**LAPORAN PRAKTIKUM**

**MATA KULIAH PEMROGRAMAN WEB LANJUT**

Dosen Pengampu : Dimas Wahyu Wibowo, ST., MT.

**JOBSHEET 2: ROUTING, CONTROLLING, DAN VIEW**

****

Nama : Yonanda Mayla Rusdiaty

NIM : 2341760184

Prodi : D-IV Sistem Informasi Bisnis

**JURUSAN TEKNOLOGI INFORMASI**

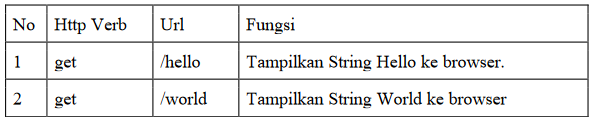
**POLITEKNIK NEGERI MALANG**

**2024**

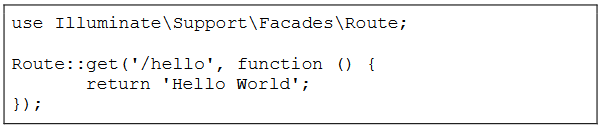
1. **ROUTING**

**PRAKTIKUM 1**

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

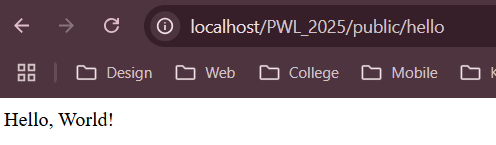


1. Buka file routes/web.php. Tambahkan sebuah route untuk nomor 1 seperti di bawah ini



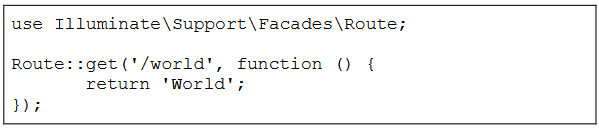
1. Buka browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL\_2024/public/hello. Perhatikan halaman yang muncul apakah sudah sesuai dan jelaskan pengamatan Anda.

**Jawab:**

****

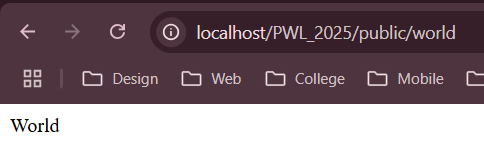
**Penjelasan:** Method Route::get('/hello', ...) akan menanggapi permintaan HTTP GET ke path '/hello'. Ketika path tersebut diakses, fungsi anonim yang didefinisikan akan dijalankan, dan dalam contoh ini, ia mengembalikan string "Hello, World!".

1. Untuk membuat route kedua, tambahkan route /world seperti di bawah ini:



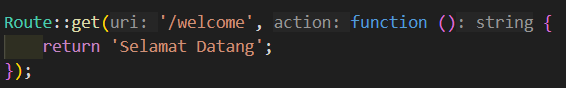
1. Bukalah pada browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL\_2024/public/world. Perhatikan halaman yang muncul apakah sudah sesuai dan jelaskan pengamatan Anda.

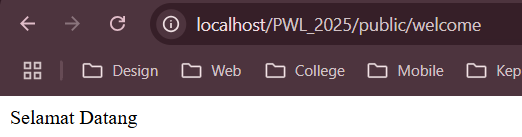
**Jawab:**

****

1. Selanjutnya, cobalah membuat route ’/’ yang menampilkan pesan ‘Selamat Datang’.

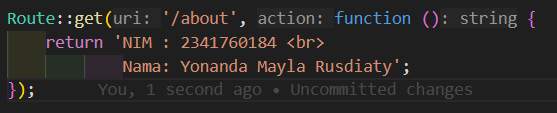
**Jawab:**

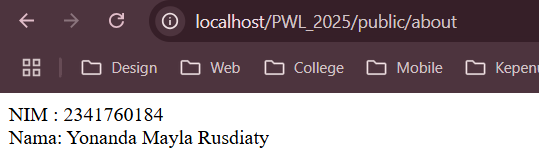
****

****

1. Kemudian buatlah route ‘/about’ yang akan menampilkan NIM dan nama Anda.

**Jawab:**

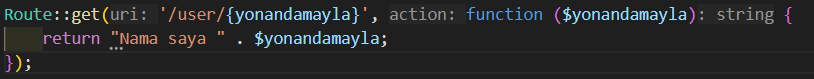
****

****

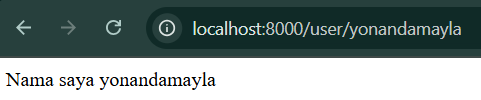
**ROUTE PARAMETERS**

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

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

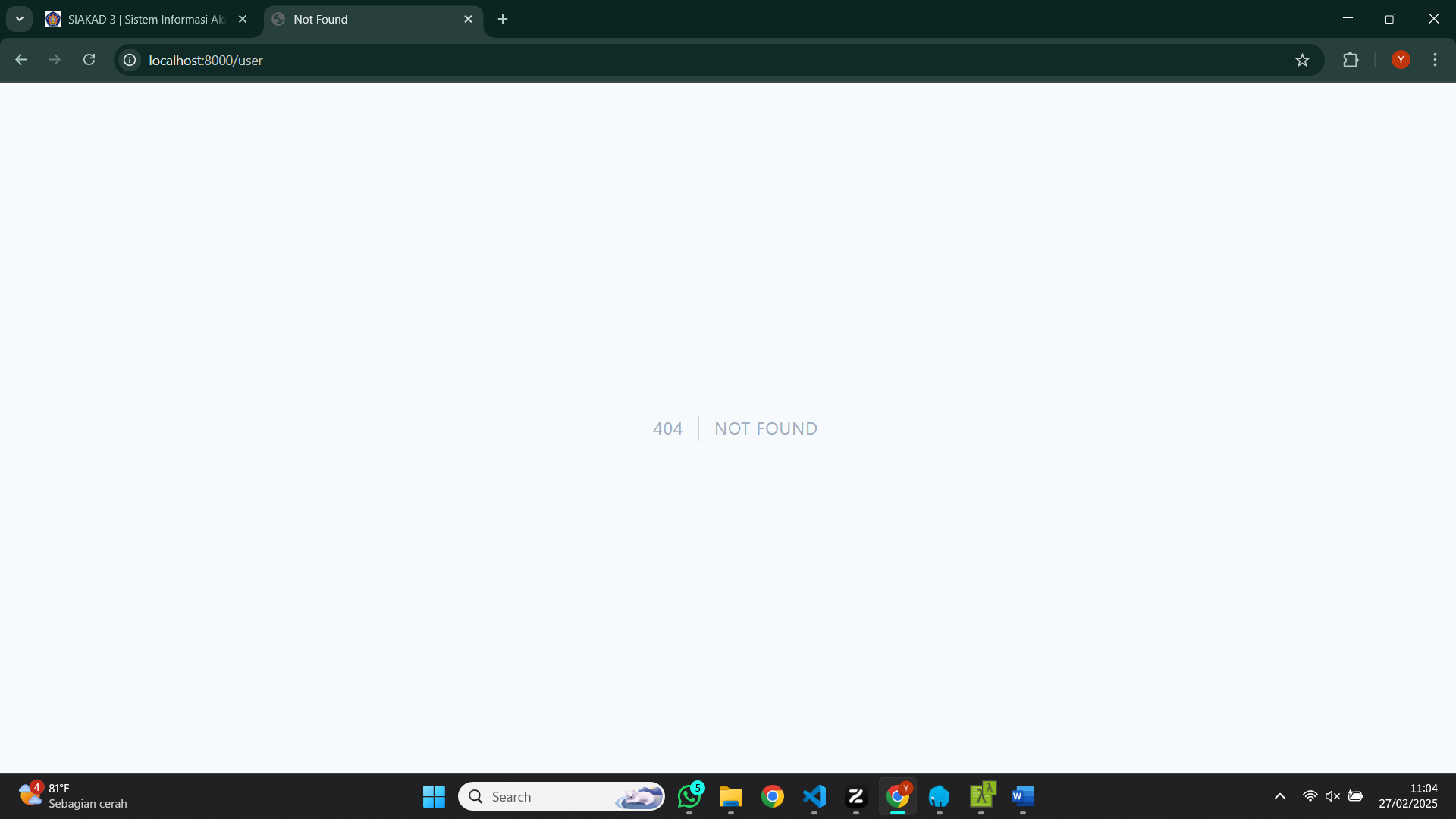


1. Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: localhost/PWL\_2024/public/user/NamaAnda. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda



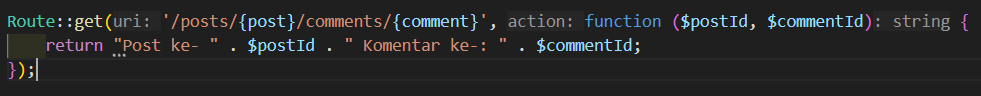
1. Selanjutnya, coba tuliskan URL: localhost/PWL\_2024/public/user/. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

**Jawab:**

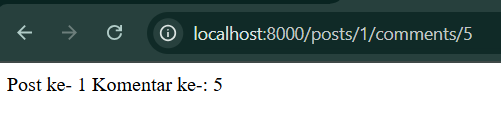


Hasilnya adalah not found karena tidak ada atau tidak ditemukan file user, dan kode yang saya tuliskan mendefinisikan parameter, jadi mengharuskan adanya route parameter.

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



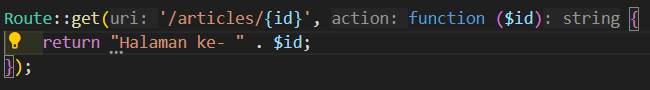
1. Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: localhost/PWL\_2024/public/posts/1/comments/5. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

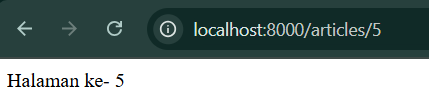
****

Jawab: route /posts/{post}/comments/{comment} dapat menerima inputan dan menampilkan nilai parameter yang diberikan di URL. Route tersebut akan menangkap dua parameter: post dan comment, dan kemudian mengembalikan string yang menggabungkan kedua parameter tersebut

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

**Jawab:**

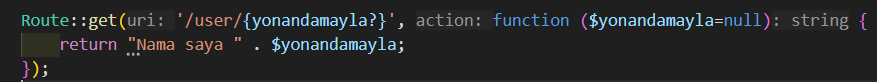
****

****

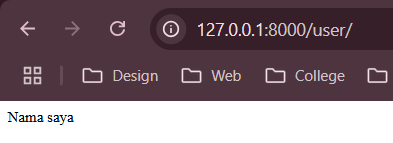
**OPTIONAL PARAMETERS**

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 ‘?’.

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

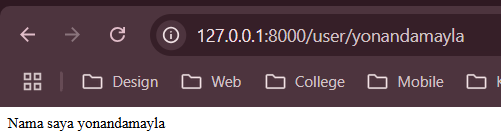


1. Jalankan kode dengan menuliskan URL: localhost/PWL\_2024/public/user/. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

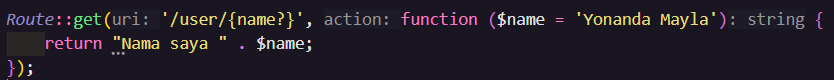


**Jawab:** Bagian {yonandamayla?} adalah parameter route. Tanda “?” menunjukkan bahwa parameter ini opsional. function ($yonandamayla=null), merupakan function yg akan dieksekusi ketika route diakses, yang berarti jika parameter tidak disediakan dalam URL, nilainya akan default ke null. Jika parameter tidak ada, maka akan mengembalikan "Nama saya " diikuti dengan kosong (karena $yonandamayla adalah null). Namun jika parameter ikut disertakan, maka returnya “Nama saya yonandamayla”

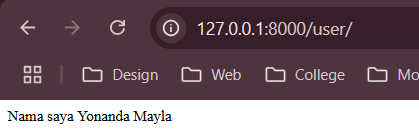
1. Selanjutnya tuliskan URL: localhost/PWL\_2024/public/user/NamaAnda. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda



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



1. Jalankan kode dengan menuliskan URL: localhost/PWL\_2024/public/user/. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

****

**Jawab:** Sama seperti sebelumnya, tanda “?” menunjukkan bahwa parameter ini opsional. Perbedaannya jika sebelumnya parameter tidak ada akan diset menjadi null, kalau ini akan diset menjadi “Yonanda Mayla”. Jadi, jika parameter tidak disediakan, akan mengembalikan "Nama saya Yonanda Mayla" (karena $name default ke 'Yonanda Mayla')

**ROUTE NAME**

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



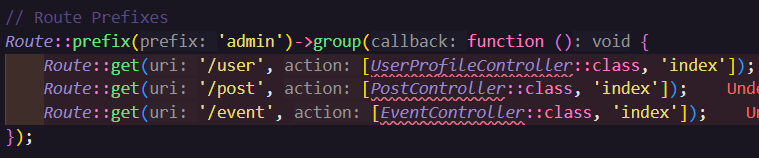
**ROUTE GROUP DAN ROUTE PREFIXES**

Beberapa route yang memiliki atribut yang sama seperti middleware yang sama dapat dikelompokkan menjadi satu kelompok untuk mempermudah penulisan route selain digunakan untuk middleware masih ada lagi penggunaan route group untuk route yang berada dibawah satu subdomain. Contoh penggunaan route group adalah sebagai berikut:



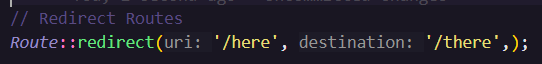
**ROUTE PREFIXES**

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



**REDIRECT ROUTES**

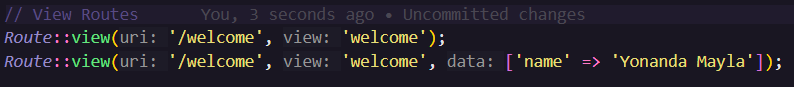
Untuk melakukan redirect pada laravel dapat dilakukan dengan menggunakan “Route::redirect” cara penggunaannya dapat dilihat pada kode program dibawah ini.



Redirect ini akan sering digunakan pada kasus kasus CRUD atau kasus lain yang membutuhkan redirect.

**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.



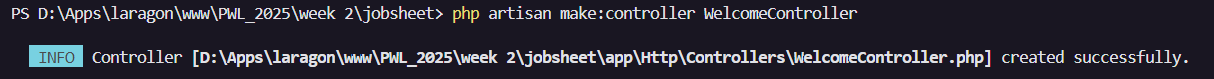
Pada view routes diatas /welcome akan menampilkan view welcome dan pada route kedua /welcome akan menampilkan view welcome dengan tambahan data berupa variabel name

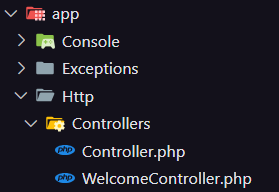
1. **CONTROLLER**

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.

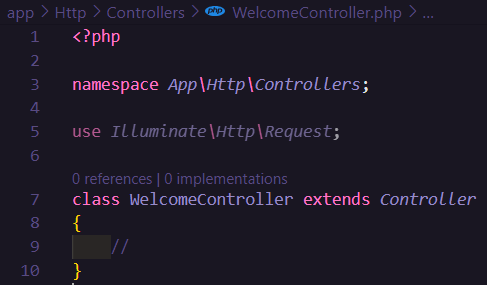
* **Membuat Controller**

1. 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

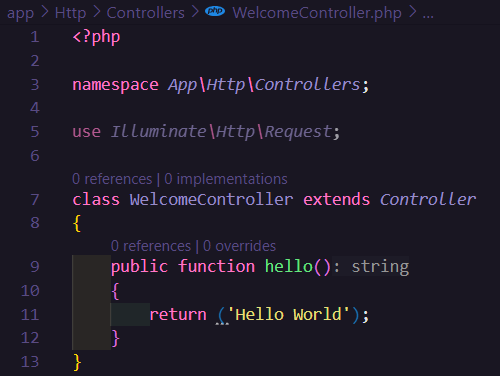




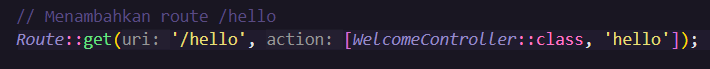
1. Buka file pada app/Http/Controllers/WelcomeController.php. Struktur pada controller dapat digambarkan sebagai berikut:



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

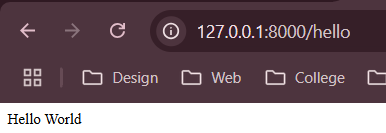


1. Setelah sebuah controller telah didefinisikan action, kita dapat menambahkan controller tersebut pada route. Ubah route /hello menjadi seperti berikut:



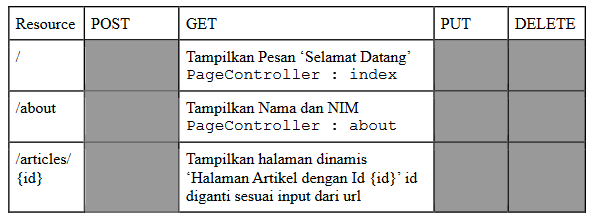
1. Buka browser, tuliskan URL untuk memanggil route tersebut: localhost/PWL\_2024/public/hello. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

**Jawab:**



**Penjelasan:** Route GET tersebut didefinisikan untuk url /hello. Ketika url /hello diakses, maka WelcomeController akan dipanggil. Saat dijalankan kita tidak perlu menambahkan /public sebelum /hello, karena tidak sesuai route nya.

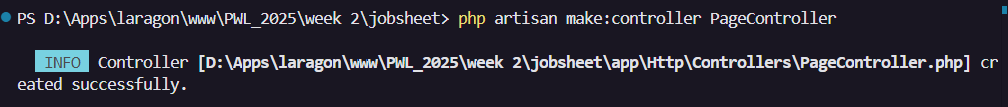
1. Modifikasi hasil pada praktikum poin 2 (Routing) dengan konsep controller. Pindahkan logika eksekusi ke dalam controller dengan nama PageController



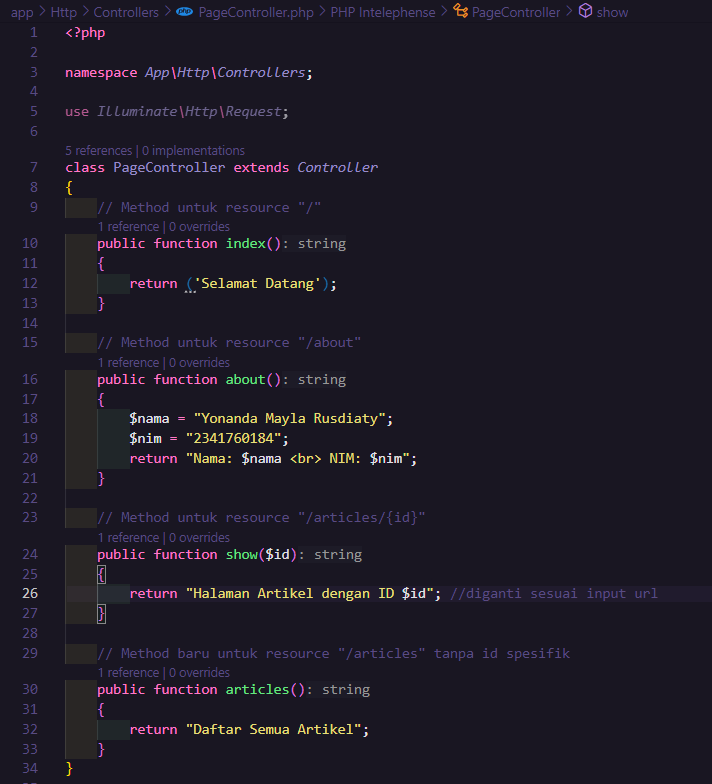


**Jawab:**

1. Kita buat dulu Controllernya dengan nama PageController

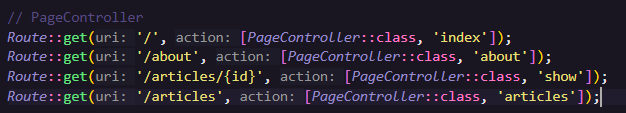


1. Tulis implementasi logika di file PageController



1. Menentukan Routing

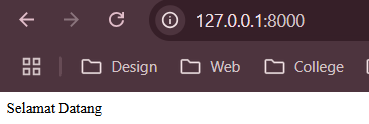
Selanjutnya kita perlu mendefinisikan route di file web.php untuk menghubungkan URL dengan method PageController



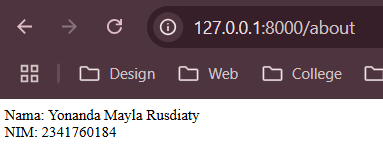
1. Uji Implementasi

Setelah menambahkan kode di atas, kita jalankan:

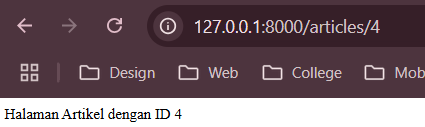
1. <http://127.0.0.1:8000/>



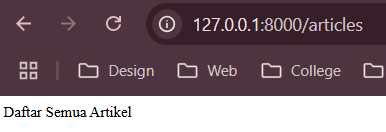
1. <http://127.0.0.1:8000/about>



1. http://127.0.0.1:8000/articles/4



1. http://127.0.0.1:8000/articles

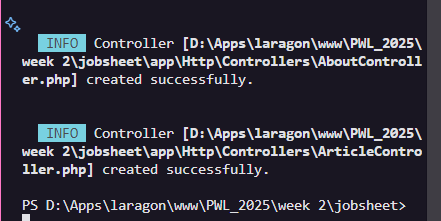


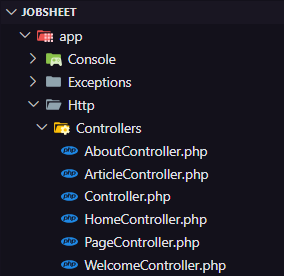
1. Modifikasi kembali implementasi sebelumnya dengan konsep Single Action Controller. Sehingga untuk hasil akhir yang didapatkan akan ada HomeController, AboutController dan ArticleController. Modifikasi juga route yang digunakan

**Jawab:**

Untuk memodifikasi implementasi sebelumnya dengan konsep Single Action Controller, kita akan memecah PageController menjadi beberapa controller yang masing-masing hanya menangani satu tindakan (action) tertentu. Konsep ini sering digunakan untuk membuat controller yang lebih sederhana, terfokus, dan mudah dipelihara, di mana setiap controller hanya memiliki satu method (\_\_invoke)

1. Buat controller baru, yakni HomeController, AboutController, ArticleController

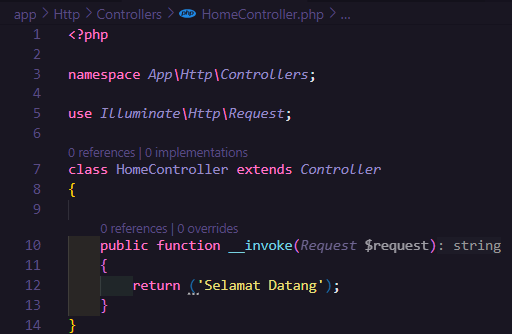




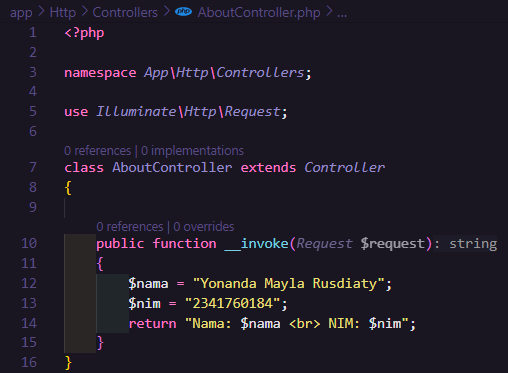
1. Implementasi Controller

Setiap controller akan memiliki satu method \_\_invoke yang menangani logika sesuai tabel sebelumnya

1. HomeController



1. AboutController

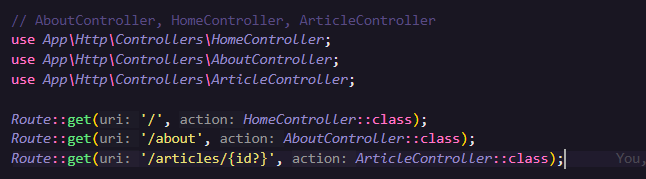


1. ArticleController

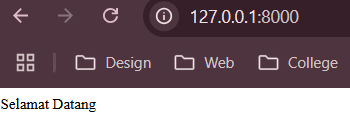
ita akan membuat ArticleController dengan method \_\_invoke yang menerima parameter opsional $id. Jika $id ada, controller akan menampilkan artikel berdasarkan ID; jika tidak, controller akan menampilkan daftar semua artikel.



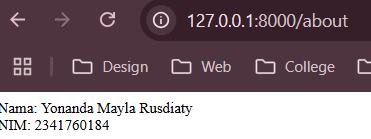
1. Modifikasi Routing



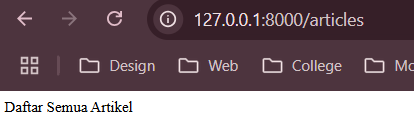
1. Uji Implementasi
2. <http://localhost/>



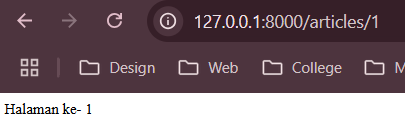
1. <http://localhost/about>



1. <http://localhost/articles>



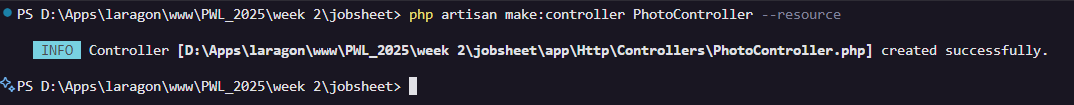
1. <http://localhost/articles/1>



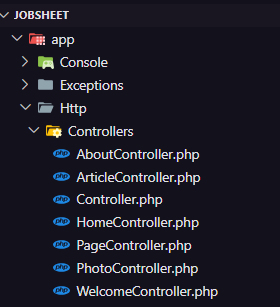
**RESOURCE CONTROLLER**

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.

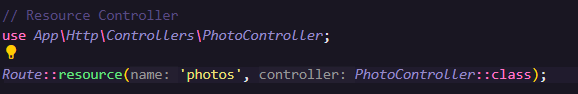
1. Untuk membuatnya dilakukan dengan menjalankan perintah berikut ini di terminal



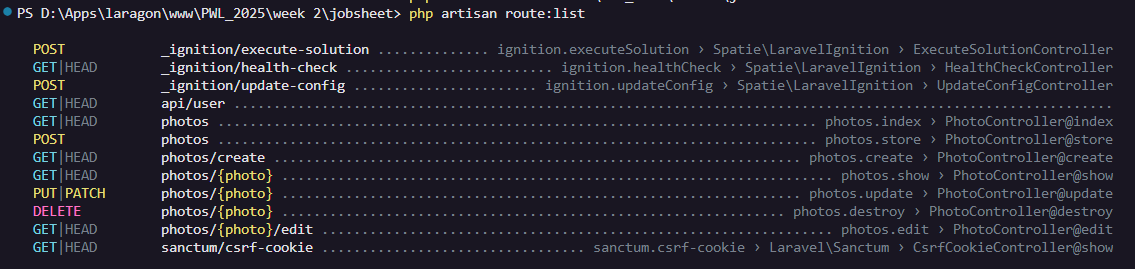
Perintah ini akan men generate sebuah controller dengan nama PhotoController yang berisi method method standar untuk proses CRUD.



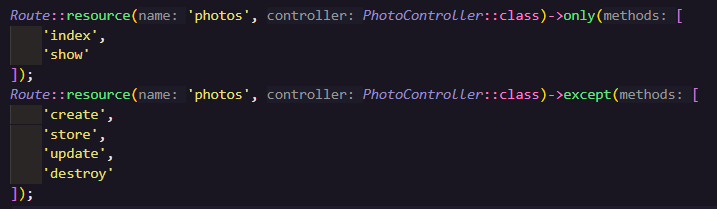
1. Setelah controller berhasil degenerate, selanjutnya harus dibuatkan route agar dapat terhubung dengan frontend. Tambahkan kode program berikut pada file web.php



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



1. 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 mengupdate route pada web.php menjadi seperti berikut ini

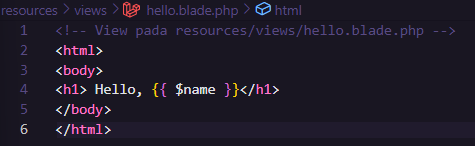


1. **VIEW**

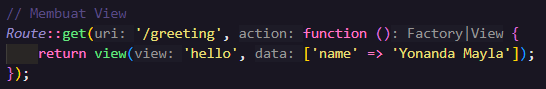
Dalam kerangka kerja Laravel, View merujuk pada bagian dari aplikasi web yang bertanggung jawab untuk menampilkan antarmuka pengguna kepada pengguna akhir. View pada dasarnya adalah file template yang digunakan untuk menghasilkan HTML yang akan ditampilkan kepada pengguna

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.

1. **Membuat View**
2. Pada direktori app/resources/views, buatlah file hello.blade.php.

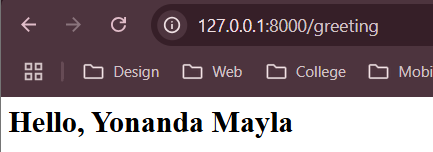


1. View tersebut dapat dijalankan melalui Routing, dimana route akan memanggil View sesuai dengan nama file tanpa ‘blade.php’. (Catatan: Gantilah Andi dengan nama Anda)



1. Jalankan code dengan membuka url localhost/PWL\_2024/public/greeting. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

**Jawab:**

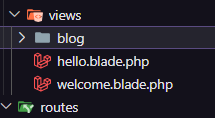
****

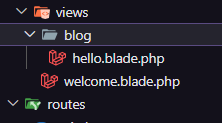
Ketika URL /greeting dijalankan, akan mengembalikan view yg berada di direktori resources/views/hello.blade.php. View tersebut akan menerima data berupa array dengan kunci name dan nilai 'Yonanda Mayla'

1. **View dalam direktori**

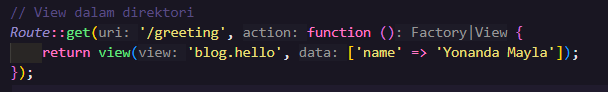
Jika di dalam direktori resources/views terdapat direktori lagi untuk menyimpan file view, sebagai contoh hello.blade.php ada di dalam direktori blog, maka kita bisa menggunakan “dot” notation untuk mereferensikan direktori,

1. Buatlah direktori blog di dalam direktori views.
2. Pindahkan file hello.blade.php ke dalam direktori blog.



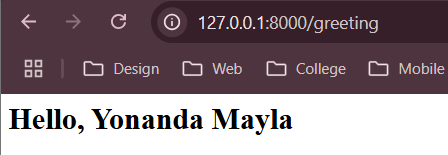


1. Selanjutnya lakukan perubahan pada route



1. Jalankan code dengan membuka url localhost/PWL\_2024/public/greeting. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.

**Jawab:**



Ketika URL /greeting dieksekusi, dia akan mereturnkan view y gada di file hello.blade.php. View tersebut akan menerima data berupa array dengan kunci name dan nilai 'Yonanda Mayla'

1. **Menampilkan View dari Controller**
2. Buka WelcomeController.php dan tambahkan fungsi baru yaitu greeting.

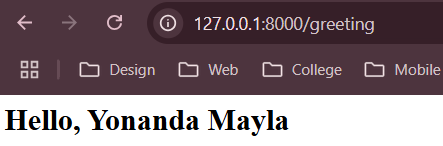


1. Ubah route /greeting dan arahkan ke WelcomeController pada fungsi greeting



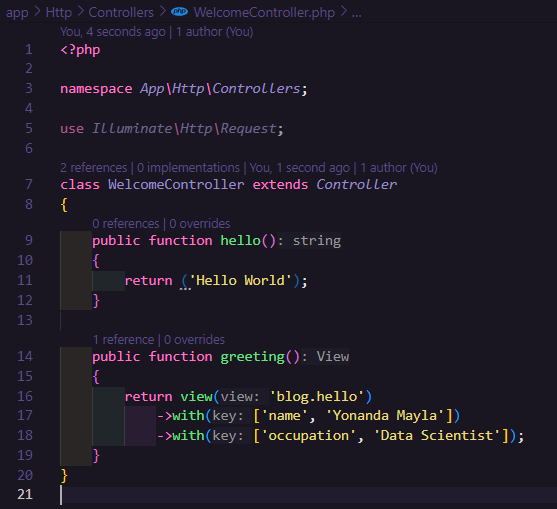
1. Jalankan code dengan membuka url localhost/PWL\_2024/public/greeting. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

**Jawab:**

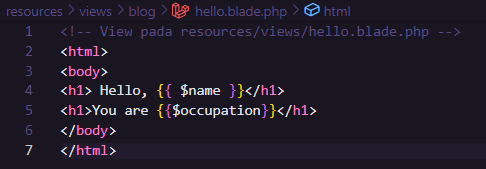


Ketika URL /geeting diakses, method greeting pada WelcomeController akan dipanggil

1. **Meneruskan Data ke View**
2. Buka WelcomeController.php dan tambahkan ubah fungsi greeting.

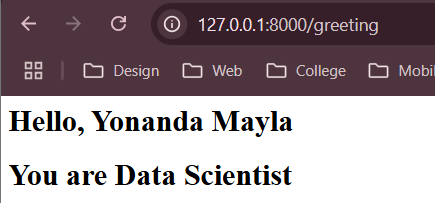


1. Ubah hello.blade.php agar dapat menampilkan dua parameter.



1. Jalankan code dengan membuka url localhost/PWL\_2024/public/greeting. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda

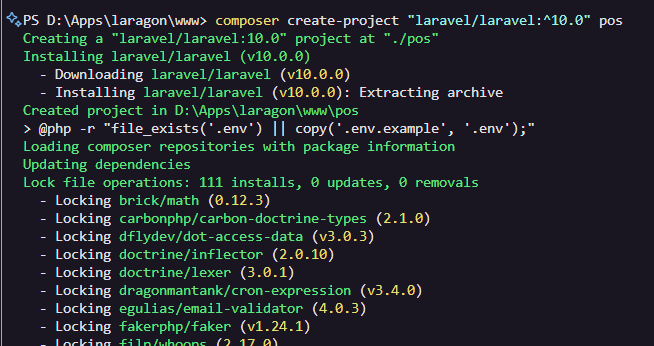
**Jawab:**



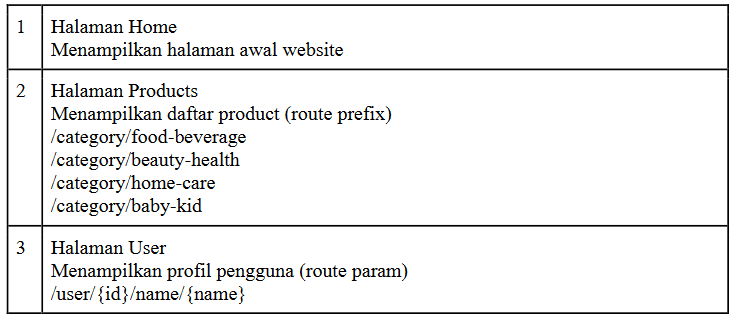
Ketika URL /greeting diakses, method greeting pada WelcomeController akan diakses. Namun disini dia menggunakan with untuk menambahkan bagian data individual ke view

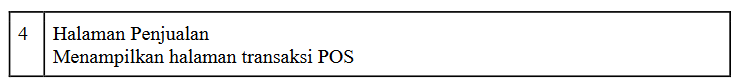
**SOAL PRAKTIKUM**

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.



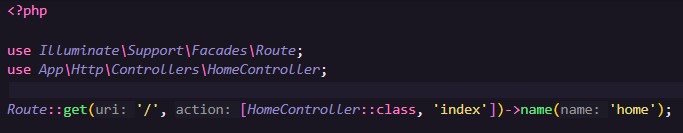
1. Buatlah beberapa route, controller, dan view sesuai dengan ketentuan sebagai berikut.





**Jawab:**

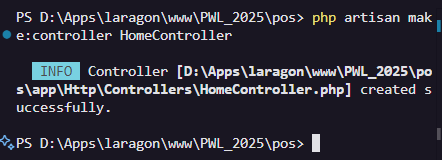
1. Halaman Home
2. Buat route untuk halaman awal website

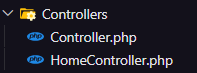


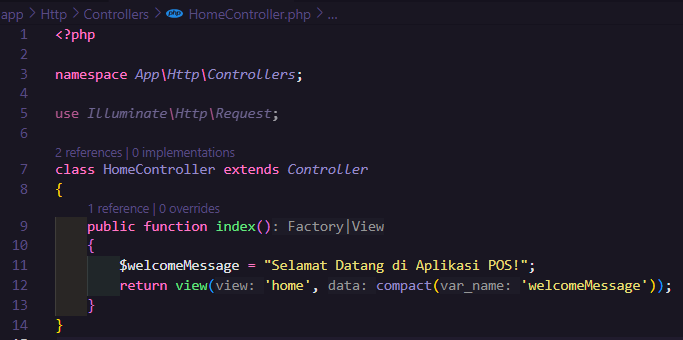
Request akan ditangani oleh method index() di kelas HomeController

1. Buat Controller (HomeController)

Kita akan membuat HomeController dengan fungsi index()



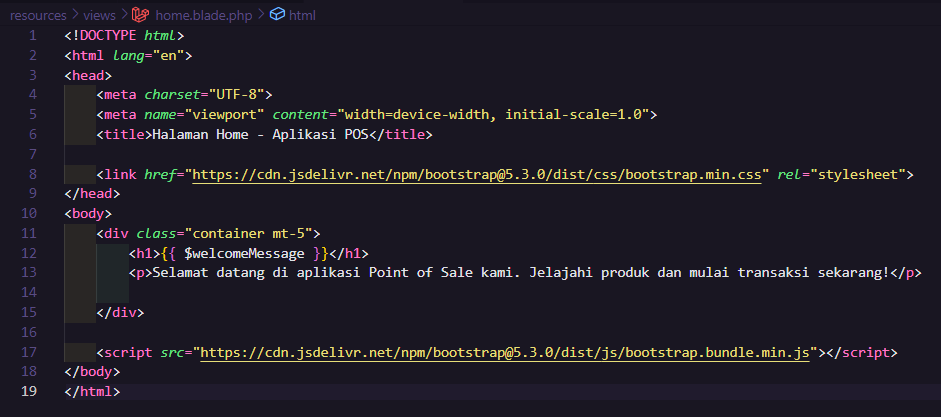




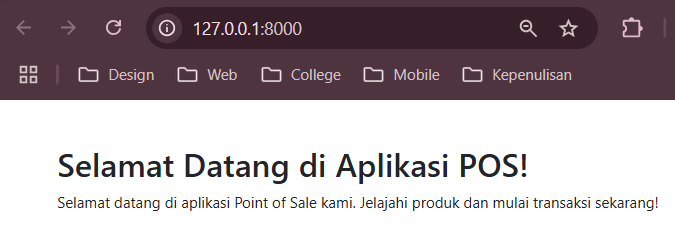
**Penjelasan:** public function index(): Fungsi ini akan dipanggil ketika route / atau /home diakses. return view('home', compact('welcomeMessage'));: Mengembalikan view bernama home (yang akan kita buat nanti) dan mengirimkan variabel $welcomeMessage ke view tersebut.

1. Membuat View (File Blade atau PHP)

Buat file view baru di direktori resources/views/. Namai file tersebut [home.blade.php](http://home.blade.php)



1. Tes Hasil



1. Halaman Products
2. Buat route dengan Prefix /products

Kita akan membuat route dengan prefix /products dan sub-route untuk setiap kategori

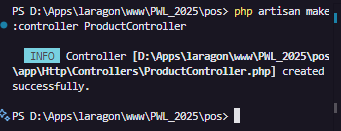


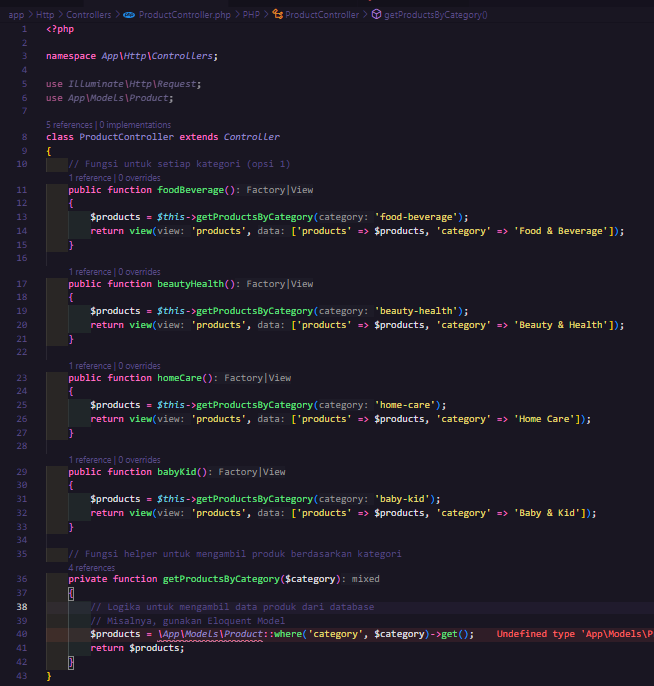
**Penjelasan:**

* Route::prefix('products'): Membuat semua route di dalam group ini memiliki prefix /products
* Route::get(): Menentukan metode HTTP GET untuk setiap kategori.
* Nama route (misalnya, products.food-beverage) memudahkan navigasi dan redirect di aplikas
* Setiap route akan memanggil fungsi di ProductController (yang akan kita buat nanti)

1. Buat Controller (ProductController)

Kita akan membuat ProductController dengan fungsi untuk setiap kategori produk





**Penjelasan:**

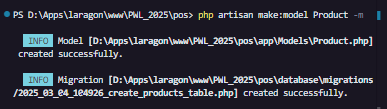
* Setiap fungsi (foodBeverage, beautyHealth, dll.) mengambil data produk berdasarkan kategori dan mengirimkannya ke view products.blade.php.
* getProductsByCategory() adalah fungsi helper yang mengambil data dari model Product (yang akan kita buat nanti).

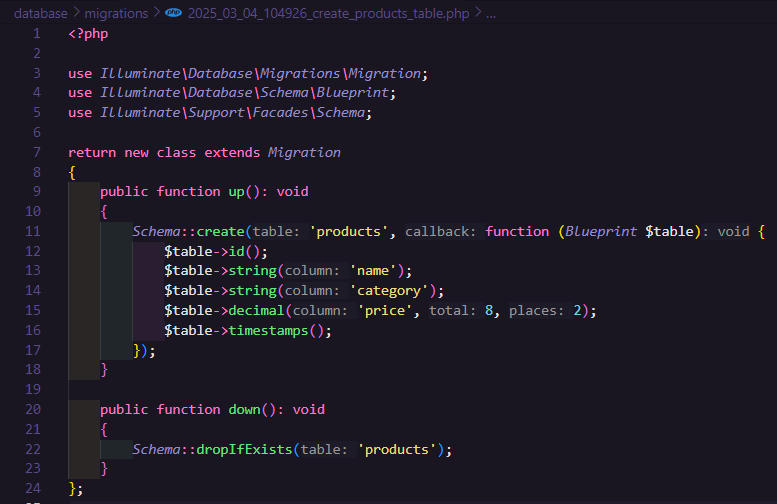
Kemudian, ubah route web.php menjadi:



1. Buat Model untuk Produk

Untuk menyimpan dan mengambil data produk, buat model Product menggunakan Artisan





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