transactions'. The page content features the heading 'Data Transaksi' followed by a table of transactions.

ID	Customer	Total	Status
1	Aqueena	Rp 50.000	Paid
2	Regita	Rp 75.000	Unpaid

4. Route tersebut menjalankan fungsi pada Controller yang berbeda di setiap halaman.
5. Fungsi pada Controller akan memanggil view sesuai halaman yang akan ditampilkan.
6. Simpan setiap perubahan yang dilakukan pada project POS pada Git, sinkronisasi perubahan ke Github.

7. Link Github :

Tugas Praktikum : <https://github.com/aqwenaaa/POS-Laravel.git>

Jobsheet : <https://github.com/aqwenaaa/www.git>

(<https://github.com/aqwenaaa/www/tree/PWL-Jobsheet-2-Routering-Controller-View>)