

Nama : Misyhel Oktavia Br Nababan

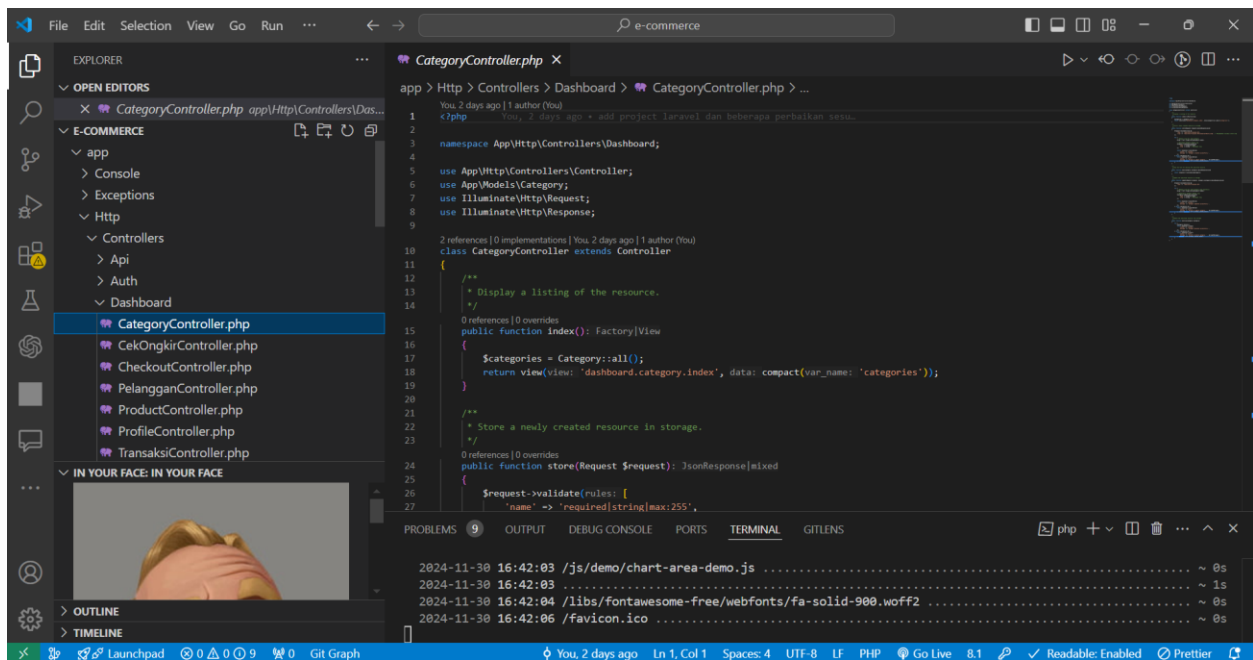
Kelas : PW 1

Kelompok : 5

## 1. Membuat Halaman Pengelolaan Katalog Kategori

Pertama dimulai dari katalog kategori controller sebagai berikut :

Karna fungsi nya laravel akan akses terlebih dahulu controller yg di request user dan lalu laravel akan mengarahkan user ke routes nya



```
CategoryController.php
app > Http > Controllers > Dashboard > CategoryController.php > ...
1 <?php
2
3 namespace App\Http\Controllers\Dashboard;
4
5 use App\Http\Controllers\Controller;
6 use App\Models\Category;
7 use Illuminate\Http\Request;
8 use Illuminate\Http\Response;
9
10
11
12
13 /**
14  * Display a listing of the resource.
15  */
16 0 references | 0 overrides
17 public function index(): Factory|View
18 {
19     $categories = Category::all();
20     return view('dashboard.category.index', data: compact('var_name: 'categories'));
21 }
22 /**
23  * Store a newly created resource in storage.
24  */
25 0 references | 0 overrides
26 public function store(Request $request): JsonResponse|mixed
27 {
28     $request->validate(rules: [
29         'name' => 'required|string|max:255',
30     ]);
31 }
```

2024-11-30 16:42:03 /js/demo/chart-area-demo.js ..... ~ 0s  
2024-11-30 16:42:03 ..... ~ 1s  
2024-11-30 16:42:04 /libs/fontawesome-free/webfonts/fa-solid-900.woff2 ..... ~ 0s  
2024-11-30 16:42:06 /favicon.ico ..... ~ 0s

2. Lalu membuat routes web.php sebagai berikut :

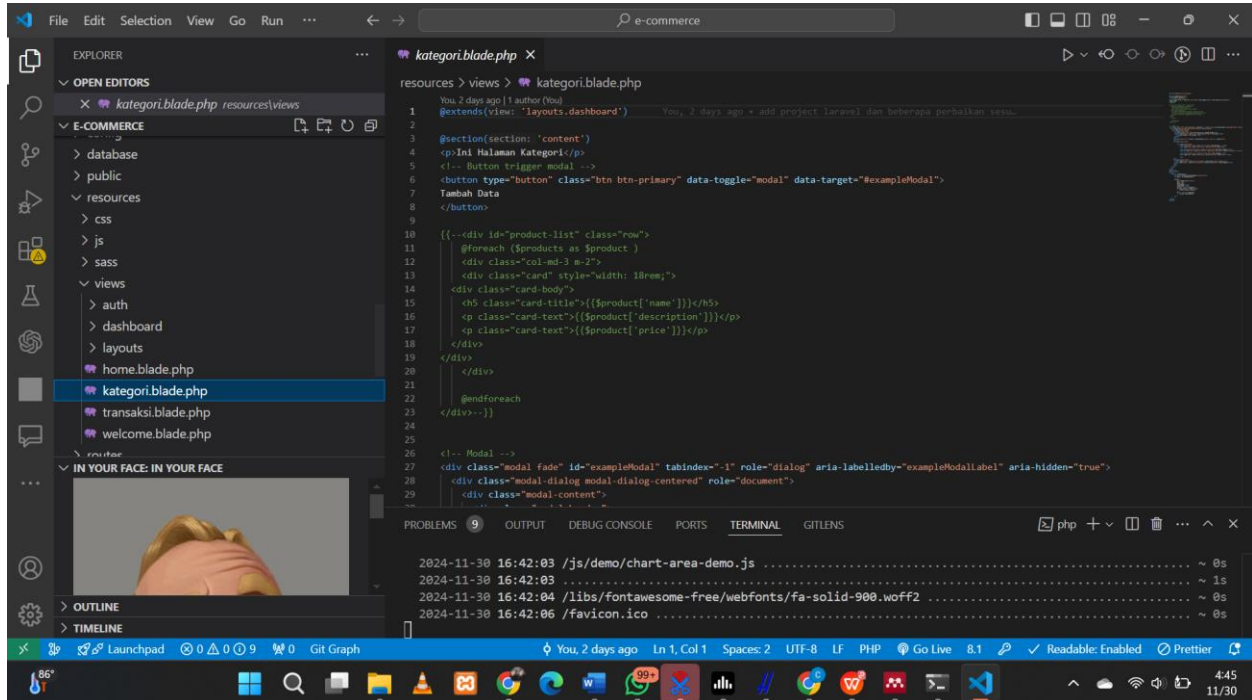
Setelah itu routes akan mengarahkan user ke halaman view yang telah kita siapkan sebagai tampilan user sebagai pengelolaan data katalog kategori

```
routes > web.php
31 Route::prefix('dashboard')->middleware(['auth', 'profile.check'])->name('dashboard.')->group(callback: function () {
47
48 Route::resource(name: 'products', controller: ProductController::class)->names(names: 'products');
49
50 Route::redirect(uri: 'test', destination: 'products');
51 });
52
53 Route::resource(name: 'dashboard/profile', controller: ProfileController::class)->only(methods: [
54 'index', 'update'
55 ])->names(names: 'dashboard.profile');
56
57
58 Route::get(uri: '/home', action: [App\Http\Controllers\HomeController::class, 'index'])->name(name: 'home');
59
60 // Route group dengan prefix 'admin'
61 Route::prefix('admin')->group(callback: function () { void {
62 Route::get(uri: '/kontak kami', action: function () { Factory|View {
63 return view(view: 'dashboard.kontak kami');
64 })->name(name: 'kontak kami');
65 });
66 });
67
68 Route::get(uri: '/parse-data/(nama_lengkap)/(email)/(jenis_kelamin)', action: [ParsingDataController::class, 'parseData'])
69 ->name(name: 'parse-data'); // You, 2 days ago • add project laravel dan beberapa perbaikan seso...
70
71 Route::get(uri: '/kategori', action: function () { Factory|View {
72 return view(view: 'kategori');
73 })->name(name: 'kategori');
74 });
75
76 Route::get(uri: '/transaksi', action: function () { Factory|View {
77 return view(view: 'transaksi');
78 });
79 });
```

2024-11-30 16:31:35 /libs/fontawesome-free/webfonts/fa-solid-900.woff2 ..... ~ 1s  
2024-11-30 16:31:35 /js/demo/chart-area-demo.js ..... ~ 1s  
2024-11-30 16:31:35 /img/baju.png ..... ~ 1s  
2024-11-30 16:31:37 /favicon.ico ..... ~ 0s

3. Setelah itu kita buat view dari tampilan untuk pengelolaan katalog kategori sebagai berikut :

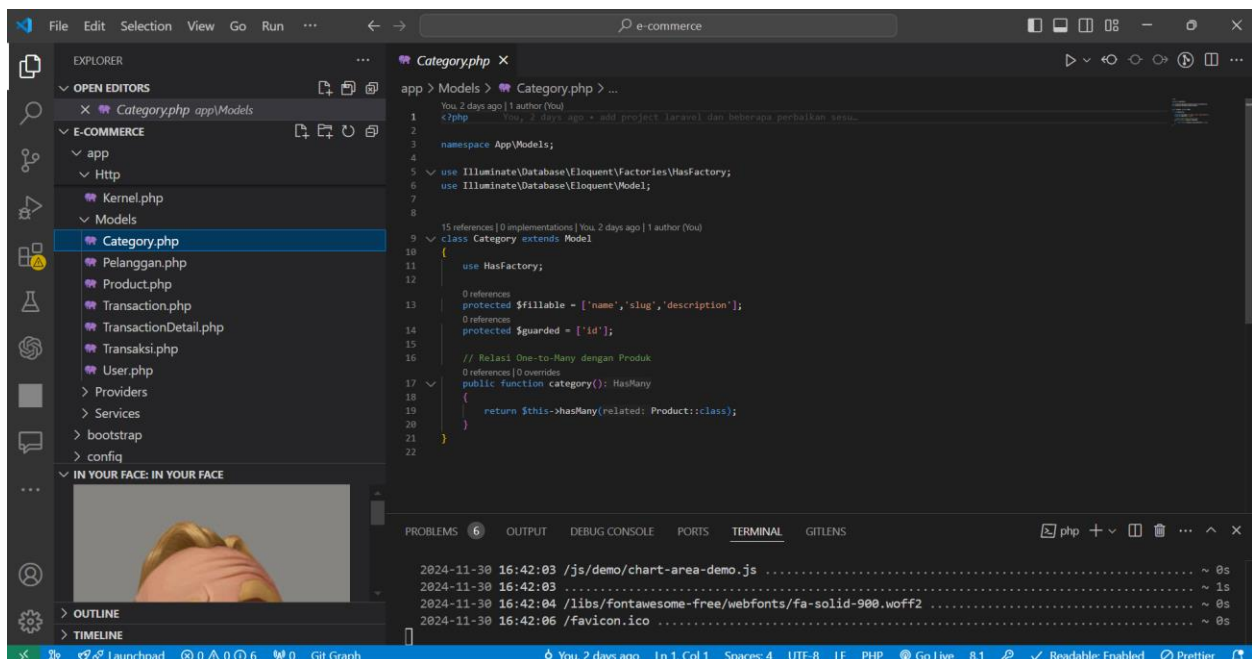
Dibawah ini adalah screenshoot kode pembuatan view untuk pengelolaan katalog kategori dan selanjutnya akan dibuatkan kode untuk validasi ketika dilakukan crud category yaitu pada bagian model dari katalog kategori tsb



The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'resources > views' directory is expanded, and 'kategori.blade.php' is selected. The main editor displays the Blade template code for the category management view. The code includes a section for the content, a button to trigger a modal, and a modal for adding data. The modal contains a table with columns for name, description, and price, and a button to add data. The code also includes a modal for editing data.

```
1 @extends('layouts.dashboard')
2
3 @section('content')
4 <!-- Button trigger modal -->
5 <button type="button" class="btn btn-primary" data-toggle="modal" data-target="#exampleModal">
6   Tambah Data
7 </button>
8
9
10 <!-- Modal -->
11 <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
12   <div class="modal-dialog modal-dialog-centered" role="document">
13     <div class="modal-content">
14       <div class="card">
15         <div class="card-body">
16           <table>
17             <thead>
18               <tr>
19                 <th>Name</th>
20                 <th>Description</th>
21                 <th>Price</th>
22             </thead>
23             <tbody>
24               <tr>
25                 <td>{{ $product['name'] }}</td>
26                 <td>{{ $product['description'] }}</td>
27                 <td>{{ $product['price'] }}</td>
28               </tr>
29             </tbody>
30           </table>
31         </div>
32       </div>
33     </div>
34   </div>
35 </div>
36
37 @endforeach
38 </div>
39
40 <!-- Modal -->
41 <div class="modal fade" id="exampleModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
42   <div class="modal-dialog modal-dialog-centered" role="document">
43     <div class="modal-content">
44       <div class="card">
45         <div class="card-body">
46           <table>
47             <thead>
48               <tr>
49                 <th>Name</th>
50                 <th>Description</th>
51                 <th>Price</th>
52             </thead>
53             <tbody>
54               <tr>
55                 <td>{{ $product['name'] }}</td>
56                 <td>{{ $product['description'] }}</td>
57                 <td>{{ $product['price'] }}</td>
58               </tr>
59             </tbody>
60           </table>
61         </div>
62       </div>
63     </div>
64   </div>
65 </div>
```

4. Pada bagian model category dibawah ini berfungsi sebagai validasi yg dilakukan untuk melakukan pengelolaan crud pada katalog kategori

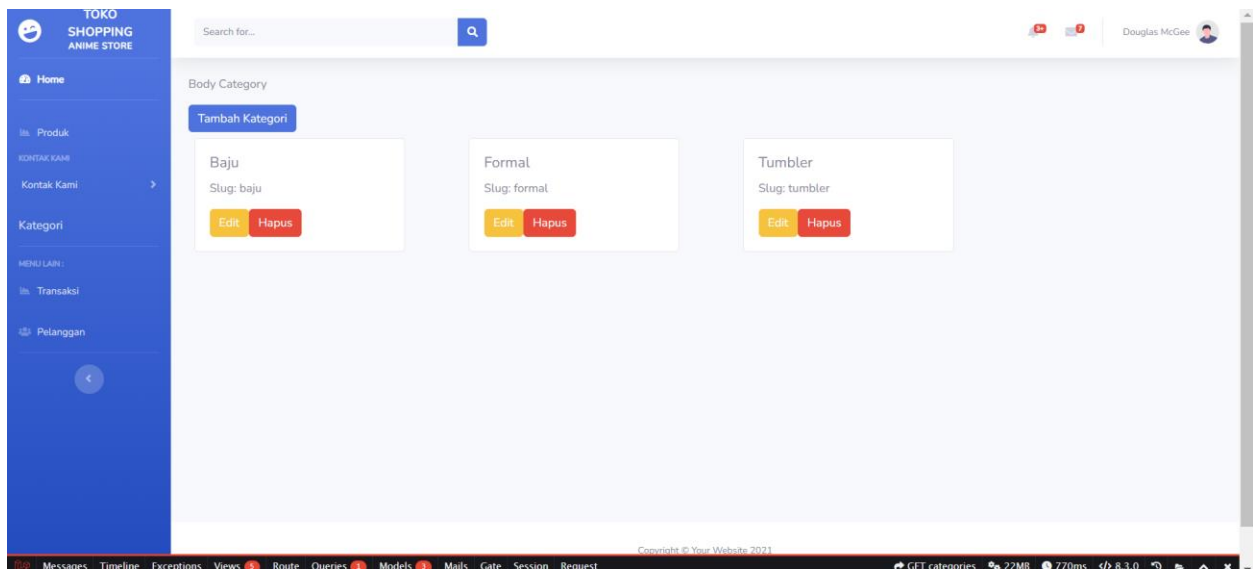


The screenshot shows the Visual Studio Code editor with the file explorer on the left. The 'app > Models' directory is expanded, and 'Category.php' is selected. The main editor displays the PHP code for the Category model. The code includes the namespace, imports, and the class definition. The class extends the Model class and has a fillable array, a guarded array, and a belongsTo relationship with the Product model.

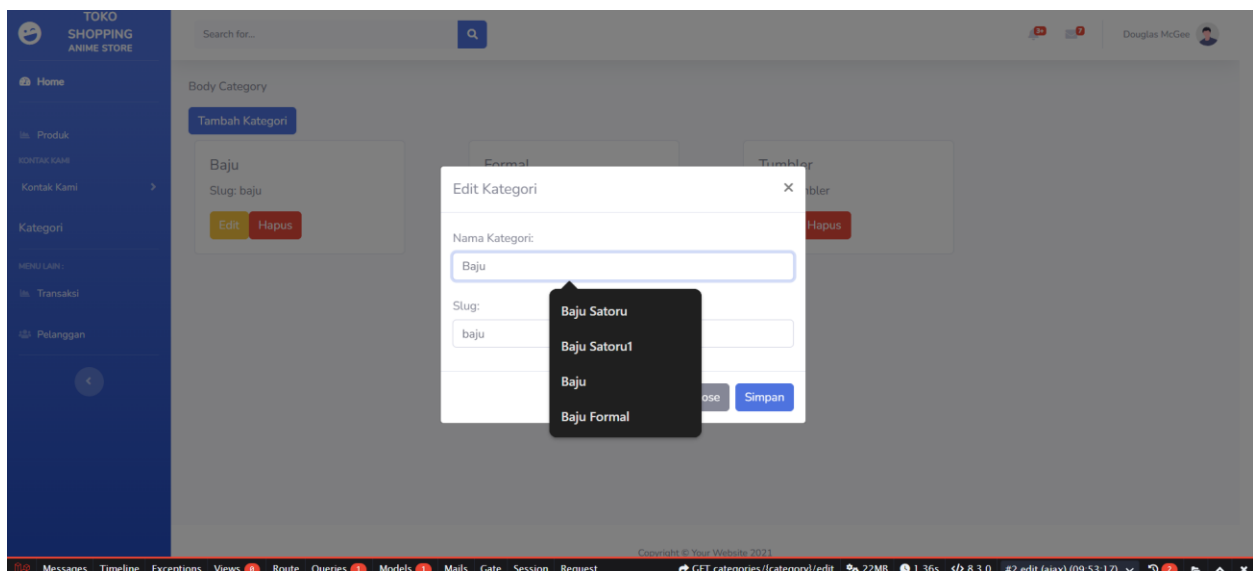
```
1 namespace App\Models;
2
3 use Illuminate\Database\Eloquent\Factories\HasFactory;
4 use Illuminate\Database\Eloquent\Model;
5
6 class Category extends Model
7 {
8     use HasFactory;
9
10     protected $fillable = ['name', 'slug', 'description'];
11     protected $guarded = ['id'];
12
13     // Relasi One-to-Many dengan Produk
14     public function category(): HasMany
15     {
16         return $this->hasMany(related: Product::class);
17     }
18 }
```

5. Tampilan crud katalog kategori dan edit katalog kategori yang telah dibuat sebagai berikut :

Tampilan menampilkan katalog kategori setelah ditambah katalog kategori

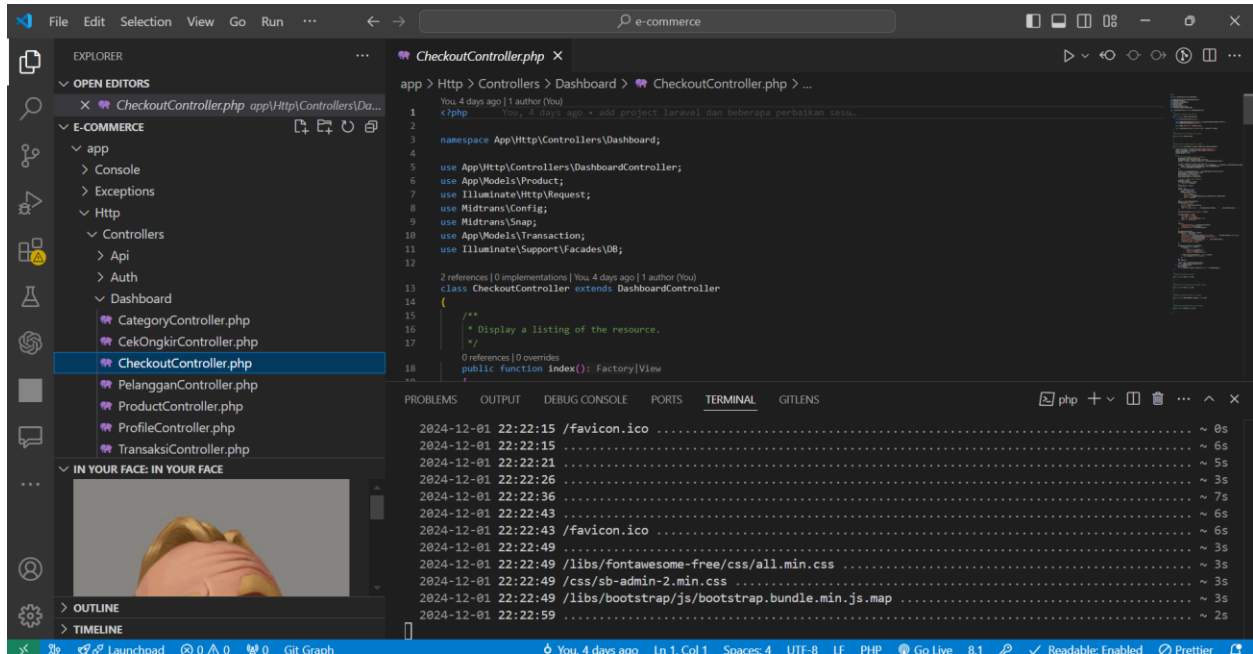


Tampilan edit katalog kategori yang telah dirancang sebagai berikut :

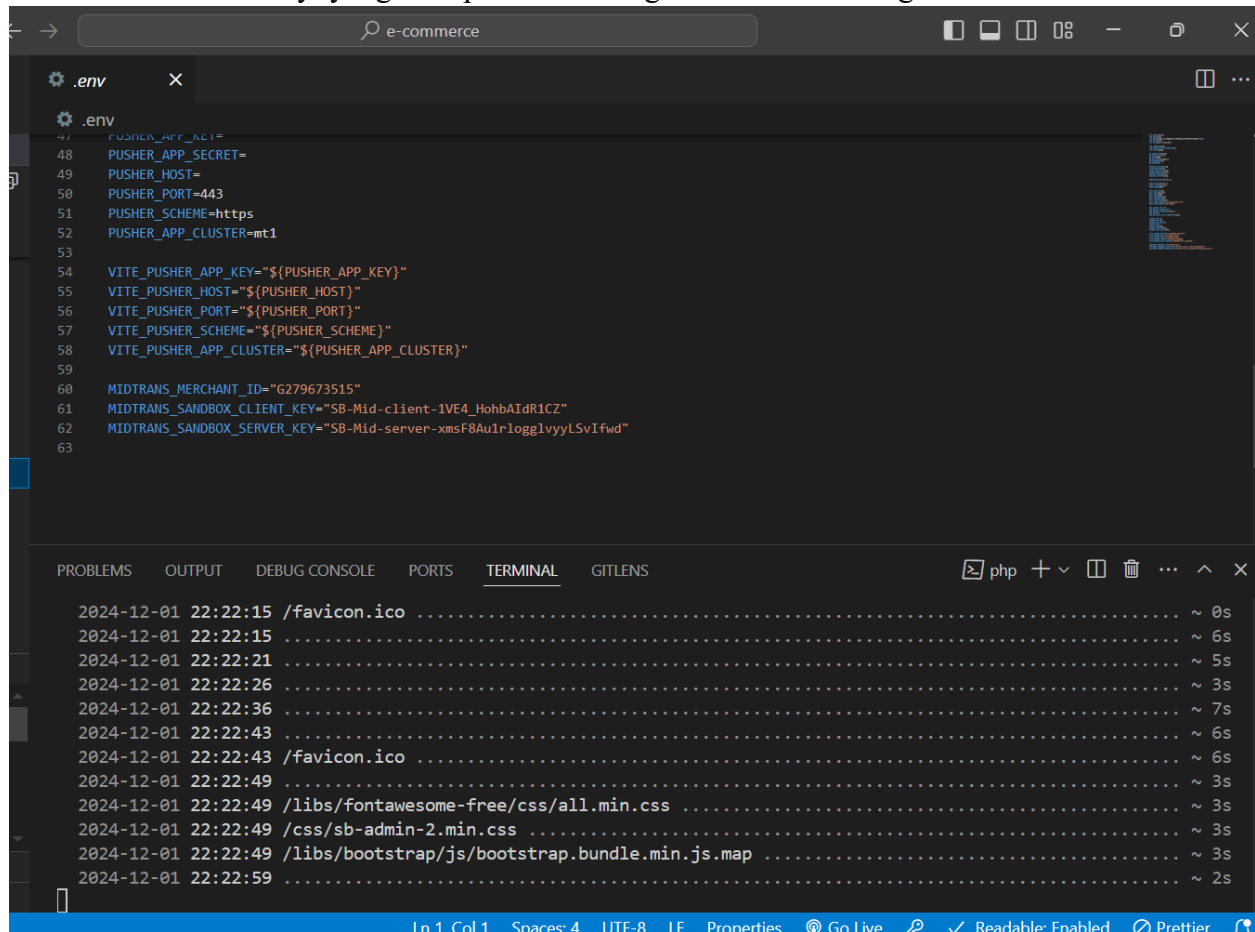


## 6. Membuat Checkout produk

Hal yang pertama adalah melakukan pembuatan checkout controller dengan screenshoot sebagai berikut :



Setelah itu kita integrasikan MIDTRANS utk lakukan halaman pembayaran dengan sesuaikan terlebih dahulu API key yang terdapat di env dengan screenshot sebagai berikut :



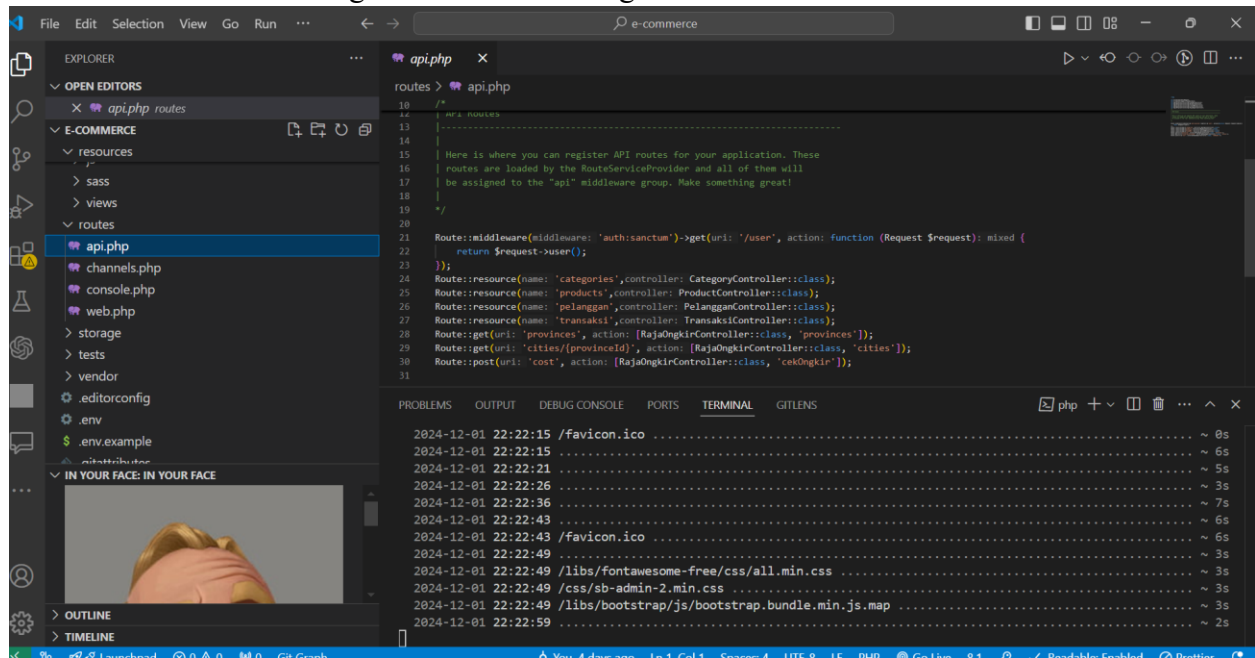
The screenshot shows a Visual Studio Code editor with a file named `.env` open. The file contains environment variables for a Pusher service and MIDTRANS payment gateway. Below the editor, the TERMINAL panel shows the output of a file watcher, listing various assets like `/favicon.ico` and `/libs/fontawesome-free/css/all.min.css` with their respective sizes.

```
.env
47: PUSHER_APP_KEY=
48: PUSHER_APP_SECRET=
49: PUSHER_HOST=
50: PUSHER_PORT=443
51: PUSHER_SCHEME=https
52: PUSHER_APP_CLUSTER=mt1
53:
54: VITE_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
55: VITE_PUSHER_HOST="${PUSHER_HOST}"
56: VITE_PUSHER_PORT="${PUSHER_PORT}"
57: VITE_PUSHER_SCHEME="${PUSHER_SCHEME}"
58: VITE_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
59:
60: MIDTRANS_MERCHANT_ID="G279673515"
61: MIDTRANS_SANDBOX_CLIENT_KEY="SB-Mid-client-1VE4_HohbAIdR1CZ"
62: MIDTRANS_SANDBOX_SERVER_KEY="SB-Mid-server-xmsF8Au1r1ogg1vvyL5vIfwd"
63:
```

```
2024-12-01 22:22:15 /favicon.ico ..... ~ 0s
2024-12-01 22:22:15 ..... ~ 6s
2024-12-01 22:22:21 ..... ~ 5s
2024-12-01 22:22:26 ..... ~ 3s
2024-12-01 22:22:36 ..... ~ 7s
2024-12-01 22:22:43 ..... ~ 6s
2024-12-01 22:22:43 /favicon.ico ..... ~ 6s
2024-12-01 22:22:49 ..... ~ 3s
2024-12-01 22:22:49 /libs/fontawesome-free/css/all.min.css ..... ~ 3s
2024-12-01 22:22:49 /css/sb-admin-2.min.css ..... ~ 3s
2024-12-01 22:22:49 /libs/bootstrap/js/bootstrap.bundle.min.js.map ..... ~ 3s
2024-12-01 22:22:59 ..... ~ 2s
```

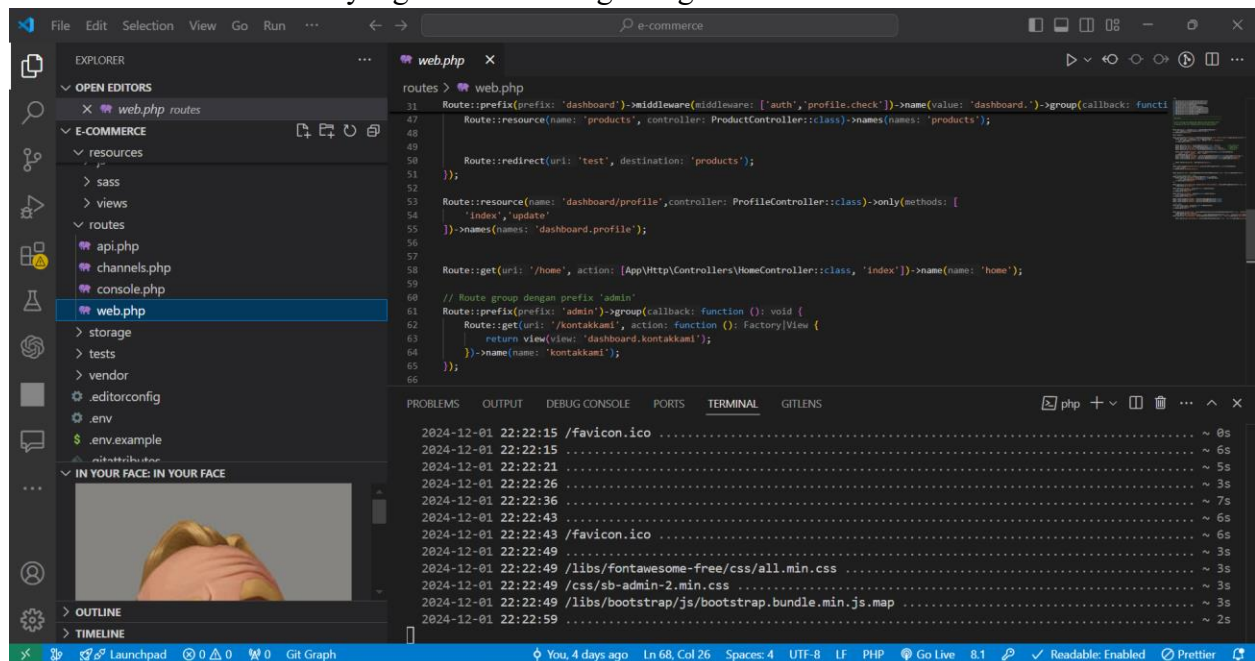
Ln 1, Col 1 Spaces: 4 UTF-8 LF Properties @ Go Live Readable: Enabled Prettier

Setelah itu kita rancang api.php untuk mendaftarkan routes api pada aplikasi yang kita rancang yang dimana api ini akan mengarahkan route dari request akses yang diakses user terlebih dahulu ke checkout controller dengan screenshoot sebagai berikut :



```
routes > api.php
10 /*
11 |  * API Routes
12 |  *
13 |  * Here is where you can register API routes for your application. These
14 |  * routes are loaded by the RouteServiceProvider and all of them will
15 |  * be assigned to the "api" middleware group. Make something great!
16 |  *
17 |  */
18
19
20
21 Route::middleware('auth:sanctum')->get('/user', action: function (Request $request): mixed {
22     return $request->user();
23 });
24
25 Route::resource(name: 'categories', controller: CategoryController::class);
26 Route::resource(name: 'products', controller: ProductController::class);
27 Route::resource(name: 'pelanggan', controller: PelangganController::class);
28 Route::resource(name: 'transaksi', controller: TransaksiController::class);
29 Route::get(uri: 'provinces', action: [RajaOngkirController::class, 'provinces']);
30 Route::get(uri: 'cities/{provinceId}', action: [RajaOngkirController::class, 'cities']);
31 Route::post(uri: 'cost', action: [RajaOngkirController::class, 'cekOngkir']);
```

Setelah itu membuat routes web.php yang untuk mengarahkan user dan dapat diarahkan ke dalam view dari checkout yang telah dirancang sebagai berikut :



```
routes > web.php
31 Route::prefix(prefix: 'dashboard')->middleware(middleware: ['auth', 'profile.check'])->name(value: 'dashboard')->group(callback: function () {
32     Route::resource(name: 'products', controller: ProductController::class)->names(names: 'products');
33 });
34
35 Route::redirect(uri: 'test', destination: 'products');
36
37
38 Route::resource(name: 'dashboard/profile', controller: ProfileController::class)->only(methods: [
39     'index', 'update'
40 ])->names(names: 'dashboard-profile');
41
42
43 Route::get(uri: '/home', action: [App\Http\Controllers\HomeController::class, 'index'])->name(name: 'home');
44
45 // Route group dengan prefix 'admin'
46 Route::prefix(prefix: 'admin')->group(callback: function () { void {
47     Route::get(uri: '/kontakami', action: function () { Factory\View {
48         return view(view: 'dashboard.kontakami');
49     }->name(name: 'kontakami');
50 });
51 });
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
```

Setelah itu , kita rancang halaman tampilan yang akan diakses user setelah kita membuatnya pada view.php dan screenshoot kode sebagai berikut :

