



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 1 (satu)

JOBSHEET 02

ROUTING, CONTROLLER, DAN VIEW

Langkah-langkah Praktikum:

- a. 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:

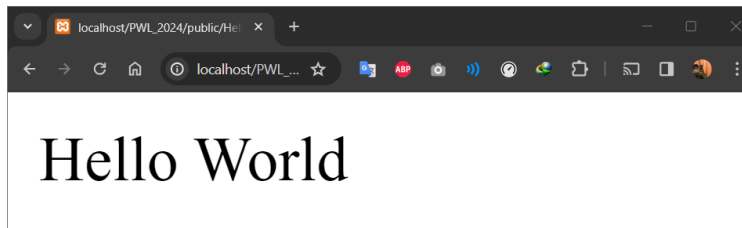
```
use Illuminate\Support\Facades\Route;

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

```
routes > web.php > ...
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  /*
6   |-----
7   | Web Routes
8   |-----
9   |
10  | Here is where you can register web routes for your application. These
11  | routes are loaded by the RouteServiceProvider and all of them will
12  | be assigned to the "web" middleware group. Make something great!
13  |
14  */
15
16  Route::get('/Hello', function () {
17      return ('Hello World');
18  });
```



- 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.

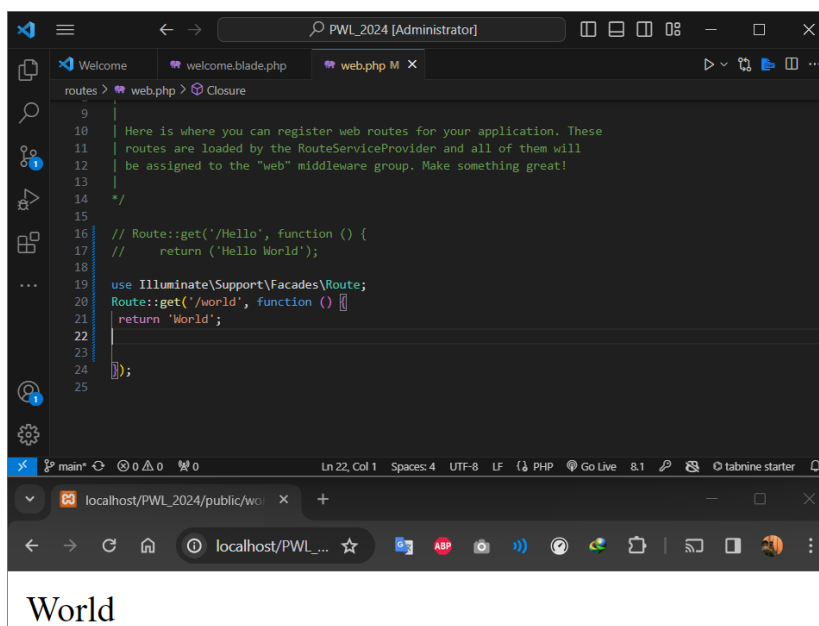


Pada percobaan kali ini akan menghasilkan output berupa “Hello World” karena pada kode program di atas melakukan return berupa “Hello World”.

- 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.



Menghasilkan output “World” karena kode program meminta return “World”



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

```
23 use Illuminate\Support\Facades\Route;
24
25 Route::get('/', function(){
26     return 'Selamat Datang';
27 });
28
29
```

localhost/PWL_2024/public/welcome

Selamat Datang

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

```
30 use Illuminate\Support\Facades\Route;
31
32 Route::get('/about', function(){
33     return 'NIM : 2241720193, NAMA : Rio Bagas Hermawan';
34 });
35
```

localhost/PWL_2024/public/about

NIM : 2241720193, NAMA : Rio Bagas Hermawan



- 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:

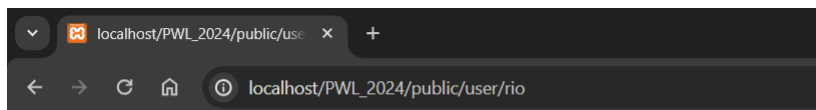
Untuk membuat routing dengan parameter dapat dilakukan dengan cara berikut ini.

- 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; } );
```

```
36 use Illuminate\Support\Facades\Route;  
37  
38 Route::get('/user/{rio}', function ($name) {  
39     return 'Nama Saya ' . $name;  
40 });
```

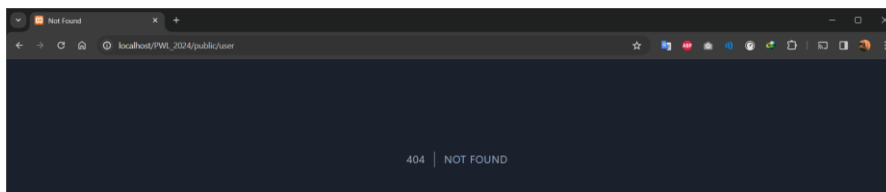
- Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: **localhost/PWL_2024/public/user>NamaAnda**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Nama Saya rio

Pada percobaan ini saat menuliskan url “user/rio/” maka akan melakukan return berupa “Nama Saya Rio”

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



Tidak menghasilkan output atau error dikarenakan pada kode program diruliskan `/user/nama` tapi pada url hanya dituliskan `/user`

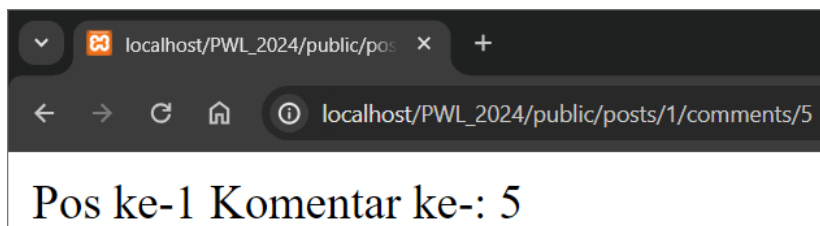


- d. Suatu route, juga bisa menerima lebih dari 1 parameter seperti kode berikut ini. Route menerima parameter \$postId dan juga \$comment.

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

```
42 use Illuminate\Support\Facades\Route;  
43  
44 Route::get('/posts/{post}/comments/{comment}', function  
45 ($postId, $commentId) {  
46     return 'Pos ke-' . $postId . " Komentar ke-: " . $commentId;  
47 });
```

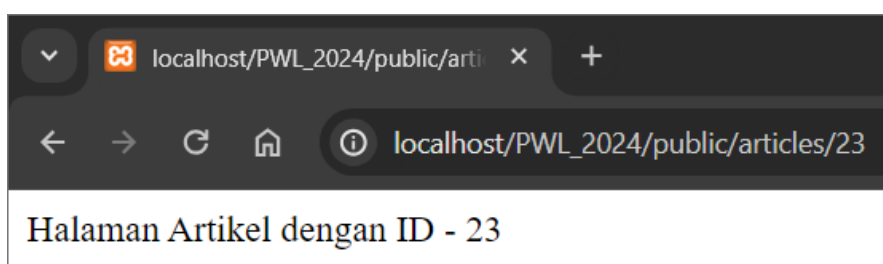
- e. Jalankan kode dengan menuliskan URL untuk memanggil route tersebut: **localhost/PWL_2024/public/posts/1/comments/5**. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Hasil output diperoleh dari penulisan url yang menginputkan secara manual urutan post ke-1 dan juga komentar ke-5

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

```
49 use Illuminate\Support\Facades\Route;  
50 Route::get('/articles/{id}', function  
51 ($id) {  
52     return 'Halaman Artikel dengan ID - ' . $id;  
53 });
```





- 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.

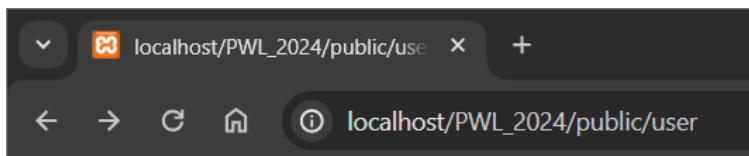
- Kita akan memanggil route `/user` sekaligus mengirimkan parameter berupa nama user `$name` dimana parameternya bersifat opsional.

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

```
55 use Illuminate\Support\Facades\Route;  
56  
57 Route::get('/user/{rio?}', function ($name=null) {  
58     return 'Nama saya '.$name;  
59 });
```

- Jalankan kode dengan menuliskan URL:

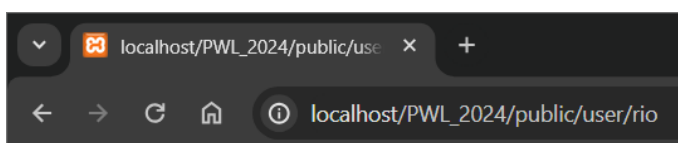
localhost/PWL_2024/public/user/. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Nama saya

Hasil output hanya menampilkan “Nama Saya” karena url hanya dituliskan `/user`.

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



Nama saya rio

Pada hasil output kali ini dapat menghasilkan “Nama saya rio” sesuai dengan kode program karena pada penulisan url dituliskan `user/rio` yang mana `rio` merupakan isi dari `$name`



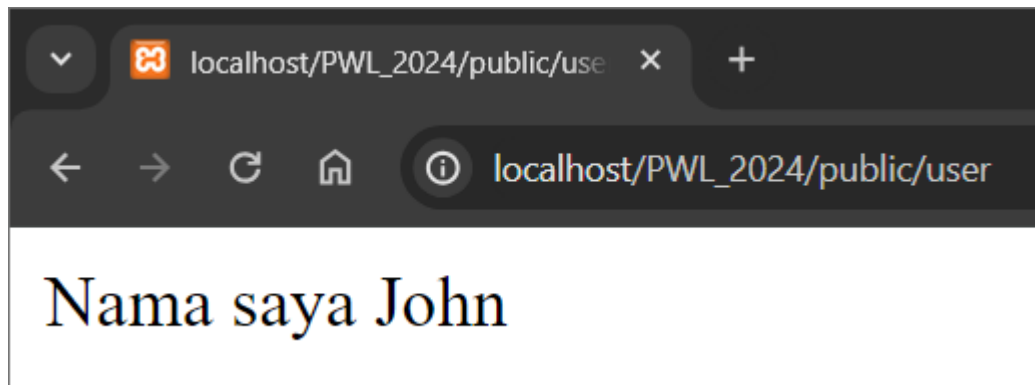
- d. Ubah kode pada route /user menjadi seperti di bawah ini.

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

```
55 use Illuminate\Support\Facades\Route;  
56  
57 Route::get('/user/{rio?}', function ($name='John') {  
58     return 'Nama saya '.$name;  
59 });
```

- e. Jalankan kode dengan menuliskan URL:

localhost/PWL_2024/public/user/. Perhatikan halaman yang muncul dan jelaskan pengamatan Anda.



Pada percobaan kali ini \$name sudah diberi nilai berupa “John” sehingga apabila menuliskan url hanya /user sudah dapat melakukan return \$name tersebut



Langkah-langkah Praktikum:

- 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.

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

- b. 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.





- 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		

- 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.

1. Web.php

```
78 use App\Http\Controllers\PageController;  
79  
80 Route::get('/', [PageController::class, 'index']);  
81 Route::get('/about', [PageController::class, 'about']);  
82 Route::get('/articles/{id}', [PageController::class, 'articles']);
```

2. PageController.php

```
app > Http > Controllers > PageController.php > PHP Intelephense > PageController > about  
1 <?php  
2 namespace App\Http\Controllers;  
3 use Illuminate\Http\Request;  
4 2 references | 0 implementations  
4 class PageController extends Controller  
5 {  
6 1 reference | 0 overrides  
6 public function index()  
7 {  
8 return 'Selamat Datang';  
9 }  
10  
11 0 references | 0 overrides  
11 public function about()  
12 {  
13 return 'NAMA : Rio Bagus Hermawan' . '<br>' . 'NIM : 2241720193';  
14 }  
15 0 references | 0 overrides  
15 public function articles($id)  
16 {  
17 return 'Halaman Artikel dengan ID ' . $id;  
18 }  
19 }
```



3. HomeController

```
app > Http > Controllers > HomeController.php > PHP Intelephense > HomeController
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Http\Request;
4  class HomeController extends Controller
5  {
6      public function index()
7      {
8          return 'Selamat Datang';
9      }
10 }
```

localhost/PWL_2024/public/

Selamat Datang

4. AboutController.php

```
app > Http > Controllers > AboutController.php > PHP Intelephense > AboutController
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Http\Request;
4  class AboutController extends Controller
5  {
6      public function index()
7      {
8          return 'Nama : Rio Bagas Hermawan' . '<br>' . 'NIM : 2241720193';
9      }
10 }
```

localhost/PWL_2024/public/abc

NAMA : Rio Bagas Hermawan
NIM : 2241720193



5. ArticleController

```
app > Http > Controllers > ArticleController.php > ...
1  <?php
2  namespace App\Http\Controllers;
3  use Illuminate\Http\Request;
4  class ArticleController extends Controller
5  {
6      public function index($id)
7      {
8          return 'Halaman Artikel dengan ID ' . $id;
9      }
10 }
```

localhost/PWL_2024/public/arti x +

localhost/PWL_2024/public/articles/23

Halaman Artikel dengan ID 23



- 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.

```
C:\laragon\www\PWL_2024>php artisan make:controller PhotoController --resource  
INFO Controller [C:\laragon\www\PWL_2024\app\Http\Controllers\PhotoController.php] created successfully.
```

- 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);
```

```
85 use App\Http\Controllers\PhotoController;  
86 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
	GET HEAD	photos	photos.index	App\Http\Controllers\PhotoController@index	auth:api
	POST	photos	photos.store	App\Http\Controllers\PhotoController@store	web
	GET HEAD	photos/create	photos.create	App\Http\Controllers\PhotoController@create	web
	GET HEAD	photos/{photo}	photos.show	App\Http\Controllers\PhotoController@show	web
	PUT PATCH	photos/{photo}	photos.update	App\Http\Controllers\PhotoController@update	web
	DELETE	photos/{photo}	photos.destroy	App\Http\Controllers\PhotoController@destroy	web
	GET HEAD	photos/{photo}/edit	photos.edit	App\Http\Controllers\PhotoController@edit	web
	GET HEAD POST PUT PATCH DELETE OPTIONS	specialMahasiswa		Closure	web
	GET POST HEAD	specialUrl		Closure	web

```
C:\laragon\www\PWL_2024>php artisan route:lis

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  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

Showing [12] 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 mengupdate route pada web.php menjadi seperti berikut ini.

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

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

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

Simpan perubahan yang telah dilakukan pada Git.



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`.

- 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>
```

```
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>
```

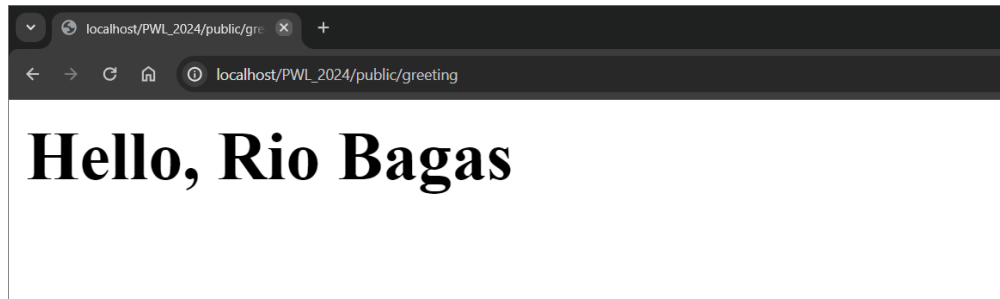
- b. View tersebut dapat dijalankan melalui Routing, dimana *route* akan memanggil View sesuai dengan nama *file* tanpa `'blade.php'`. (Catatan: Gantilah Andi dengan nama Anda)

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

```
91 Route::get('/greeting', function () {return view('hello', ['name' => 'Rio Bagus']);});
```



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



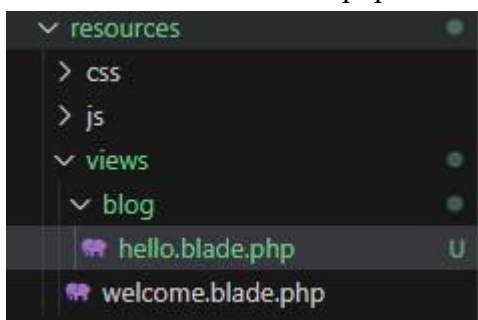


- View dalam direktori

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']); });
```

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

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





- 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']);
    }
}
```

```
12 class WelcomeController extends Controller{
13     0 references | 0 overrides
14     public function hello(){
15         return('Hello World');
16     }
17     0 references | 0 overrides
18     public function greeting(){
19         return view('blog.hello', ['name' => 'Rio Bagus']);
20     }
21 }
```

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

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

```
98 | 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.





- 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:

- Buka WelcomeController.php dan tambahkan ubah fungsi greeting.

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

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

```
22 class WelcomeController extends Controller{
23     0 references | 0 overrides
24     public function hello(){
25         return('Hello World');
26     }
27     1 reference | 0 overrides
28     public function greeting(){
29         return view('blog.hello')
30         ->with('name', 'Rio Bagus')
31         ->with('occupation', 'Astronaut');
    }
```

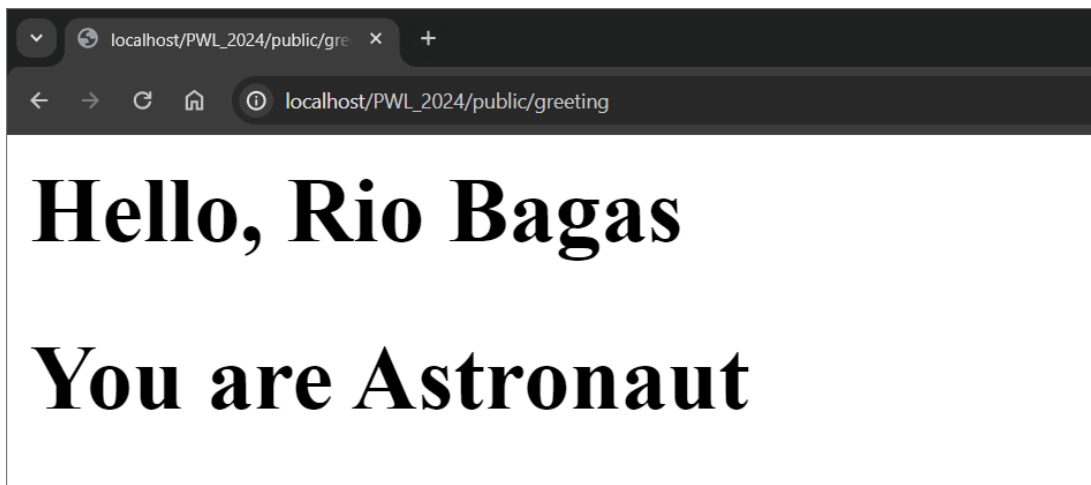


- b. Ubah hello.blade.php agar dapat menampilkan dua parameter.

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

```
9  <html>
10 <body>
11   <h1>Hello, {{ $name }}</h1>
12   <h1>You are {{ $occupation }}</h1>
13 </body>
14 </html>
```

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



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.

```
C:\Users\ASUS TUF>cd C:\laragon\www  
  
C:\laragon\www>composer create-project laravel/laravel="10.3.*" POS  
Creating a "laravel/laravel=10.3.*" project at "./POS"  
Installing laravel/laravel (v10.3.3)  
- Installing laravel/laravel (v10.3.3): Extracting archive  
Created project in C:\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 (0.11.0)  
- Locking carbonphp/carbon-doctrine-types (2.1.0)  
- Locking dflydev/dot-access-data (v3.0.2)  
- Locking doctrine/inflector (2.0.10)  
- Locking doctrine/lexer (3.0.1)  
- Locking dragonmantank/cron-expression (v3.3.3)  
- Locking egulias/email-validator (4.0.2)  
- Locking fakerphp/faker (v1.23.1)  
- Locking filp/whoops (2.15.4)  
- Locking fruitcake/php-cors (v1.3.0)  
- Locking graham-campbell/result-type (v1.1.2)  
- Locking guzzlehttp/guzzle (7.8.1)  
- Locking guzzlehttp/promises (2.0.2)  
- Locking guzzlehttp/psr7 (2.6.2)  
- Locking guzzlehttp/uri-template (v1.0.3)  
- Locking hamcrest/hamcrest-php (v2.0.1)  
- Locking laravel/framework (v10.46.0)
```



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

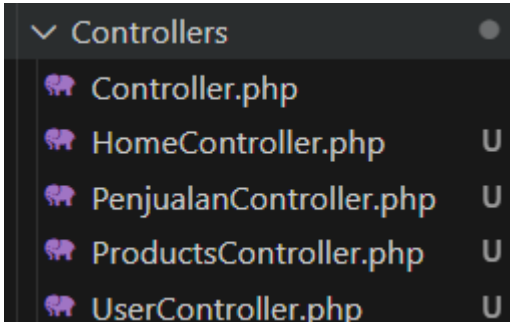
a. Controller

```
C:\laragon\www\POS>php artisan make:controller HomeController --resource
INFO Controller [C:\laragon\www\POS\app\Http\Controllers\HomeController.php] created successfully.

INFO Controller [C:\laragon\www\POS\app\Http\Controllers\ProductsController.php] created successfully.

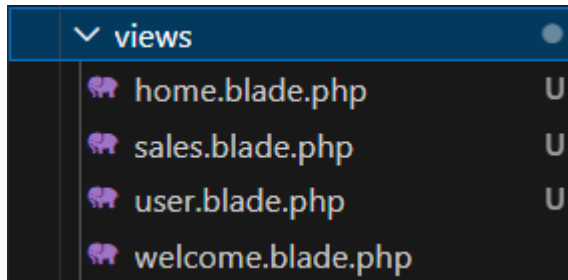
C:\laragon\www\POS>php artisan make:controller SalesController --resource
INFO Controller [C:\laragon\www\POS\app\Http\Controllers\SalesController.php] created successfully.

C:\laragon\www\POS>php artisan make:controller UserController --resource
INFO Controller [C:\laragon\www\POS\app\Http\Controllers\UserController.php] created successfully.
```

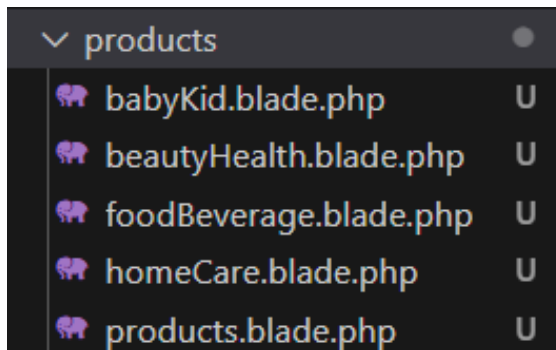




b. Views



c. Products



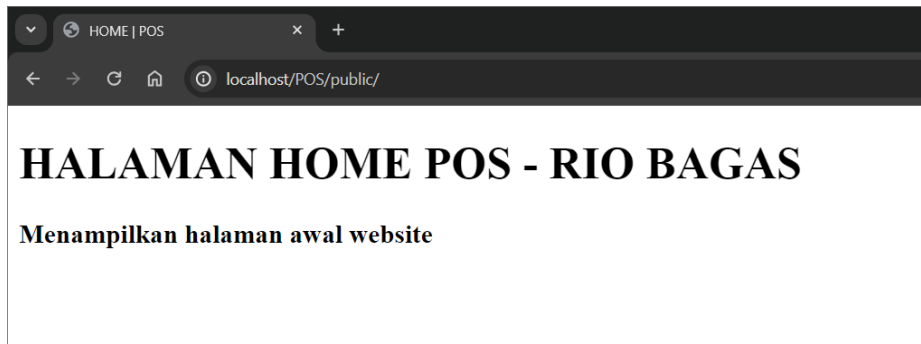
4. Route tersebut menjalankan fungsi pada Controller yang berbeda di setiap halaman.

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

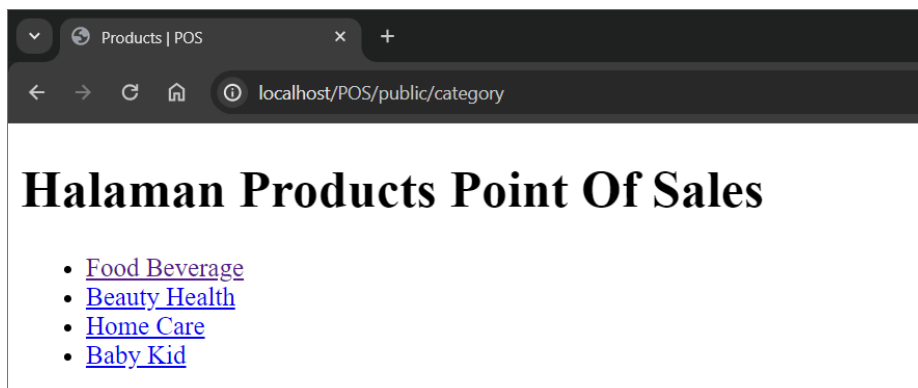


5. Fungsi pada Controller akan memanggil view sesuai halaman yang akan ditampilkan.

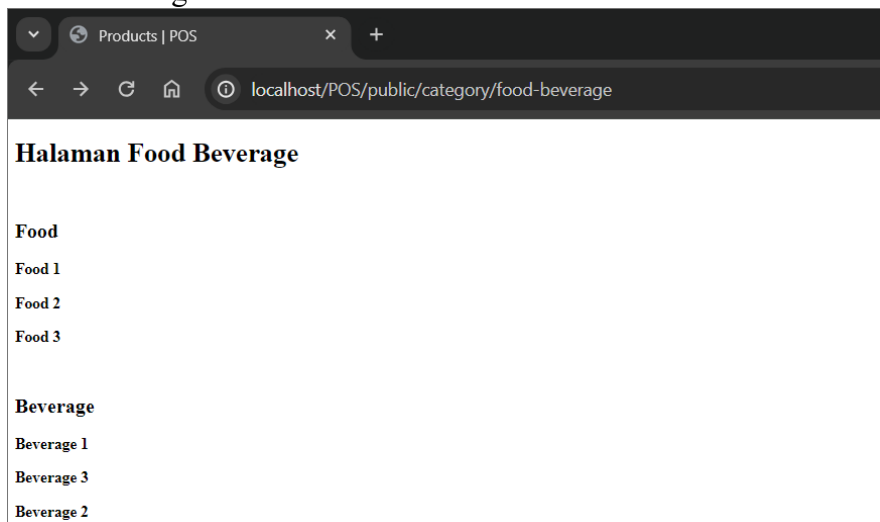
1. Halaman Home



2. Halaman Products

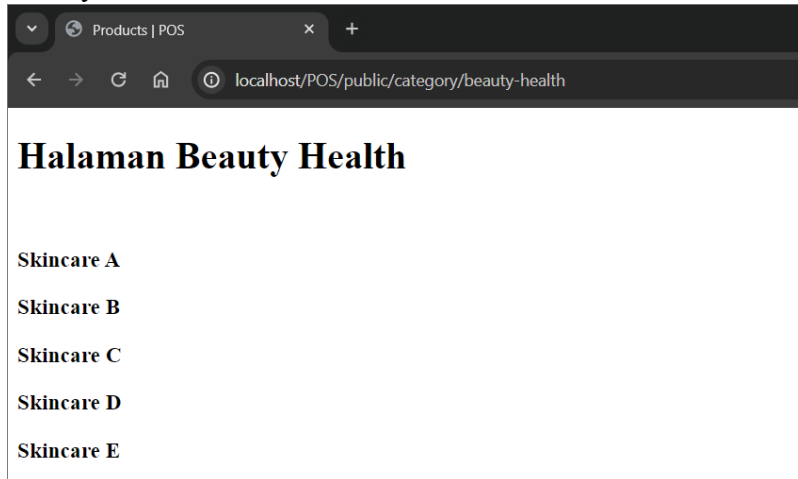


a. Food Beverage

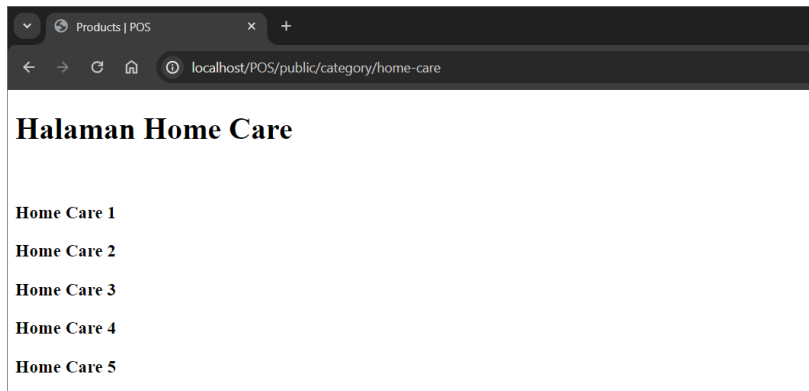




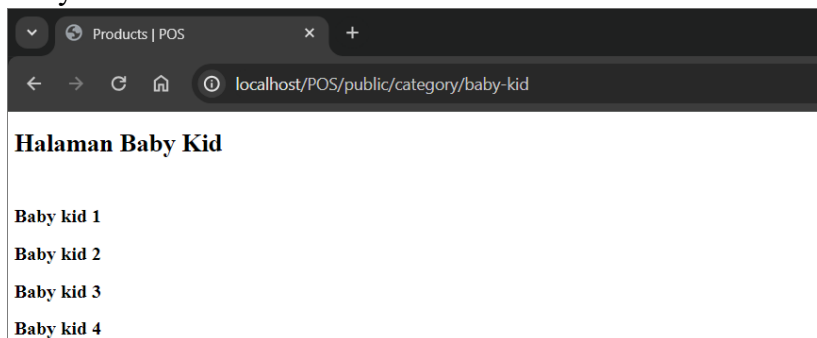
b. Beauty Health



c. Home Care

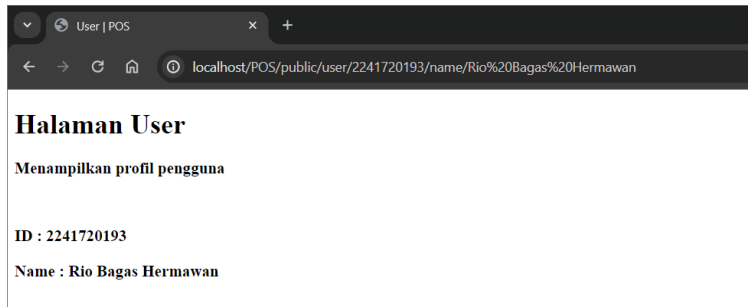


d. Baby Kid

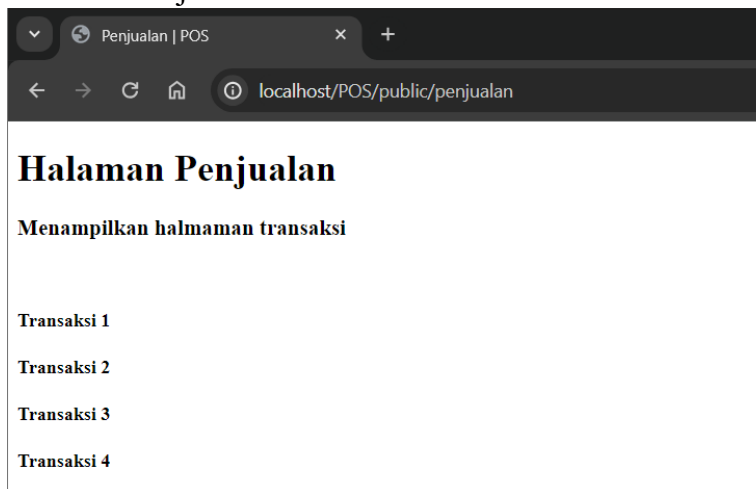




3. Halaman User

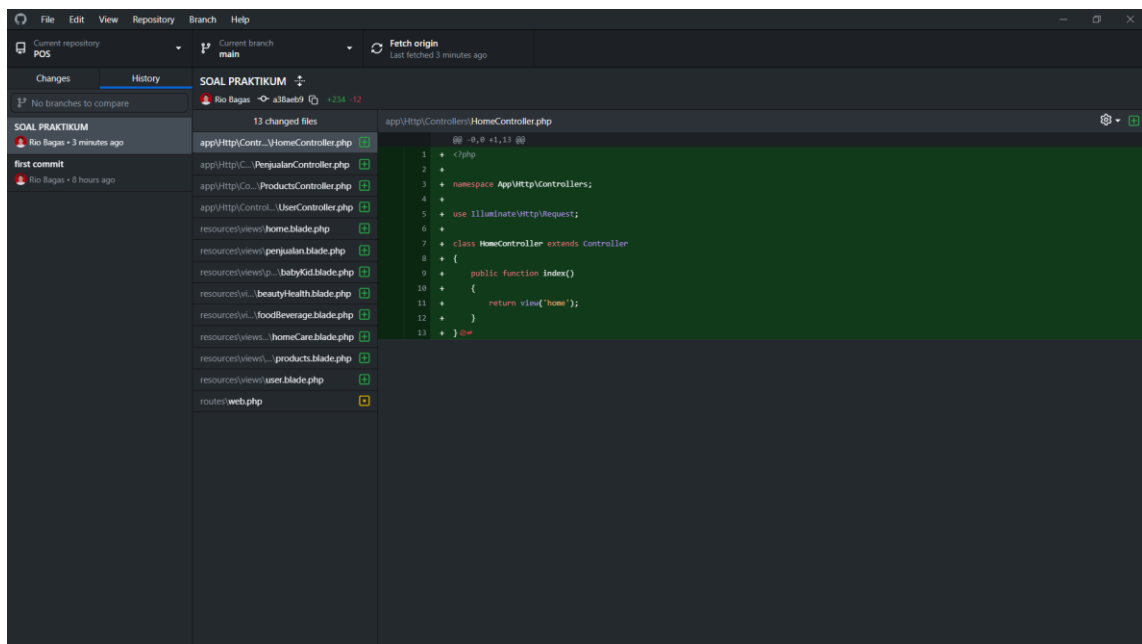
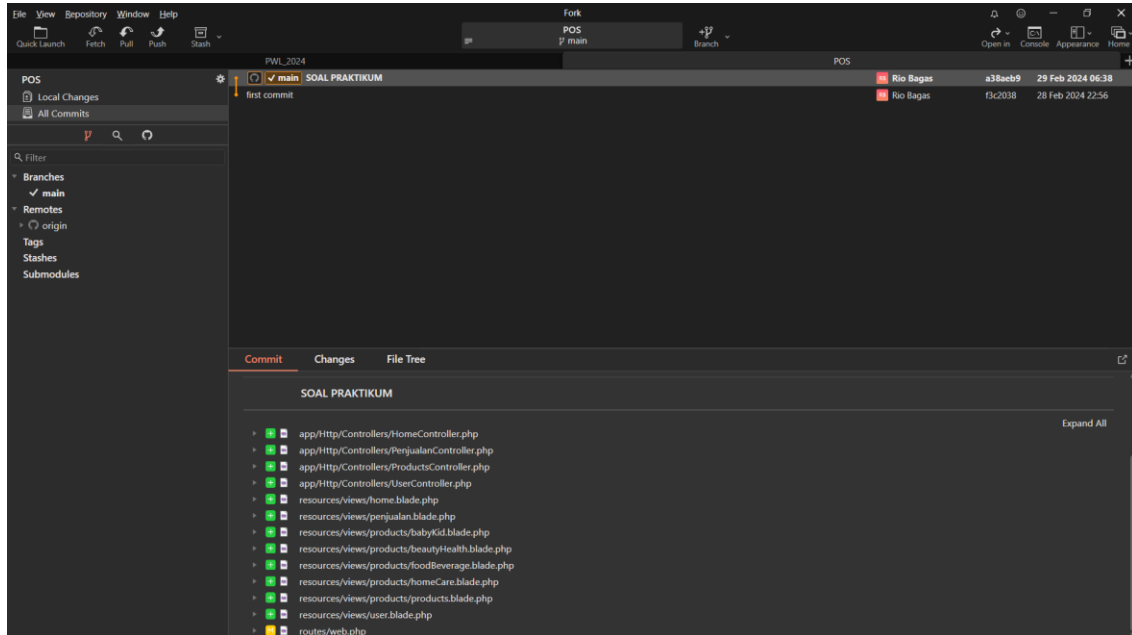


4. Halaman Penjualan





6. Simpan setiap perubahan yang dilakukan pada project POS pada Git, sinkronisasi perubahan ke Github.



*** Sekian, dan selamat belajar ***