

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.

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

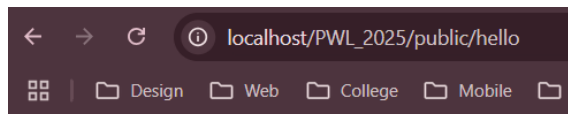
2. Buka file routes/web.php. Tambahkan sebuah route untuk nomor 1 seperti di bawah ini

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

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

3. 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:



Hello, World!

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!"`.

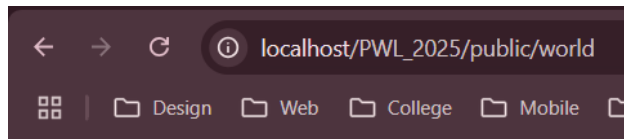
4. Untuk membuat route kedua, tambahkan route `/world` seperti di bawah ini:

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

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

5. 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:

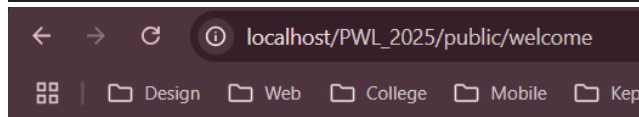


World

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

Jawab:

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



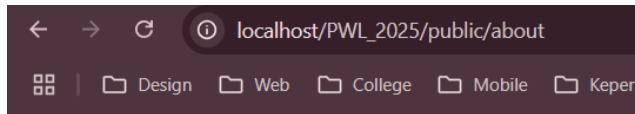
Selamat Datang

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

Jawab:

```
Route::get(uri: '/about', action: function (): string {
    return 'NIM : 2341760184 <br>
    Nama: Yonanda Mayla Rusdiaty';
});
```

You, 1 second ago • Uncommitted changes



NIM : 2341760184

Nama: Yonanda Mayla Rusdiaty

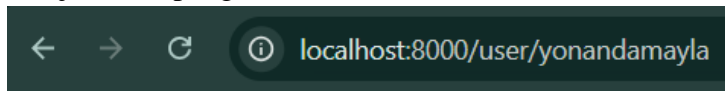
ROUTE PARAMETERS

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

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

```
Route::get(uri: '/user/{yonandamayla}', action: function ($yonandamayla): string {  
    return "Nama saya " . $yonandamayla;  
});
```

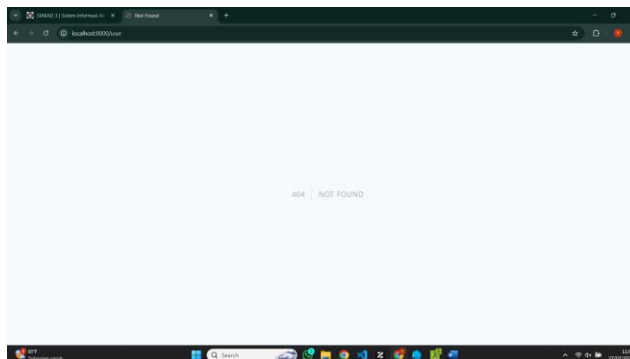
- Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: `localhost/PWL_2024/public/user>NamaAnda`. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda



Nama saya yonandamayla

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

Jawab:

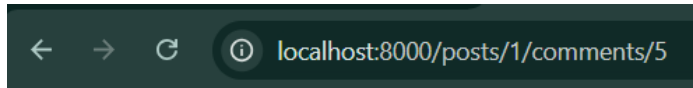


Hasilnya adalah not found karena tidak ada atau tidak ditemukan file user, dan kode yang saya tuliskan mendefinisikan parameter, jadi mengharuskan adanya route parameter.

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

```
Route::get(uri: '/posts/{post}/comments/{comment}', action: function ($postId, $commentId): string {  
    return "Post 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.



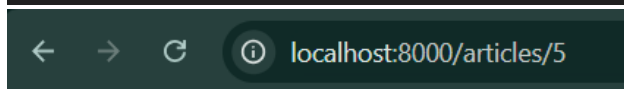
Post ke- 1 Komentar ke-: 5

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

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

Jawab:

```
Route::get(uri: '/articles/{id}', action: function ($id): string {  
    return "Halaman ke- " . $id;  
});
```



Halaman ke- 5

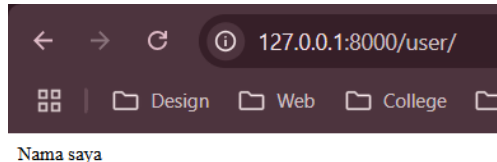
OPTIONAL PARAMETERS

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 ‘?’.

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

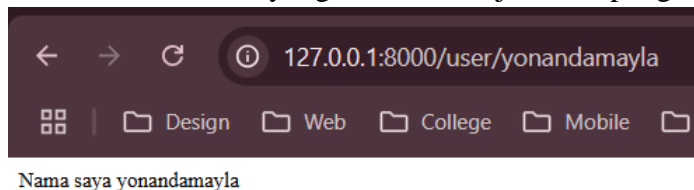
```
Route::get(uri: '/user/{yonandamayla?}', action: function ($yonandamayla=null): string {  
    return "Nama saya " . $yonandamayla;  
});
```

- b. 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 returnnya “Nama saya yonandamayla”

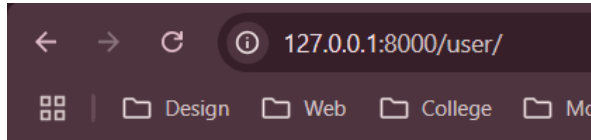
- c. Selanjutnya tuliskan URL: localhost/PWL_2024/public/user>NamaAnda. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda



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

```
Route::get(uri: '/user/{name?}', action: function ($name = 'Yonanda Mayla'): string {  
    return "Nama saya " . $name;  
});
```

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



Nama saya Yonanda Mayla

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 Name  
Route::get(uri: '/user/profile', action: function (): void {  
    //  
})->name(name: 'profile');  
  
Route::get (  
    uri: '/user/profile',  
    action: [UserProfileController::class, 'show'] Undefined  
)->name(name: 'profile');  
// Generating URLs..  
$url = route(name: 'profile');  
// Generating Redirects..  
return redirect()->route(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

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

```
// Route Group dan Route Prefix
Route::middleware(middleware: ['first', 'second'])->group(callback: function (): void {
    Route::get(uri: '/', action: function (): void {
        // Uses first & second Middleware..
    });

    Route::get(uri: '/user/profile', action: function (): void {
        // Uses first & second Middleware..
    });
});

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

Route::middleware(middleware: 'auth') -> group(callback: function (): void {
    Route::get(uri: '/user', action: [UserProfileController::class, 'index']);
    Route::get(uri: '/post', action: [PostController::class, 'index']);
    Route::get(uri: '/event', action: [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 Prefixes
Route::prefix(prefix: 'admin')->group(callback: function (): void {
    Route::get(uri: '/user', action: [UserProfileController::class, 'index']);
    Route::get(uri: '/post', action: [PostController::class, 'index']);
    Route::get(uri: '/event', action: [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.

```
// Redirect Routes
Route::redirect(uri: '/here', destination: '/there',);
```

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.

```
// View Routes You, 3 seconds ago • Uncommitted changes
Route::view(uri: '/welcome', view: 'welcome');
Route::view(uri: '/welcome', view: 'welcome', data: ['name' => 'Yonanda Mayla']);
```

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

2. CONTROLLER

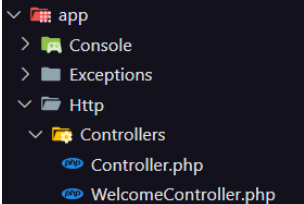
Controller digunakan untuk mengorganisasi logika aplikasi menjadi lebih terstruktur. Logika action aplikasi yang masih ada kaitan dapat dikumpulkan dalam satu kelas Controller. Atau sebuah Controller dapat juga hanya berisi satu buah action. Controller pada Laravel disimpan dalam folder `app/Http/Controllers`.

- Membuat Controller

- a. 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

```
PS D:\Apps\laragon\www\PwL_2025\week 2\jobsheet> php artisan make:controller WelcomeController
```

INFO Controller: [D:\Apps\laragon\www\PwL_2025\week 2\jobsheet\app\Http\Controllers\WelcomeController.php] created successfully.



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

```
app > Http > Controllers > WelcomeController.php > ...
```

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  0 references | 0 implementations
8  class WelcomeController extends Controller
9  {
10     //
11 }
```

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

```

app > Http > Controllers > WelcomeController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class WelcomeController extends Controller
8  {
9      public function hello(): string
10     {
11         return __('Hello World');
12     }
13 }

```

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

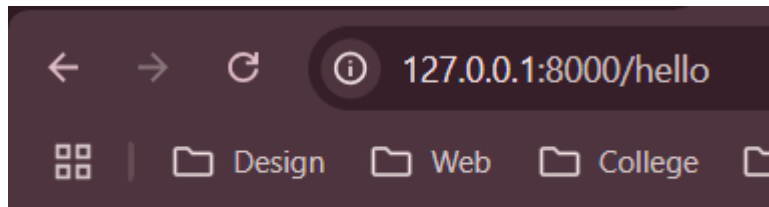
```

// Menambahkan route /hello
Route::get(uri: '/hello', action: [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

Jawab:



Hello World

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.

- 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		
		PageController : articles		

Jawab:

1) Kita buat dulu Controllernya dengan nama PageController

```
PS D:\Apps\Laragon\www\PWL_2025\week 2\jobsheet> php artisan make:controller PageController
INFO controller [D:\Apps\Laragon\www\PWL_2025\week 2\jobsheet\app\Http\Controllers\PageController.php] created successfully.
```

2) Tulis implementasi logika di file PageController

```
app > Http > Controllers > PageController.php > PHP Intelephense > PageController > show
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  5 references | 0 implementations
8  class PageController extends Controller
9  {
10     // Method untuk resource "/"
11     1 reference | 0 overrides
12     public function index(): string
13     {
14         return __('Selamat Datang');
15     }
16
17     // Method untuk resource "/about"
18     1 reference | 0 overrides
19     public function about(): string
20     {
21         $nama = "Yonanda Mayla Rusdiaty";
22         $nim = "2341760184";
23         return "Nama: $nama <br> NIM: $nim";
24     }
25
26     // Method untuk resource "/articles/{id}"
27     1 reference | 0 overrides
28     public function show($id): string
29     {
30         return "Halaman Artikel dengan ID $id"; //diganti sesuai input url
31     }
32
33     // Method baru untuk resource "/articles" tanpa id spesifik
34     1 reference | 0 overrides
35     public function articles(): string
36     {
37         return "Daftar Semua Artikel";
38     }
39 }
```

3) Menentukan Routing

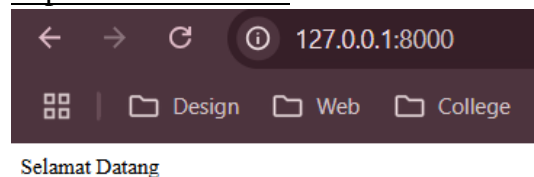
Selanjutnya kita perlu mendefinisikan route di file web.php untuk menghubungkan URL dengan method PageController

```
// PageController
Route::get(uri: '/', action: [PageController::class, 'index']);
Route::get(uri: '/about', action: [PageController::class, 'about']);
Route::get(uri: '/articles/{id}', action: [PageController::class, 'show']);
Route::get(uri: '/articles', action: [PageController::class, 'articles']);
```

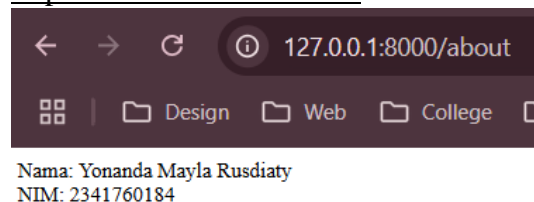
4) Uji Implementasi

Setelah menambahkan kode di atas, kita jalankan:

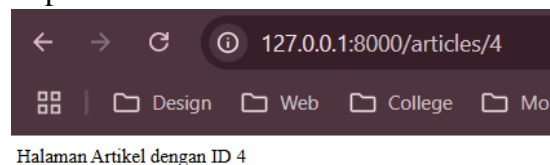
a. <http://127.0.0.1:8000/>



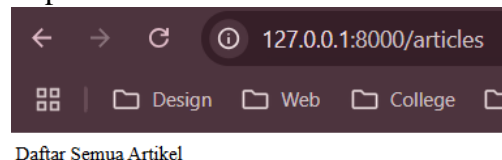
b. <http://127.0.0.1:8000/about>



c. <http://127.0.0.1:8000/articles/4>



d. <http://127.0.0.1:8000/articles>

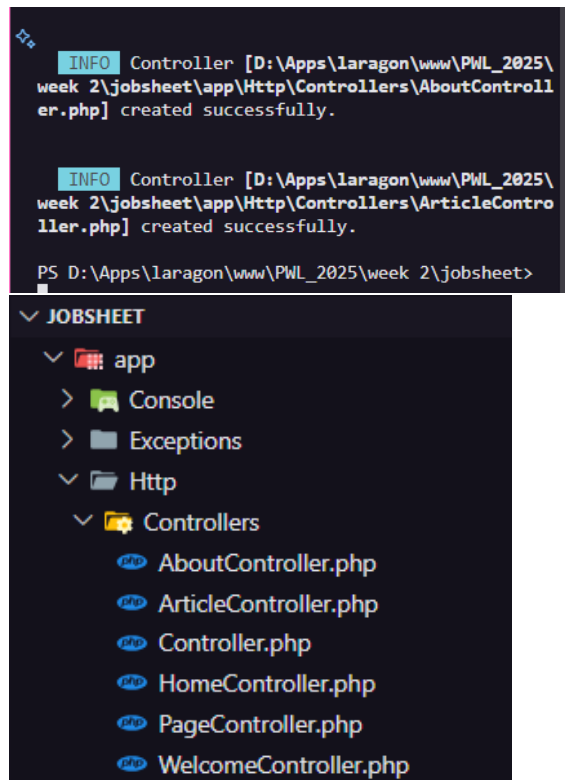


- g. Modifikasi kembali implementasi sebelumnya dengan konsep Single Action Controller. Sehingga untuk hasil akhir yang didapatkan akan ada HomeController, AboutController dan ArticleController. Modifikasi juga route yang digunakan

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



2) Implementasi Controller

Setiap controller akan memiliki satu method `__invoke` yang menangani logika sesuai tabel sebelumnya

a. HomeController

```
app > Http > Controllers > HomeController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class HomeController extends Controller
8  {
9
10     public function __invoke(Request $request): string
11     {
12         return 'Selamat Datang';
13     }
14 }
```

b. AboutController

```
app > Http > Controllers > AboutController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class AboutController extends Controller
8  {
9
10     public function __invoke(Request $request): string
11     {
12         $nama = "Yonanda Mayla Rusdiaty";
13         $nim = "2341760184";
14         return "Nama: $nama <br> NIM: $nim";
15     }
16 }
```

c. 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.

```
app > Http > Controllers > ArticleController.php > PHP > ArticleController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  class ArticleController extends Controller
6  {
7      public function __invoke($id = null): string
8      {
9          if ($id) {
10             return "Halaman Artikel dengan Id $id";
11          }
12          return "Daftar Semua Artikel";
13      }
14 }
```

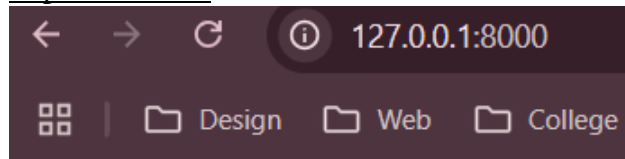
3) Modifikasi Routing

```
// AboutController, HomeController, ArticleController
use App\Http\Controllers\HomeController;
use App\Http\Controllers>AboutController;
use App\Http\Controllers\ArticleController;

Route::get(uri: '/', action: HomeController::class);
Route::get(uri: '/about', action: AboutController::class);
Route::get(uri: '/articles/{id?}', action: ArticleController::class);
```

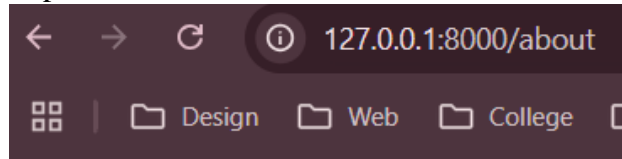
4) Uji Implementasi

a. <http://localhost/>



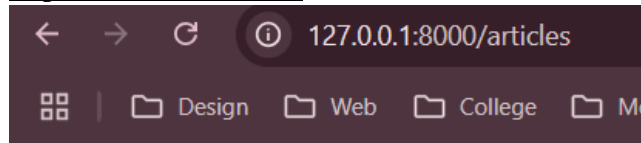
Selamat Datang

b. <http://localhost/about>



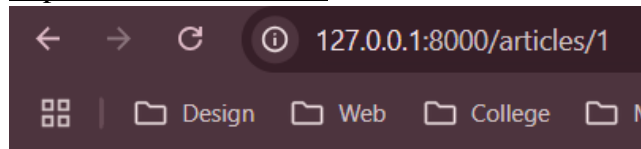
Nama: Yonanda Mayla Rusdiaty
NIM: 2341760184

c. <http://localhost/articles>



Daftar Semua Artikel

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



Halaman ke- 1

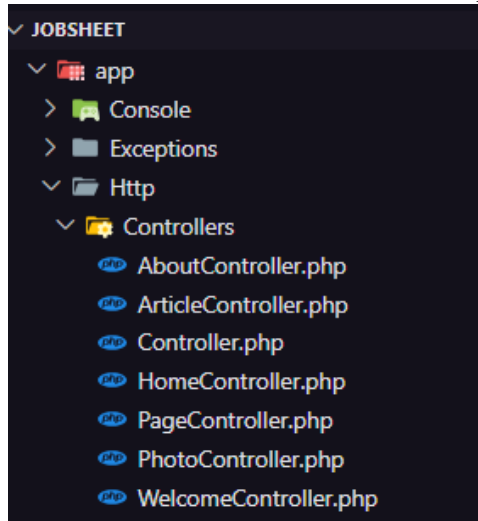
RESOURCE CONTROLLER

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

```
PS D:\Apps\laragon\www\PWL_2025\week 2\jobsheet> php artisan make:controller PhotoController --resource  
  
INFO Controller [D:\Apps\laragon\www\PWL_2025\week 2\jobsheet\app\Http\Controllers\PhotoController.php] created successfully.  
  
PS D:\Apps\laragon\www\PWL_2025\week 2\jobsheet> |
```

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



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

```
// Resource Controller
use App\Http\Controllers\PhotoController;

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

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

```
PS D:\Apps\laragon\www\PWL_2025\week 2\jobsheet> php artisan route:list

POST      _ignition/execute-solution ..... ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
GET|HEAD  _ignition/health-check ..... ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST      _ignition/update-config ..... ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET|HEAD  api/user .....
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  sanctum/csrf-cookie ..... sanctum.csrf-cookie > Laravel\Sanctum > CsrfCookieController@show
```

- 4) 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

```
Route::resource(name: 'photos', controller: PhotoController::class)->only(methods: [  
    'index',  
    'show'  
]);  
Route::resource(name: 'photos', controller: PhotoController::class)->except(methods: [  
    'create',  
    'store',  
    'update',  
    'destroy'  
]);
```



3. 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.

a. Membuat View

- 1) Pada direktori app/resources/views, buatlah file hello.blade.php.

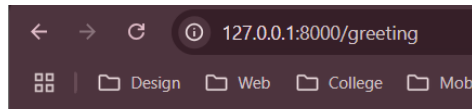
```
resources > views >  hello.blade.php >  html
1  <!-- View pada resources/views/hello.blade.php -->
2  <html>
3  <body>
4  <h1> Hello, {{ $name }}</h1>
5  </body>
6  </html>
```

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

```
// Membuat View
Route::get(uri: '/greeting', action: function (): Factory|View {
    return view(view: 'hello', data: ['name' => 'Yonanda Mayla']);
});
```

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

Jawab:



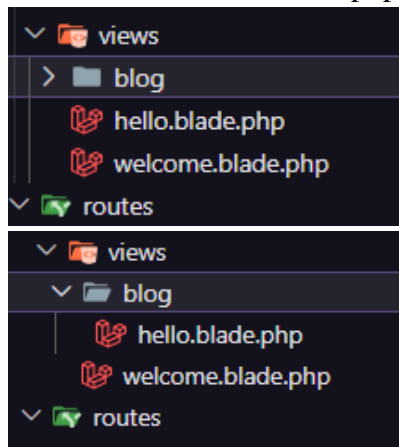
Hello, Yonanda Mayla

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'

b. 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.

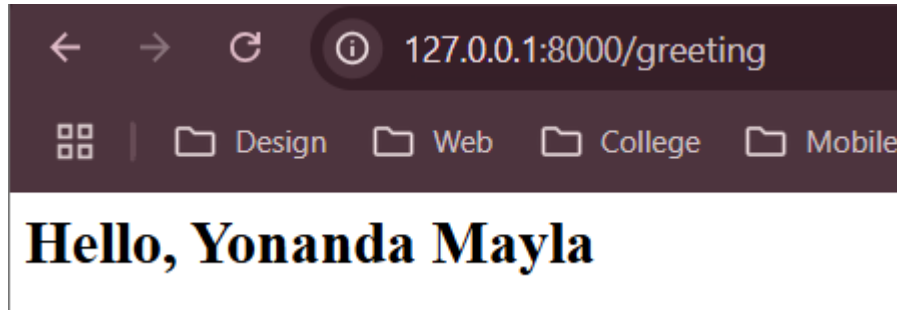


- 3) Selanjutnya lakukan perubahan pada route

```
// View dalam direktori
Route::get(uri: '/greeting', action: function (): Factory|View {
    return view(view: 'blog.hello', data: ['name' => 'Yonanda Mayla']);
});
```

- 4) 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'

c. Menampilkan View dari Controller

- 1) Buka WelcomeController.php dan tambahkan fungsi baru yaitu greeting.

```
app > Http > Controllers > WelcomeController.php > ...
You, 1 second ago | 1 author (You)
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  1 reference | 0 implementations | You, 1 second ago | 1 author (You)
8  class WelcomeController extends Controller
9  {
10     0 references | 0 overrides
11     public function hello(): string
12     {
13         return __('Hello World');
14     }
15
16     0 references | 0 overrides
17     public function greeting(): Factory|View
18     {
19         return view('blog.hello', data: ['name' => 'Yonanda Mayla']);
20     }
21 }
```

- 2) Ubah route /greeting dan arahkan ke WelcomeController pada fungsi greeting

```
// Route greeting menampilkan view dalam controller
Route::get(uri: '/greeting', action: [WelcomeController::class, 'greeting']);
```

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

Jawab:



Hello, Yonanda Mayla

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

d. Meneruskan Data ke View

- 1) Buka WelcomeController.php dan tambahkan ubah fungsi greeting.

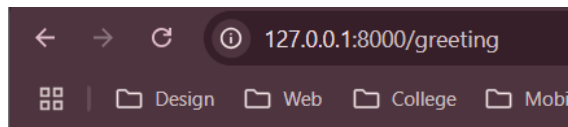
```
app > Http > Controllers > WelcomeController.php > ...  
You, 4 seconds ago | 1 author (You)  
1 <?php  
2  
3 namespace App\Http\Controllers;  
4  
5 use Illuminate\Http\Request;  
6  
2 references | 0 implementations | You, 1 second ago | 1 author (You)  
7 class WelcomeController extends Controller  
8 {  
9     0 references | 0 overrides  
10     public function hello(): string  
11     {  
12         return ('Hello World');  
13     }  
14     1 reference | 0 overrides  
15     public function greeting(): View  
16     {  
17         return view(view: 'blog.hello')  
18             ->with(key: ['name', 'Yonanda Mayla'])  
19             ->with(key: ['occupation', 'Data Scientist']);  
20     }  
21 }
```

- 2) Ubah hello.blade.php agar dapat menampilkan dua parameter.

```
resources > views > blog > hello.blade.php > html
1 <!-- View pada resources/views/hello.blade.php -->
2 <html>
3 <body>
4 <h1> Hello, {{ $name }}</h1>
5 <h1>You are {{$occupation}}</h1>
6 </body>
7 </html>
```

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

Jawab:



Hello, Yonanda Mayla

You are Data Scientist

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.

```
PS D:\Apps\laragon\www> composer create-project "laravel/laravel:^10.0" pos
Creating a "laravel/laravel:10.0" project at "./pos"
Installing laravel/laravel (v10.0.0)
- Downloading laravel/laravel (v10.0.0)
- Installing laravel/laravel (v10.0.0): Extracting archive
Created project in D:\Apps\laragon\www\pos
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 111 installs, 0 updates, 0 removals
- Locking brick/math (0.12.3)
- Locking carbonphp/carbon-doctrine-types (2.1.0)
- Locking dflydev/dot-access-data (v3.0.3)
- Locking doctrine/inflector (2.0.10)
- Locking doctrine/lexer (3.0.1)
- Locking dragonmantank/cron-expression (v3.4.0)
- Locking egulias/email-validator (4.0.3)
- Locking fakerphp/faker (v1.24.1)
- Locking filp/whoops (2.17.0)
```

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

1	Halaman Home Menampilkan halaman awal website
2	Halaman Products Menampilkan daftar product (route prefix) /category/food-beverage /category/beauty-health /category/home-care /category/baby-kid
3	Halaman User Menampilkan profil pengguna (route param) /user/{id}/name/{name}

4	Halaman Penjualan Menampilkan halaman transaksi POS
---	--

Jawab:

1. Route

```

1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\HomeController;
5  use App\Http\Controllers\ProductController;
6  use App\Http\Controllers\ProfileController;
7  use App\Http\Controllers\TransactionController;
8
9  Route::get(uri: '/', action: [HomeController::class, 'index'])->name(name: 'home');
10
11 Route::prefix(prefix: '/category')->group(callback: function (): void {
12     Route::get(uri: '/food-beverage', action: [ProductController::class, 'foodBeverage']);
13     Route::get(uri: '/beauty-health', action: [ProductController::class, 'beautyHealth']);
14     Route::get(uri: '/home-care', action: [ProductController::class, 'homeCare']);
15     Route::get(uri: '/baby-kid', action: [ProductController::class, 'babyKid']);
16 });
17
18 Route::get(uri: '/user/{id}/name/{name}', action: [ProfileController::class, 'index']);
19
20 Route::get(uri: '/transactions', action: [TransactionController::class, 'index']);

```

2. Controller

Controller Home

```

app > Http > Controllers > HomeController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  2 references | 0 implementations
8  class HomeController extends Controller
9  {
10     1 reference | 0 overrides
11     public function index(): Factory|View
12     {
13         $welcomeMessage = "Selamat Datang di Aplikasi POS!";
14         return view(view: 'home', data: compact(var_name: 'welcomeMessage'));
15     }
16 }

```

Controller Product

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  5 references | 0 implementations
8  class ProductController extends Controller
9  {
10     1 reference | 0 overrides
11     public function foodBeverage(): Factory|View
12     {
13         return view(view: "category.food-baverage");
14     }
15
16     1 reference | 0 overrides
17     public function babyKid(): Factory|View
18     {
19         return view(view: "category.baby-kid");
20     }
21
22     1 reference | 0 overrides
23     public function beautyHealth(): Factory|View
24     {
25         return view(view: "category.beauty-health");
26     }
27
28     1 reference | 0 overrides
29     public function homeCare(): Factory|View
30     {
31         return view(view: "category.home-care");
32     }
33 }
```

Controller Profile

```
app > Http > Controllers > ProfileController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  2 references | 0 implementations
8  class ProfileController extends Controller
9  {
10     1 reference | 0 overrides
11     public function index($id, $name): View
12     {
13         return view('profile')->with(key: 'id', value: '2341760184')->with(key: 'name', value: 'Yonanda Mayla Rusdiaty');
```

Controller Transaction

```
app > Http > Controllers > TransactionController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  2 references | 0 implementations
8  class TransactionController extends Controller
9  {
10     1 reference | 0 overrides
11     public function index(): mixed|View
12     {
13         $transactions = [
14             ['product' => 'Nasi Goreng', 'qty' => 2, 'price' => 25000, 'total' => 50000],
15             ['product' => 'Ayam Geprek', 'qty' => 3, 'price' => 20000, 'total' => 60000],
16             ['product' => 'Mie Ayam', 'qty' => 1, 'price' => 18000, 'total' => 18000],
17             ['product' => 'Es Teh Manis', 'qty' => 2, 'price' => 5000, 'total' => 10000],
18             ['product' => 'Bakso', 'qty' => 4, 'price' => 22000, 'total' => 88000],
19         ];
20
21         return view('transactions')->with(key: 'transactions', value: $transactions);
22     }
23 }
```

3. View

Home

```
resources > views > home.blade.php > html
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <title>Halaman Home - Aplikasi POS</title>
8  </head>
9
10 <body>
11     <div class="container mt-5">
12         <h1>{{ $welcomeMessage }}</h1>
13         <p>Selamat datang di aplikasi Point of Sale kami. Jelajahi produk dan mulai transaksi sekarang!</p>
14     </div>
15 </body>
16 </html>
```

Baby and kid

```
resources > views > category > baby-kid.blade.php > html > body > h1
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Document</title>
9  </head>
10
11 <body>
12     <h1>INI HALAMAN BABY KID</h1>
13 </body>
14
15 </html>
```

Beauty and health

```
resources > views > category > beauty-health.blade.php > html > body > h1
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Document</title>
9  </head>
10
11 <body>
12     <h1>INI HALAMAN BEAUTY HEALTH</h1>
13 </body>
14
15 </html>
```

Food and beverage

```
resources > views > category > food-bavage.blade.php > html > body > h1
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Document</title>
9  </head>
10
11 <body>
12     <h1>INI HALAMAN FOOD BAVERAGE</h1>
13 </body>
14
15 </html>
```

Home care

```
resources > views > category > home-care.blade.php > html
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Document</title>
9  </head>
10
11 <body>
12     <h1>INI HALAMAN HOME CARE</h1>
13 </body>
14
15 </html>
```

Profile

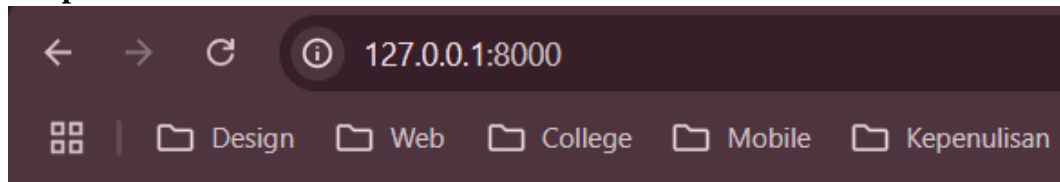
```
resources > views > profile.blade.php > html
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Document</title>
9  </head>
10
11 <body>
12     <h1>PROFILE</h1>
13     <h4>ID : {{ $id }}</h4>
14     <h4>ID : {{ $name }}</h4>
15 </body>
16
17 </html>
```

Transaction

resources > views > transaction.blade.php > html

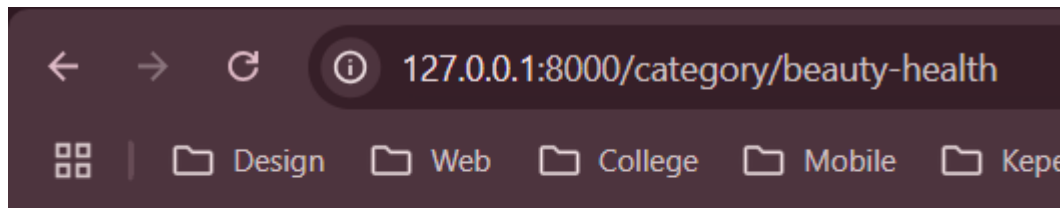
```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>Document</title>
9  </head>
10
11 <body>
12
13     <body>
14         <h1>HISTORY PENJUALAN</h1>
15         <table>
16             <tr>
17                 <th>NO</th>
18                 <th>PRODUCT</th>
19                 <th>QTY</th>
20                 <th>PRICE</th>
21                 <th>TOTAL</th>
22             </tr>
23             @foreach ($transactions as $transaction)
24                 <tr>
25                     <td>{{ $loop->iteration }}</td>
26                     <td>{{ $transaction['product'] }}</td>
27                     <td>{{ $transaction['qty'] }}</td>
28                     <td>{{ $transaction['price'] }}</td>
29                     <td>{{ $transaction['total'] }}</td>
30                 </tr>
31             @endforeach
32         </table>
33     </body>
34 </body>
35
36 </html>
```


Output :

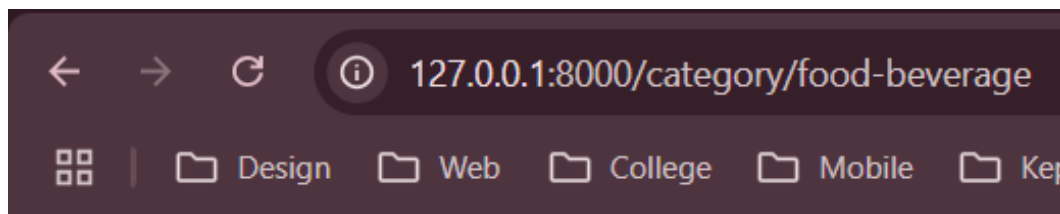


Selamat Datang di Aplikasi POS!

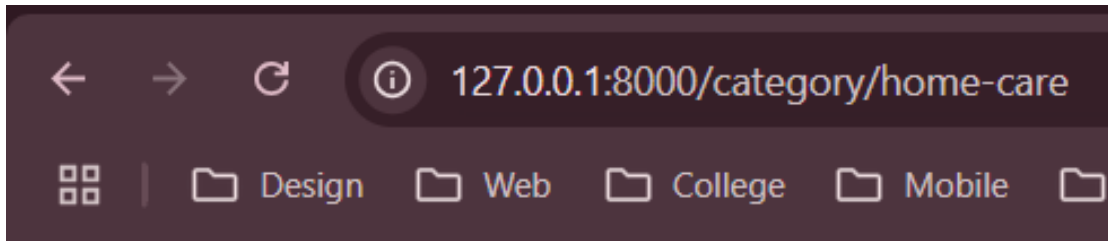
Selamat datang di aplikasi Point of Sale kami. Jelajahi produk dan mulai transaksi sekarang!



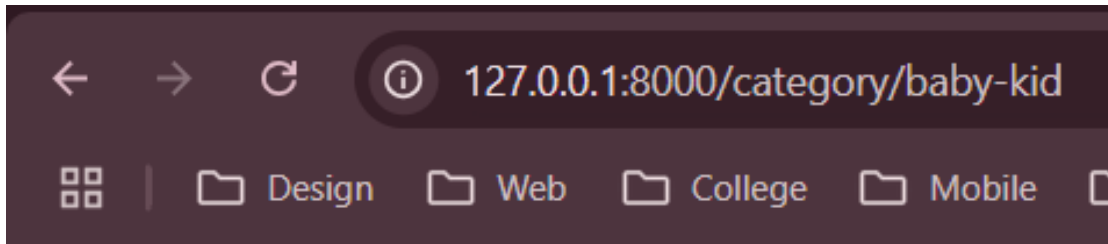
INI HALAMAN BEAUTY HEALTH



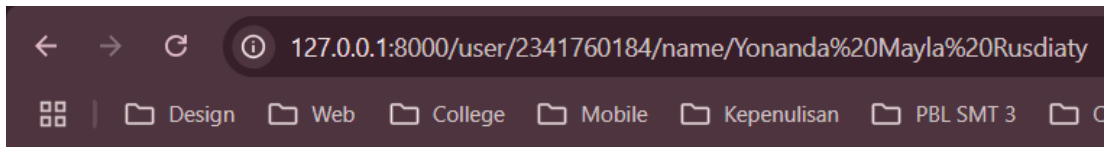
INI HALAMAN FOOD BAVERAGE



INI HALAMAN HOME CARE



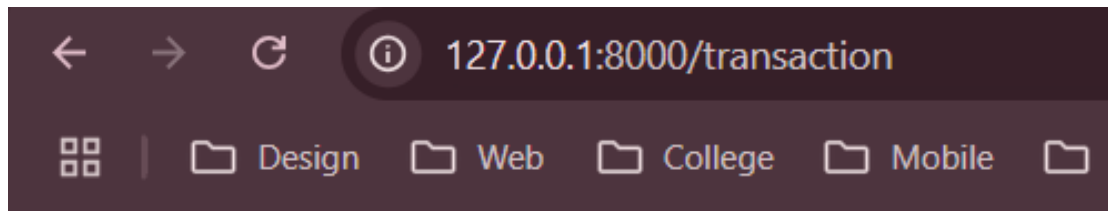
INI HALAMAN BABY KID



PROFILE

ID : 2341760184

ID : Yonanda Mayla Rusdiaty



HISTORY PENJUALAN

NO	PRODUCT	QTY	PRICE	TOTAL
1	Nasi Goreng	2	25000	50000
2	Ayam Geprek	3	20000	60000
3	Mie Ayam	1	18000	18000
4	Es Teh Manis	2	5000	10000
5	Bakso	4	22000	88000

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