**LAPORAN PRAKTIKUM**

**MATA KULIAH PEMROGRAMAN WEB LANJUT**

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

**JOBSHEET 5: BLADE VIEW, WEB TEMPLATING (ADMIN LTE), DATATABLES**

****

Nama : Yonanda Mayla Rusdiaty

NIM : 2341760184

Prodi : D-IV Sistem Informasi Bisnis

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

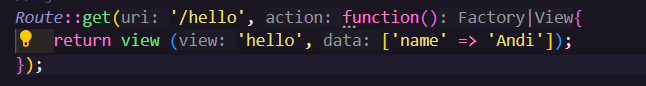
**2024**

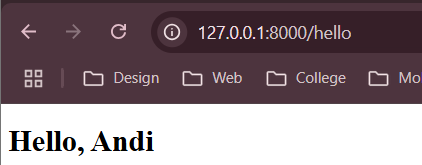
1. **VIEW**

Karena Laravel menggunakan templating engine bawaan Blade, maka setiap file View

diakhiri dengan .blade.php. Misal: index.blade.php, [home.blade.php](http://home.blade.php), product.blade.php. Contoh sederhana dari view dengan nama file hello.blade.php adalah sebagai berikut.





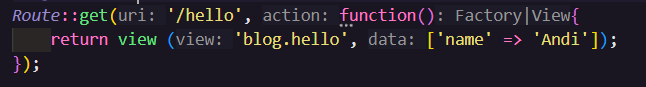


Setiap kita membuat view, kita mungkin akan menampilkannya melalui router atau controller menggunakan global view helper. Selain menggunakan global view helper kita juga dapat menggunakan view facade untuk menampilkan view



1. **View di 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, sehingga syntax dalam route akan seperti berikut



1. **Menampilkan view dari controller**

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





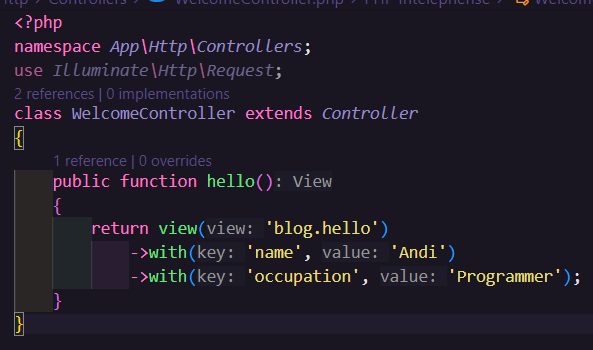
1. **Meneruskan data ke view**

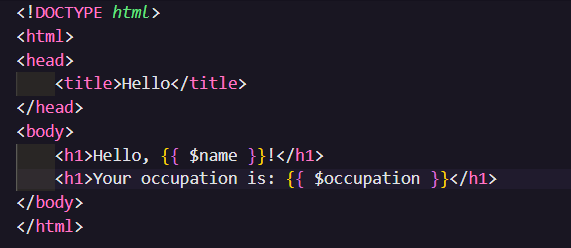
Seperti yang anda lihat di contoh sebelumnya, anda dapat meneruskan data array ke view agar data tersebut tersedia untuk view

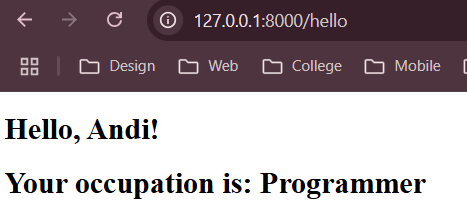


Saat meneruskan informasi dengan cara ini, data harus berupa array dengan pasangan kunci / nilai. Setelah memberikan data ke view, anda kemudian dapat mengakses setiap nilai dalam view menggunakan kunci data seperti: <?php echo $name; ?> atau {{$name}}.

Sebagai alternatif untuk meneruskan array data lengkap ke fungsi view helper, anda dapat menggunakan metode with untuk menambahkan bagian data individual ke view. Metode with mengembalikan instance view objek sehingga anda dapat melanjutkan rangkaian metode sebelum mengembalikan tampilan

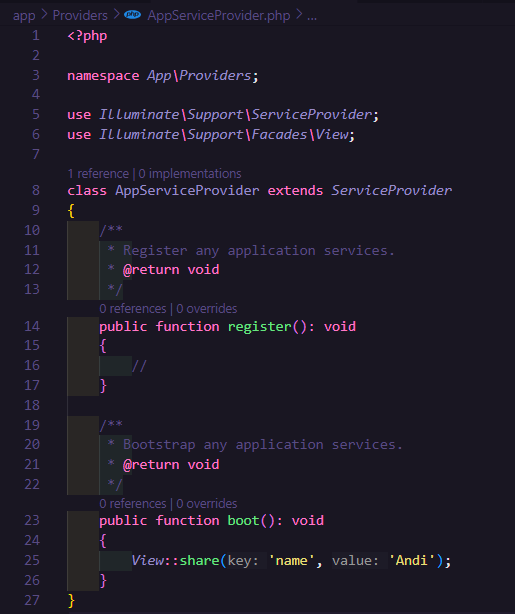






1. **Berbagi data dengan semua view**

Terkadang, anda mungkin perlu berbagi data dengan semua tampilan yang dirender oleh aplikasi anda. anda dapat melakukannya dengan menggunakan metode berbagi tampilan fasad. Biasanya, anda harus melakukan panggilan ke share methodi dalam metode boot penyedia layanan. anda bebas menambahkannya ke kelas App\Providers\AppServiceProvider atau membuat penyedia layanan terpisah untuk menampungnya

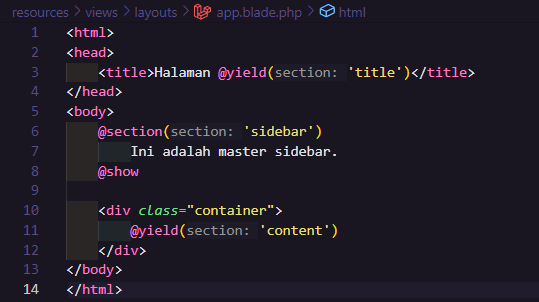


1. **BLADE LAYOUT, SECTION, DAN COMPONENT**

Dalam membuat suatu tampilan, seringkali ada beberapa bagian yang sama dan selalu ditampilkan di setiap halaman view. Bagian-bagian ini dapat dibuat template sehingga tidak perlu dibuat berulang kali di setiap halaman view.

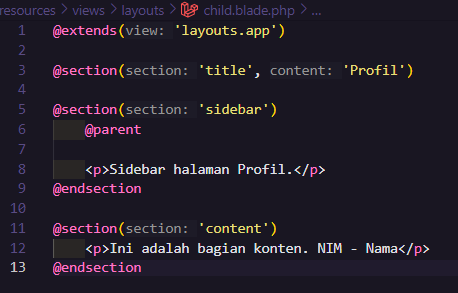
1. **Layout dan Section**

Berikut adalah contoh dari sebuah layout dari master/parent view.

****

Kode yang merupakan layout global, mendefinisikan title, sidebar, dan content yang dapat diisi oleh child view yang menggunakannya.

Berikut adalah contoh child view yang menggunakan layout app.blade.php.

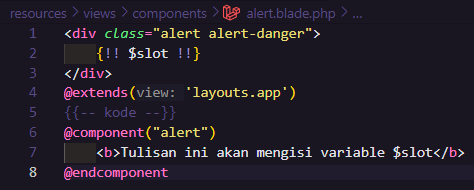


Keterangan: @extends(“layouts.app”) digunakan untuk menjadikan file view app.blade.php. sebagai master view. Kemudian kita isi section title dengan “Profil” yang akan dirender sebagai <title> Halaman Profil </title>. Kemudian kita juga mengisi section sidebar menggunakan directive @parent. Sehingga ditampilkan konten sidebar parent/master serta menampilkan kalimat “Sidebar halaman Profil”. Setelah itu kita juga isi section “content” dengan tulisan “Ini adalah bagian konten. NIM - Nama”.

1. **Component**

Component berfungsi untuk membuat view yang dapat kita gunakan berulang kali. Berbeda dengan layout yang bertindak sebagai master, component dapat dianggap sebagai child view yang bisa kita pakai di view lain yang membutuhkannya. Misalnya dalam pengembangan sebuah aplikasi kita akan membutuhkan view untuk alert yang memberikan notifikasi kepada

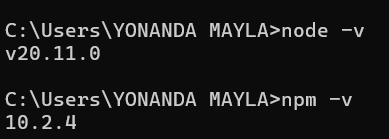
pengguna aplikasi terkait informasi, peringatan ataupun pesan error. Alert ini akan digunakan berulang kali di aplikasi. Oleh karena itu kita bisa membuatnya sebagai component yang bisa digunakan di view lainnya. Berikut adalah contoh component untuk membuat alert



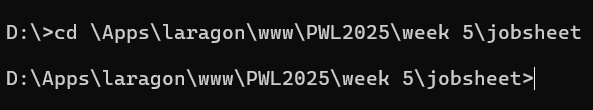
1. **BOOTSTRAP PADA LARAVEL**

Secara default, Laravel menggunakan NPM untuk menginstal paket frontend ini. Untuk menggunakannya, dapat dilakukan langkah-langkah berikut:

1. Instalasi Node.js
2. Memeriksa instalasi Node.js dan NPM

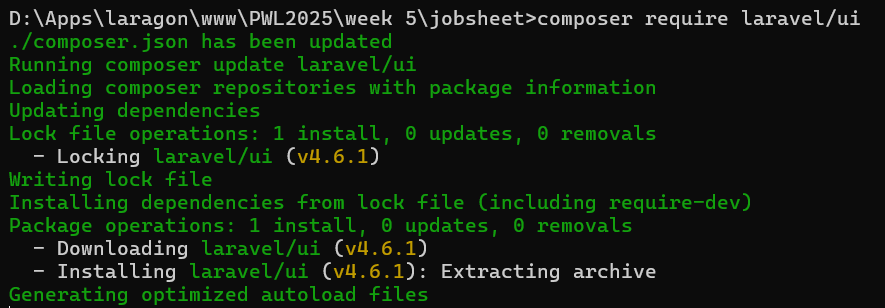


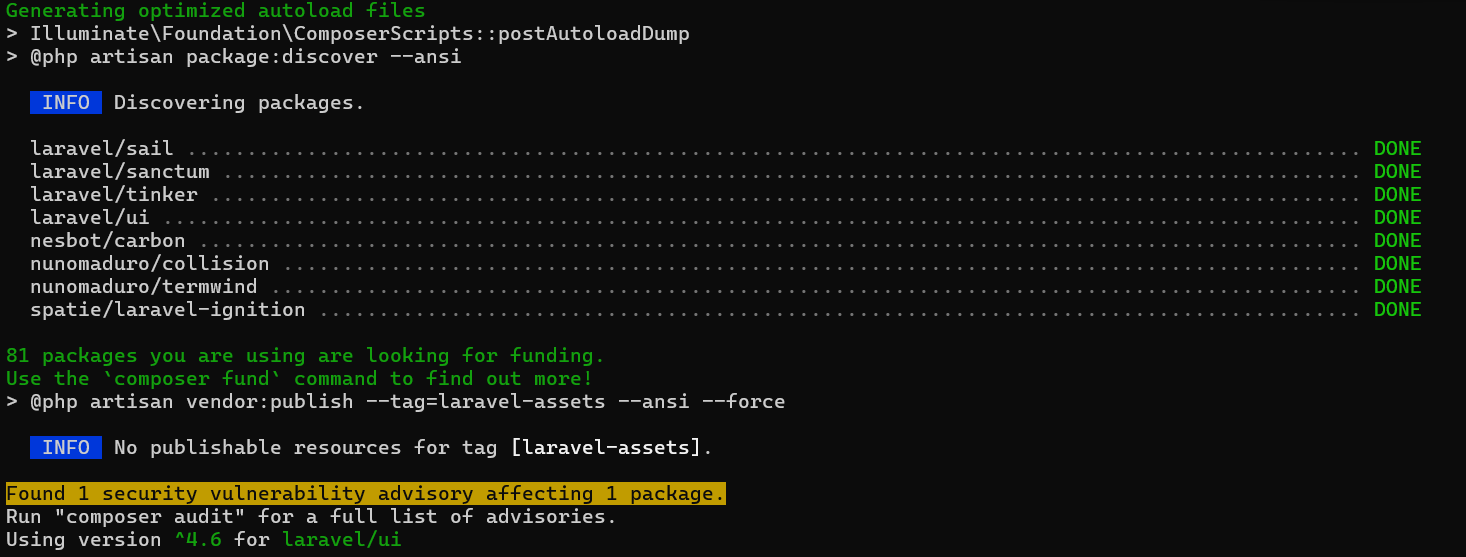
1. Kemudian melalui command prompt, ubah direktori ke dalam direktori project Laravel yang telah dibuat sebelumnya.



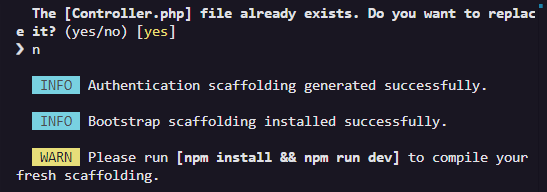
1. Scaffolding Bootstrap yang disediakan oleh Laravel terletak di dalam paket Composer, yang dapat diinstal menggunakan Composer:laravel/ui.

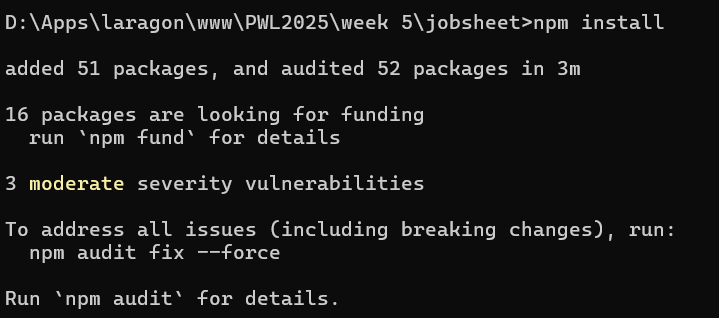
composer require laravel/ui

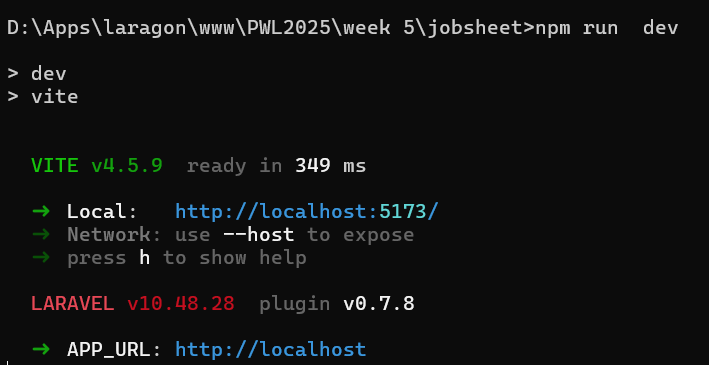




1. Setelah paket diinstal, kita dapat menginstal scaffolding frontend menggunakan perintah Artisan:laravel/ui



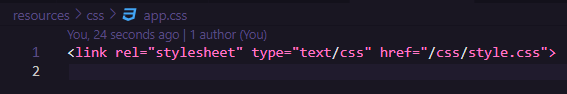
1. Sebelum mengompilasi CSS, install dependensi frontend proyek Anda menggunakan Node package manager (NPM) :
2. Setelah paket diinstal, Anda dapat menggunakan perintah berikut untuk mengompilasi aset Anda



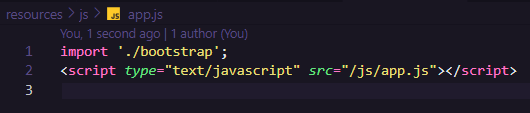
1. **MENGGUNAKAN TEMPLATE BOOTSTRAP DI LARAVEL**

Sebenarnya pada project laravel, sudah ada file css bootstrap secara default pada pertama kali kita menginstall laravel. Letaknya ada pada file app.css dalam folder css. Anda dapat langsung menggunakannya dengan menghubungkan file app.css tersebut dengan file view anda

Untuk mengubungkan file css ke laravel anda dapat menggunakan syntax berikut:

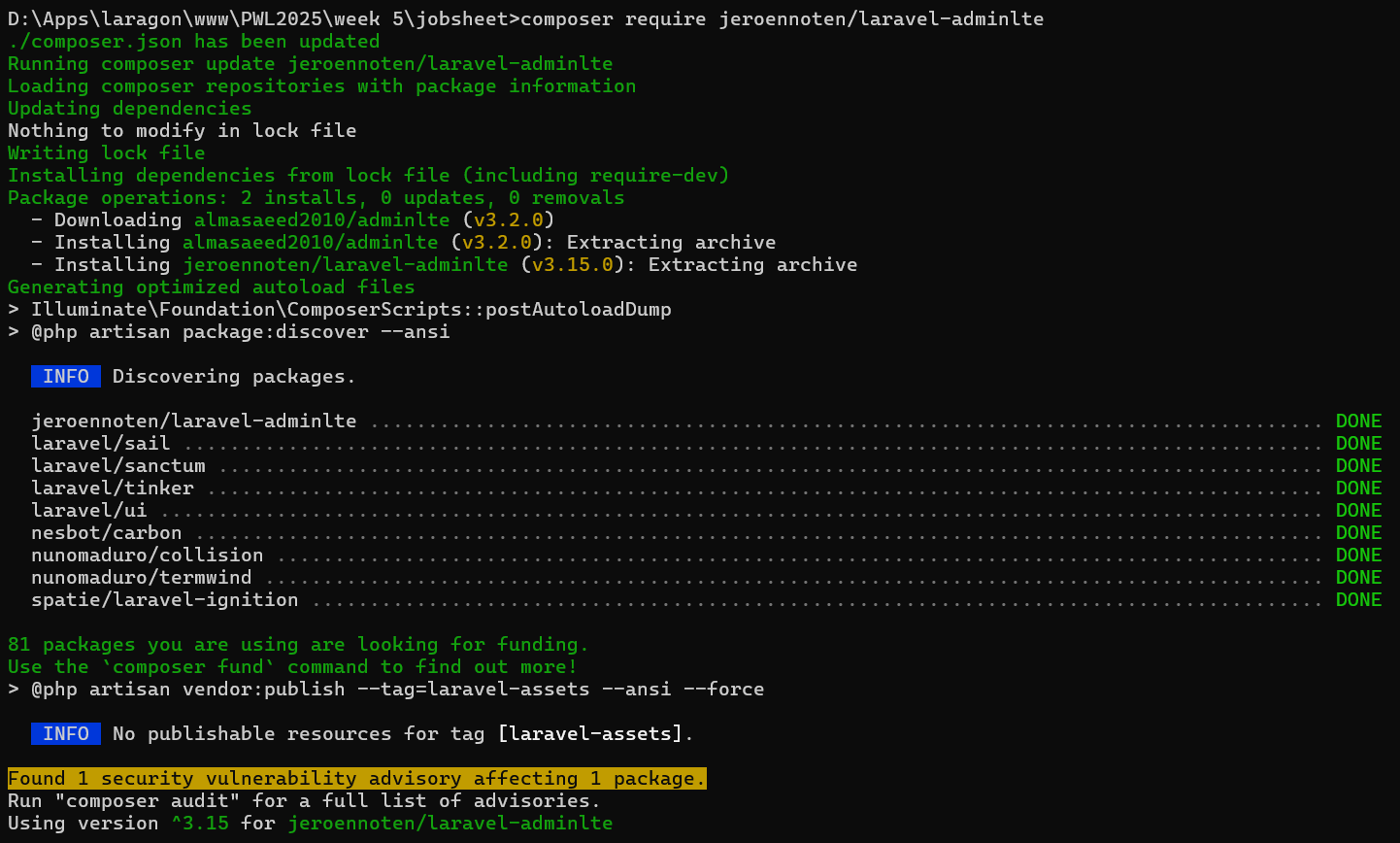


Dan file JSnya:

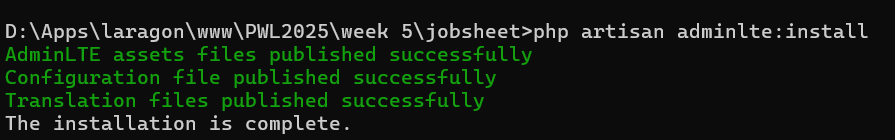


**PRAKTIKUM 1- INTEGRASI LARAVEL DENGAN ADMIN LTE 3**

1. Dalam root folder project lakukan command berikut, untuk mendefinisikan requirement project. composer require jeroennoten/laravel-adminlte



1. Melakukan instalasi requirement project di atas dengan command berikut: php artisan adminlte:install

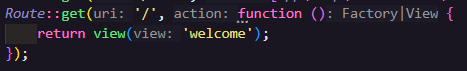


1. Buat file resources/views/layout/app.blade.php. Isi dengan kode berikut.

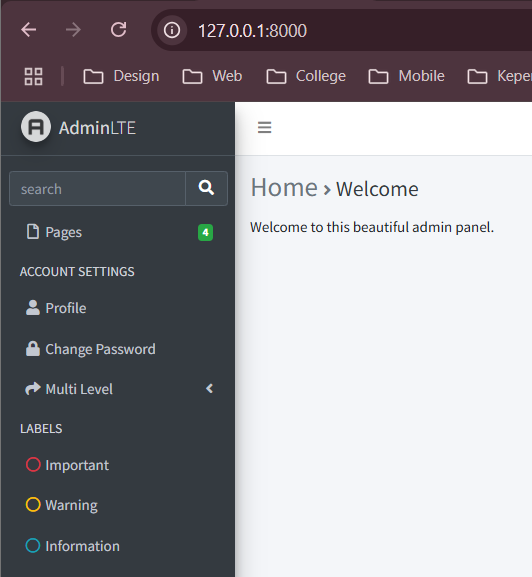


1. Edit resources/views/welcome.blade.php, kemudian replace seluruh kodenya dengan kode berikut





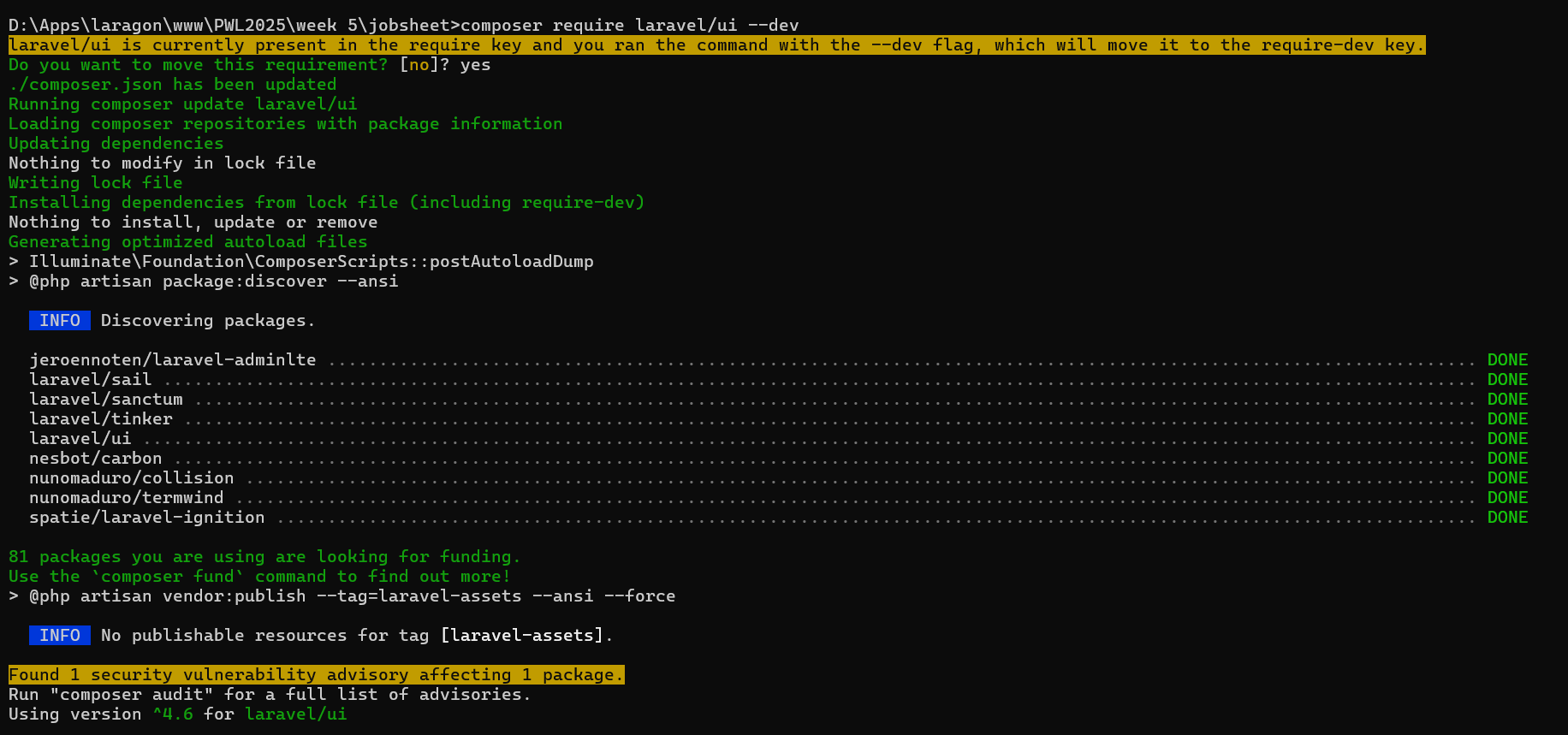
1. Kembali ke browser, menuju ke halaman awal

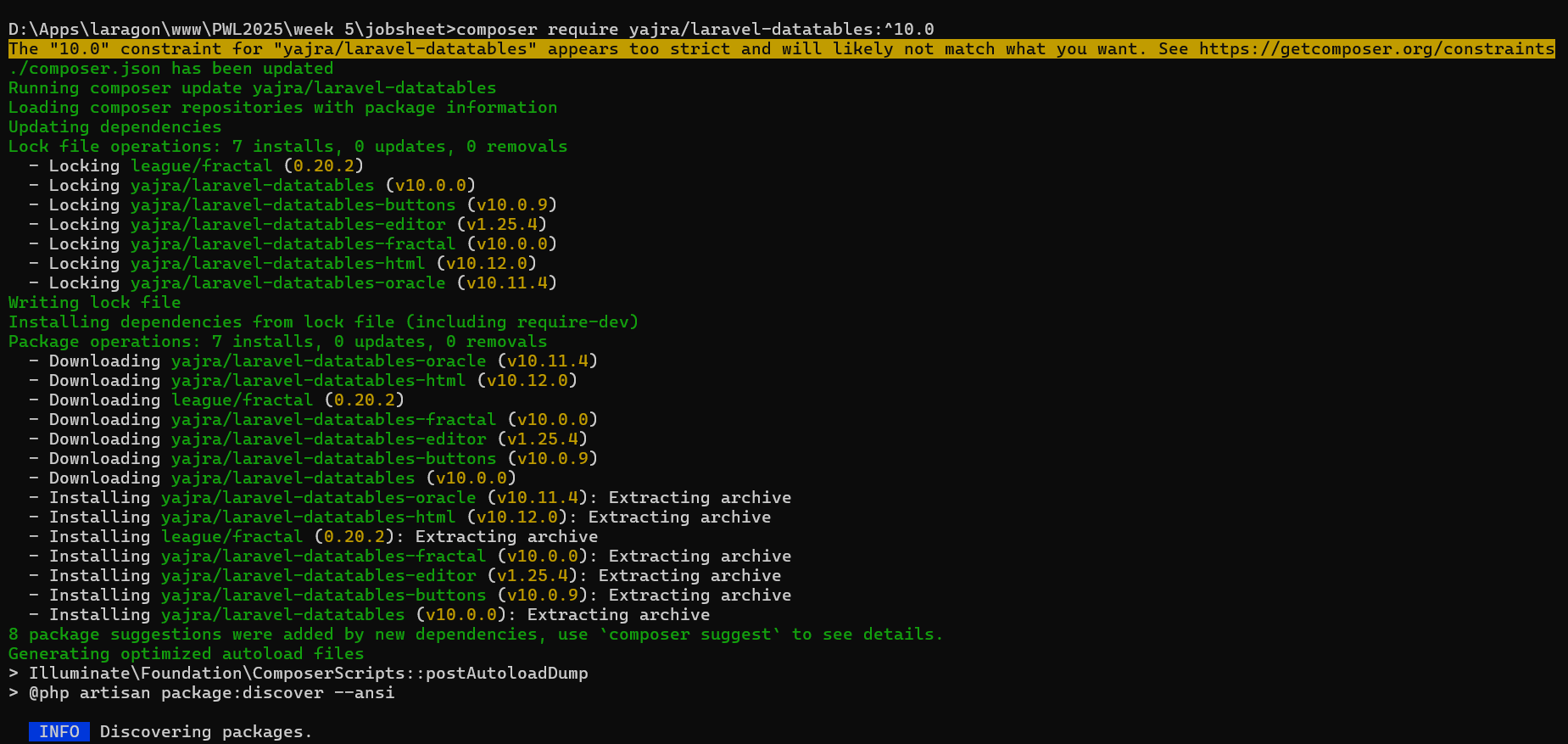


**PRAKTIKUM 2 – INTEGRASI DENGAN DATA TABLES**

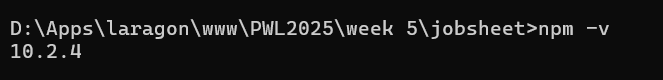
Datatables biasa digunakan untuk menampilkan list data dengan banyak fitur seperti searching, pagination,sorting dan lain-lainnya. Dalam laravel biasa digunankan yajra datatables karena didesain agar sesuai dengan Laravel

1. Install Laravel DataTables

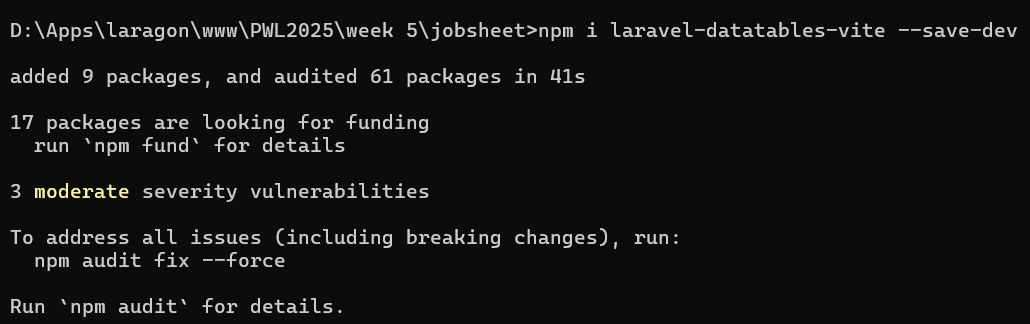


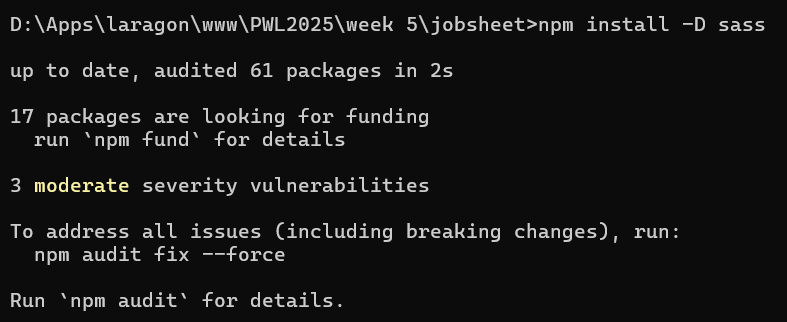


1. Pastikan nodejs sudah terinstall, dengan perintah npm -v. Npm digunakan untuk manajemen package javascript.

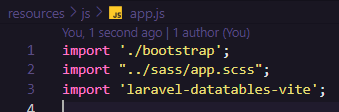


1. Install Laravel DataTables Vite dan sass





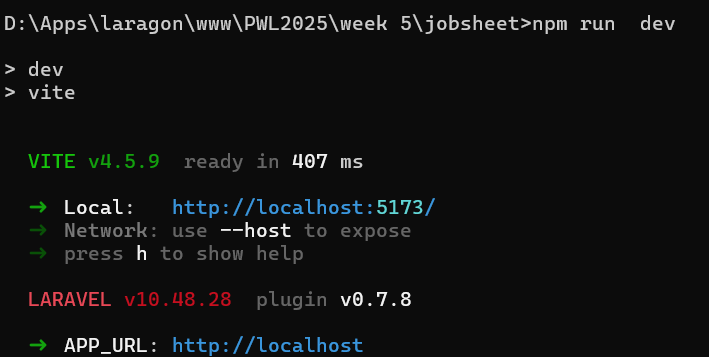
1. Edit file resources/js/app.js



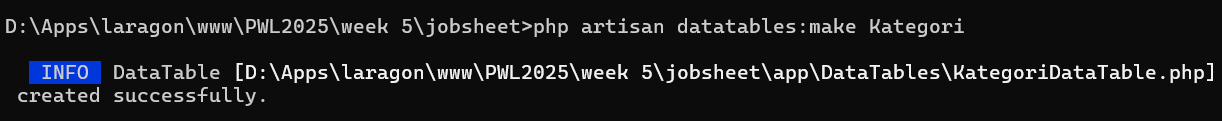
1. Buatlah file resources/saas/app.scss



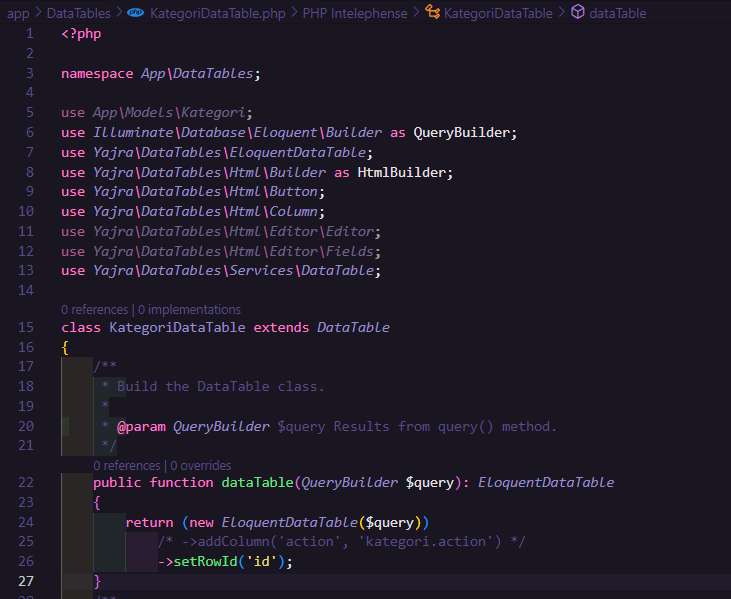
1. Jalankan dengan npm run dev



1. Selanjutnya kita akan buat datatables untuk kategori



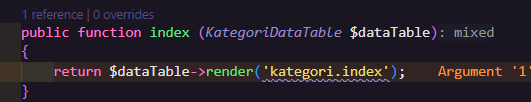
1. Kita edit KategoriDatable untuk mengatur kolom apasaja yang ingin ditampilkan Sesuaikan dengan kodeprogram berikut



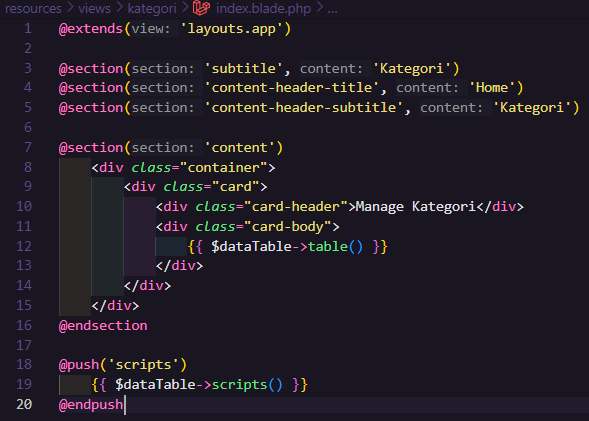
1. Ubah kategori model, sesuaikan seperti berikut



1. Ubah Kategori Controller sesuaikan dengan skrip berikut:



1. Buat folder kategori di dalam resources/view, kemudian buat view blade index untuk kategori resources/views/kategori/index.blade.php



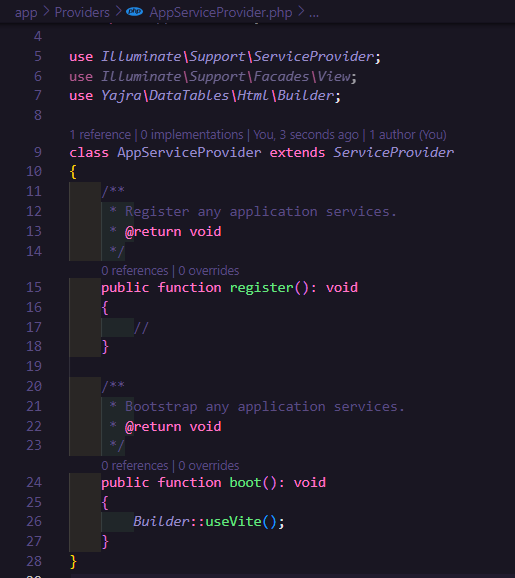
1. Pastikan route untung kategori sudah tersedia



1. Sesuaikan app layout seperti kode berikut



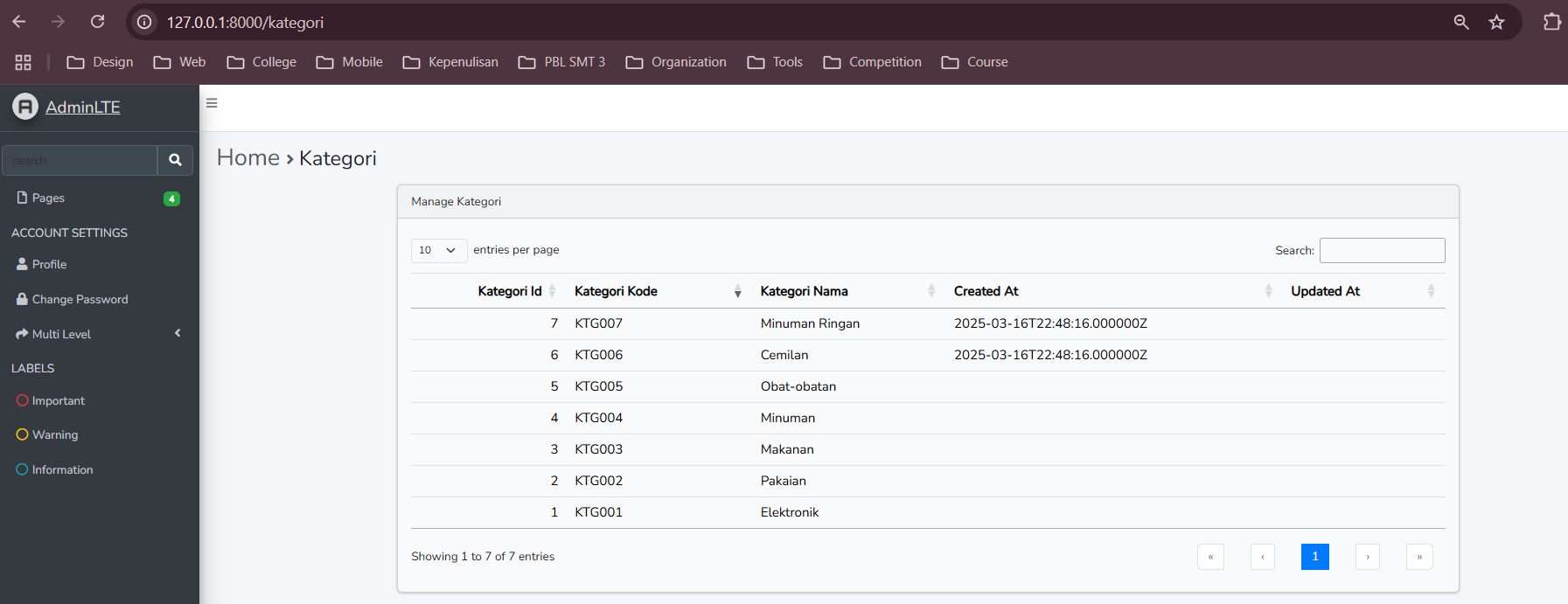
1. Menset ViteJs / script type defaults



1. . Isikan beberapa data ke table kategori



1. Datatables sudah dapat di load di /kategori

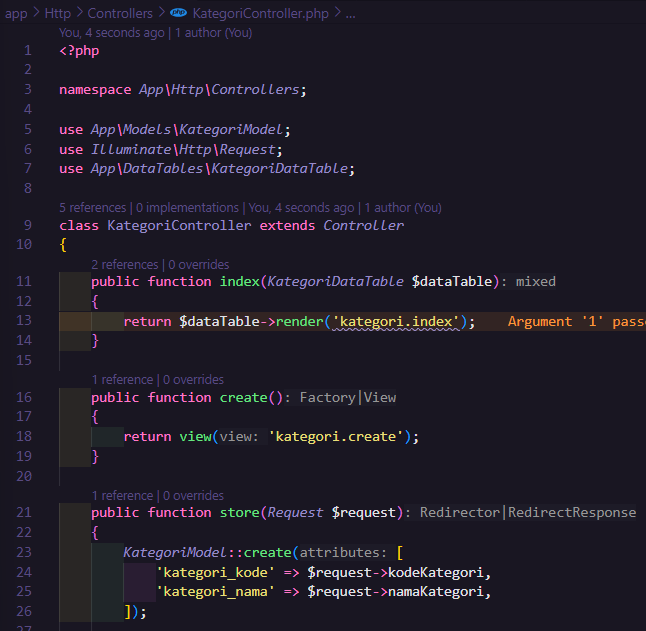


**PRAKTIKUM 3 – MEMBUAT FORM KEMUDIAN MENYIMPAN DATA DALAM DATABASE**

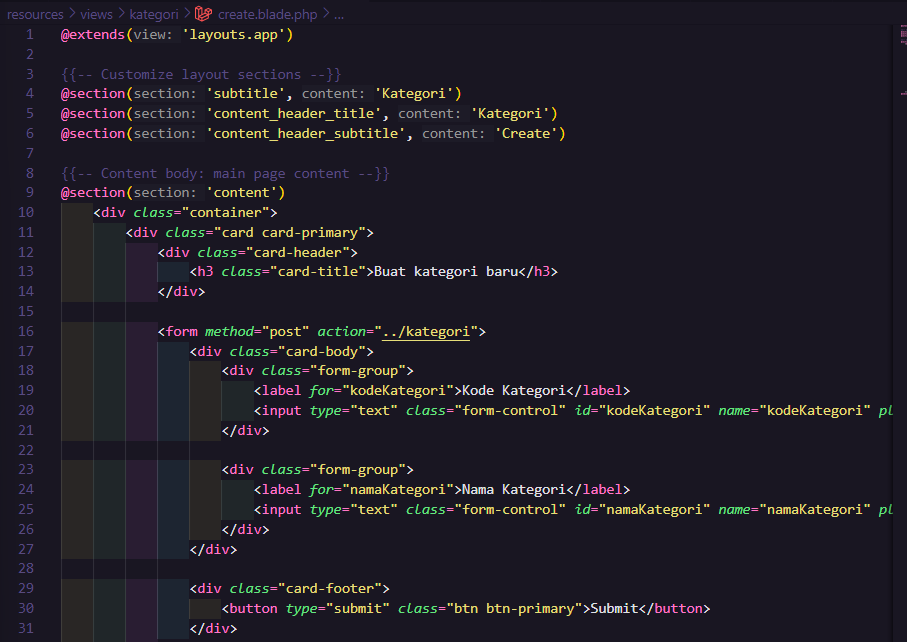
1. Menyesuaikan routing, tambahkan dua routing berikut



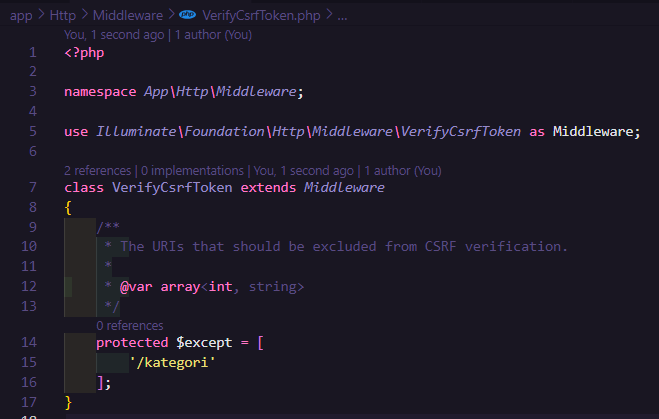
1. Tambahkan dua function berikut dalam KategoriController



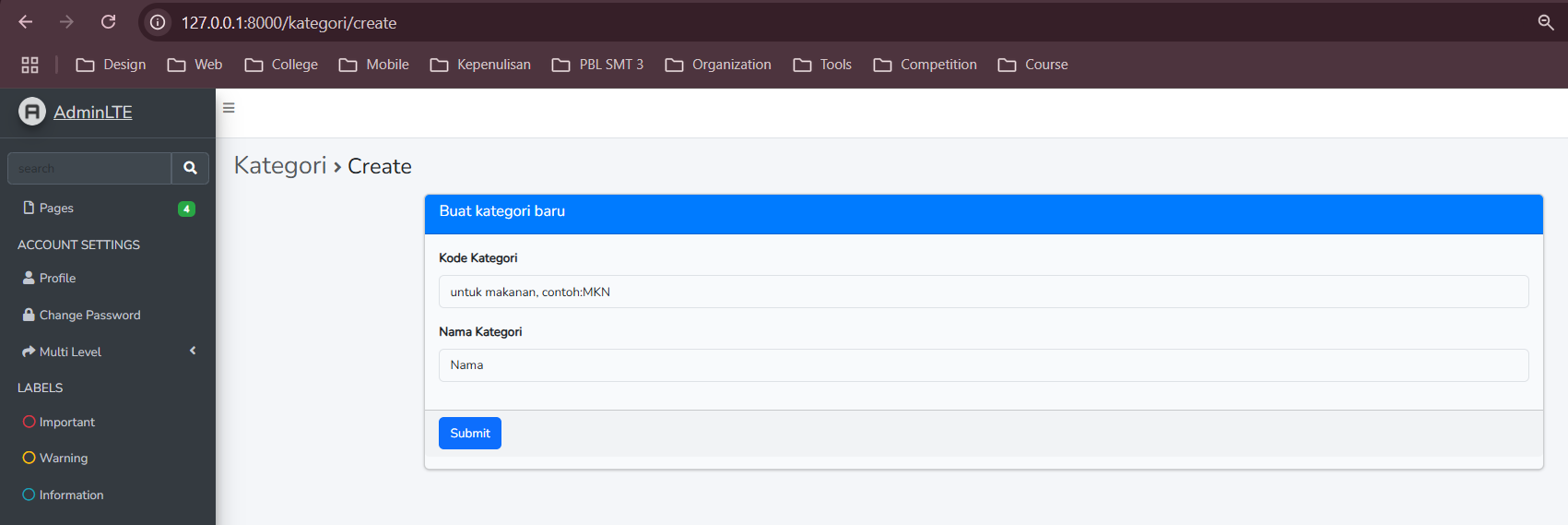
1. Dalam folder views/kategori, buatlah file dengan nama create.blade.php



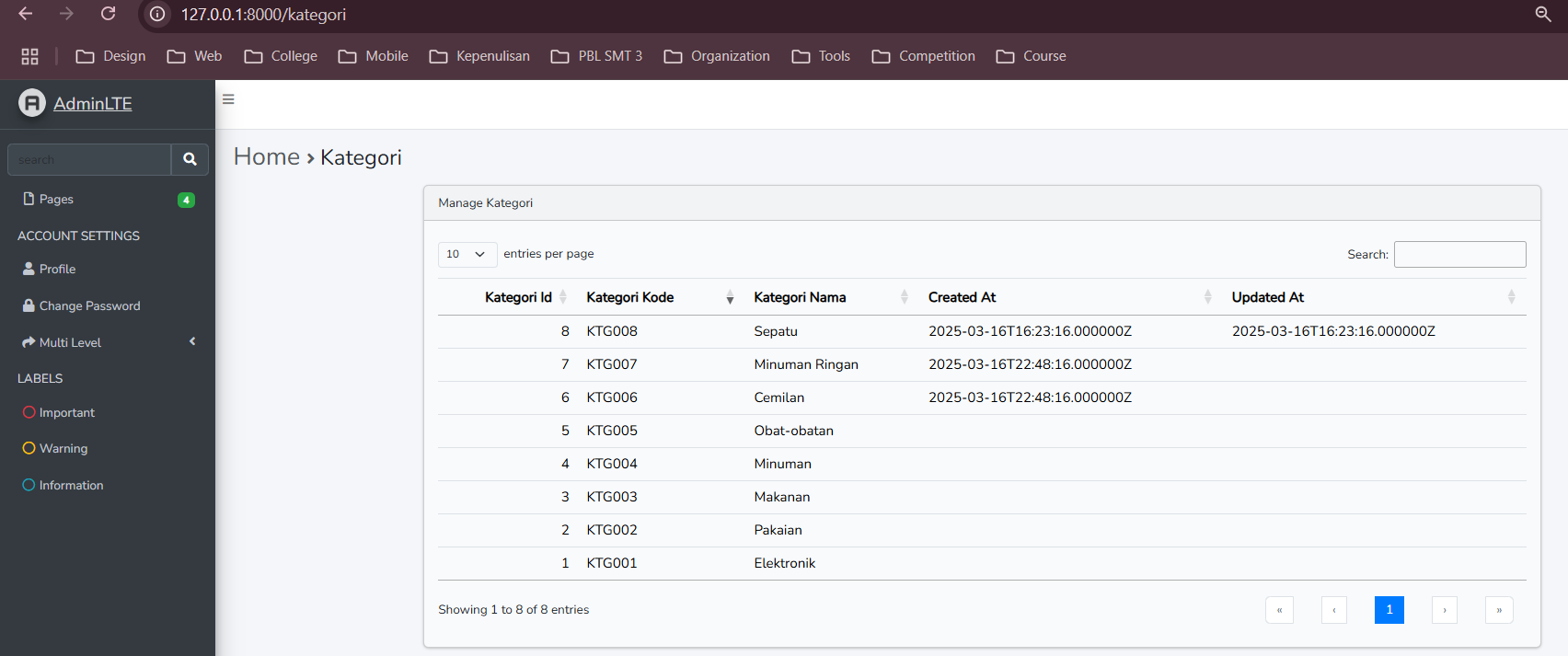
1. Kita lakukan pengecualian proteksi CsrfToken. Karena kita belum melakukan otentikasi. Kita edit dalam file berikut



1. Akses kategori/create



1. Halaman Kategori

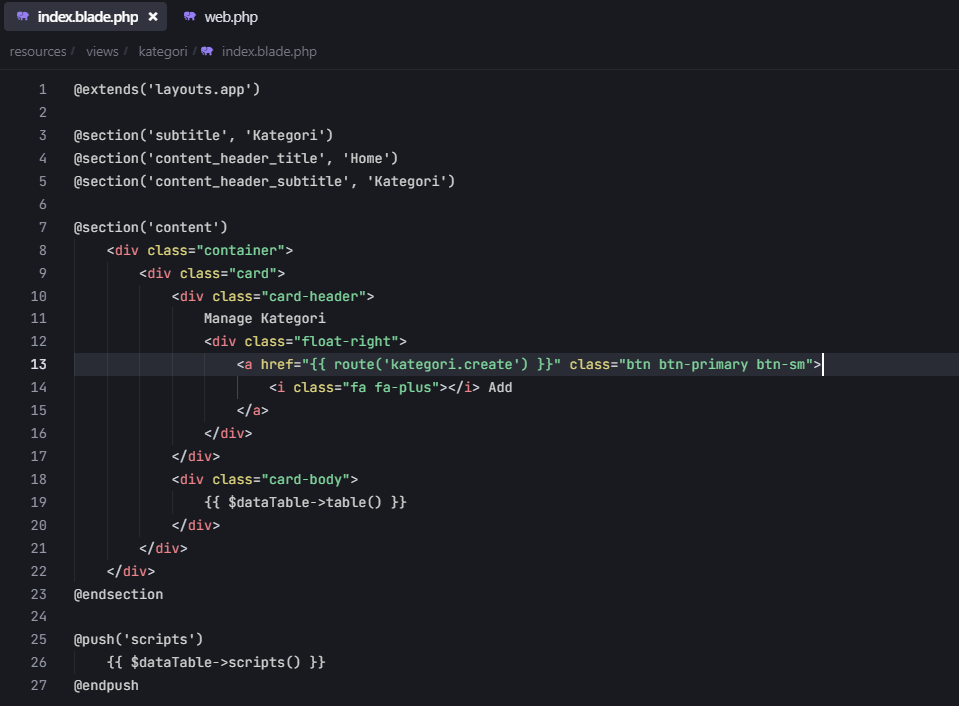


**TUGAS PRAKTIKUM**

1. Tambahkan button Add di halam manage kategori, yang mengarah ke create kategori baru

**Jawab:**

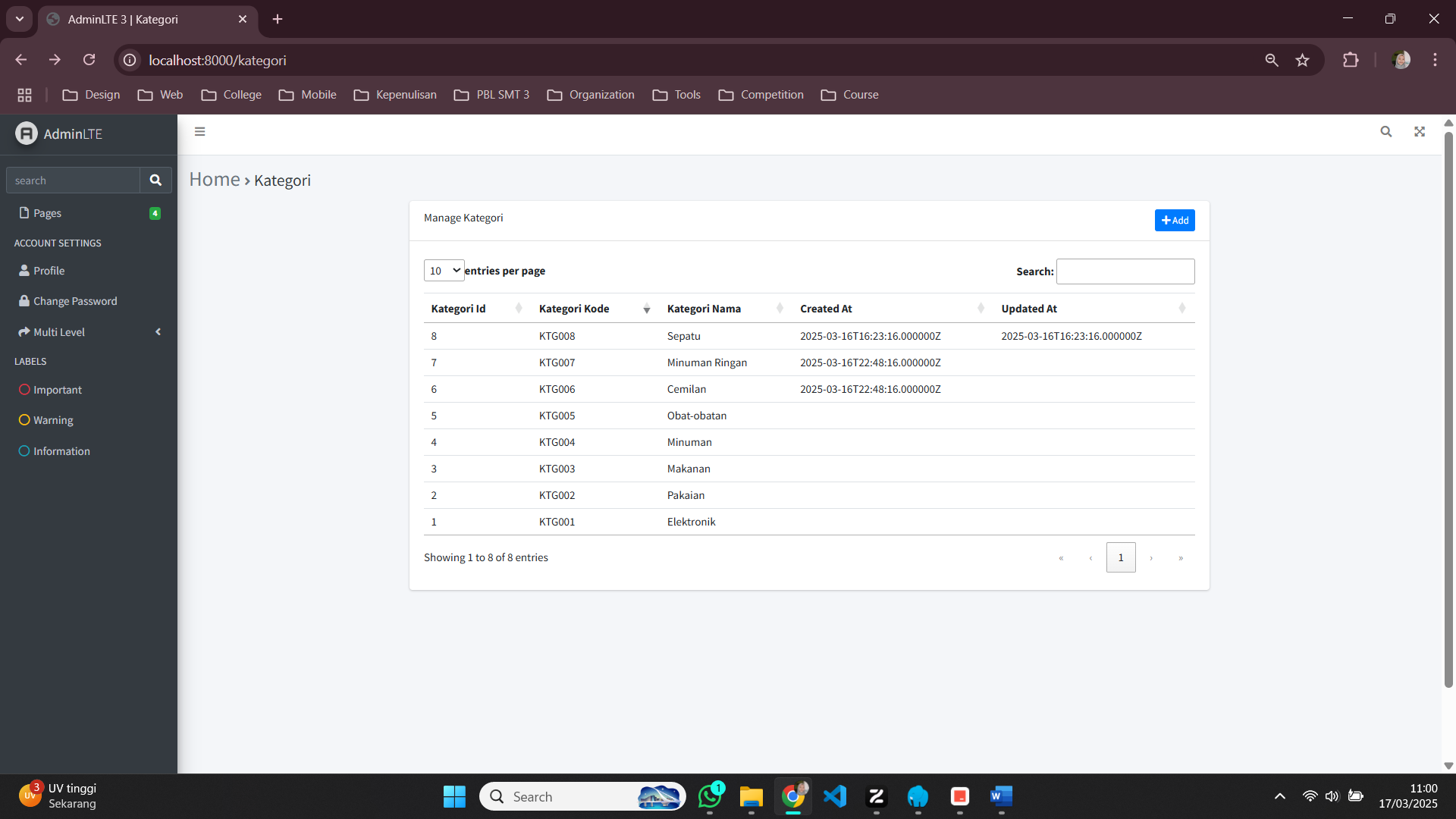
1. Modifikasi index.blade.php untuk menambahkan button add

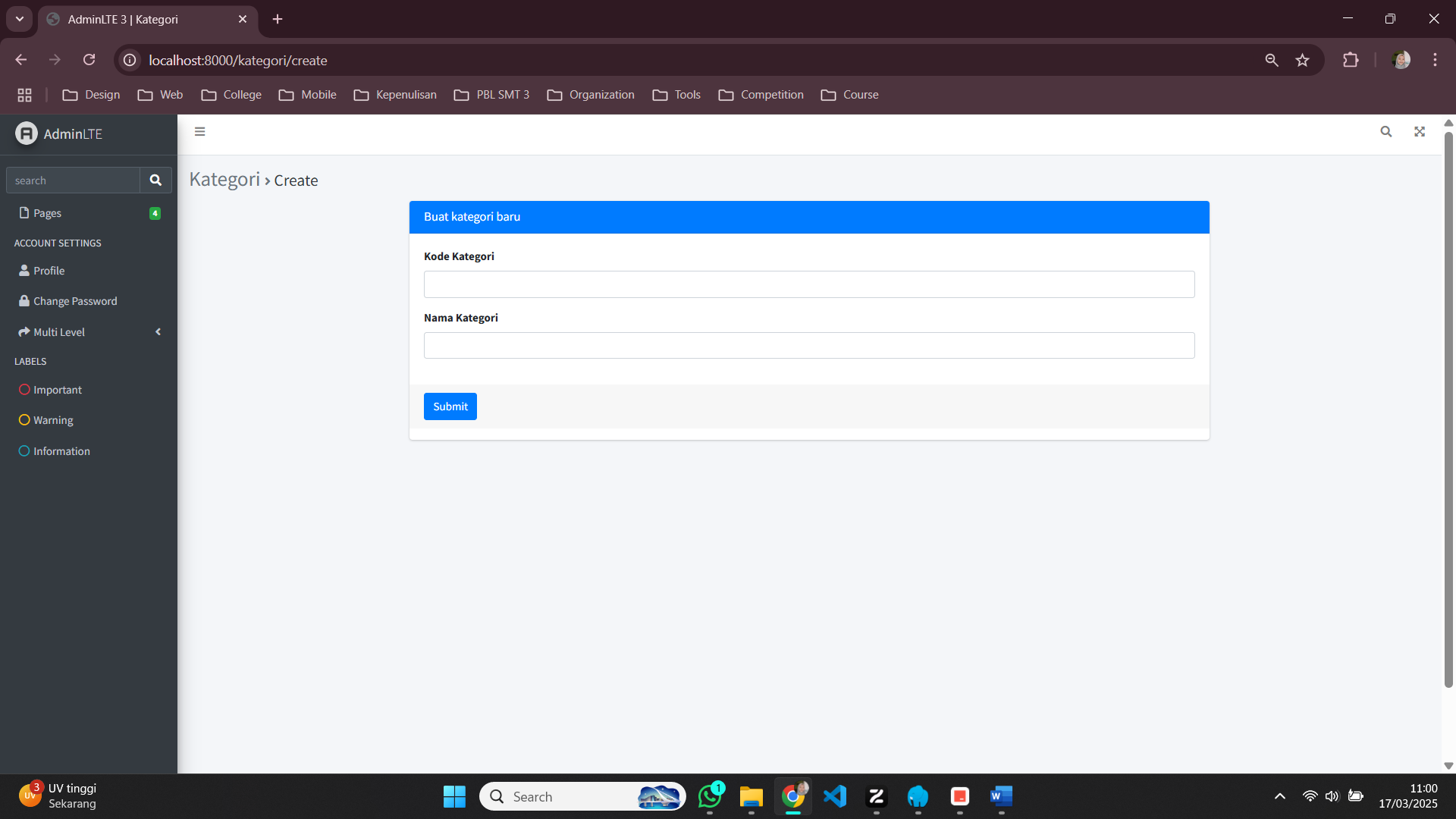


1. Modifikasi routingnya pada web.php



Ketika button add diklik pada /kategori, dia akan merouting ke /kategori/create





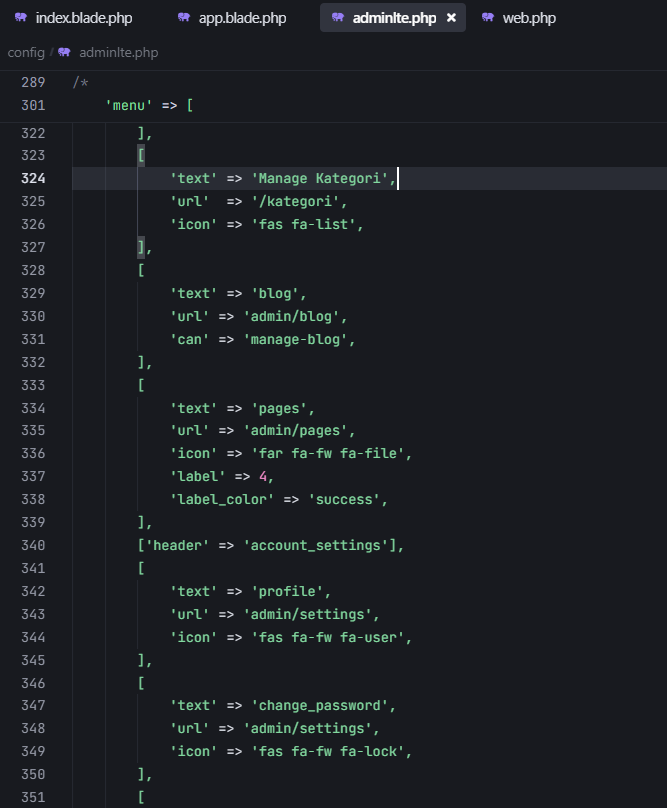
1. Tambahkan menu untuk halaman manage kategori, di daftar menu navbar

**Jawab:**

1. Untuk menambahkan menu item pada Manage Kategori, kita perlu meodifikasi layout file dimana navbar menu didefine, yaitu main layoutnya ada di app.blade.php, yg mengextends adminlte::page.



1. Modifikasi /config/adminlte.php karena konfigurasi item menunya ada disini

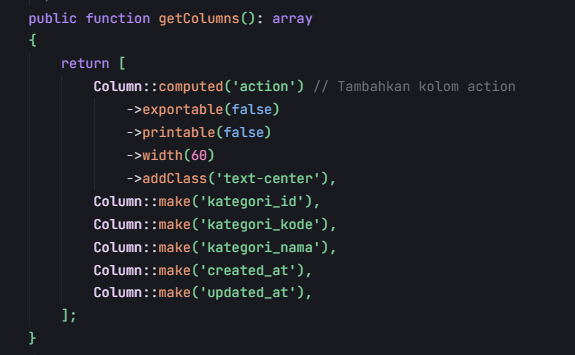


1. Tambahkan action edit di datatables dan buat halaman edit serta controllernya

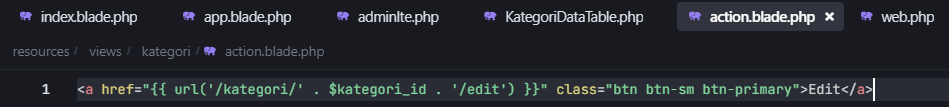
**Jawab:**

1. Kita tambahkan kolom “action” di datatables yg berisi edit untuk setiap baris data di table kategori





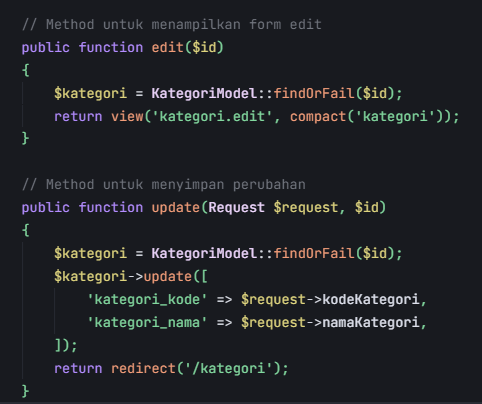
1. Membuat view untuk menampilkan tombil edit di kolom action



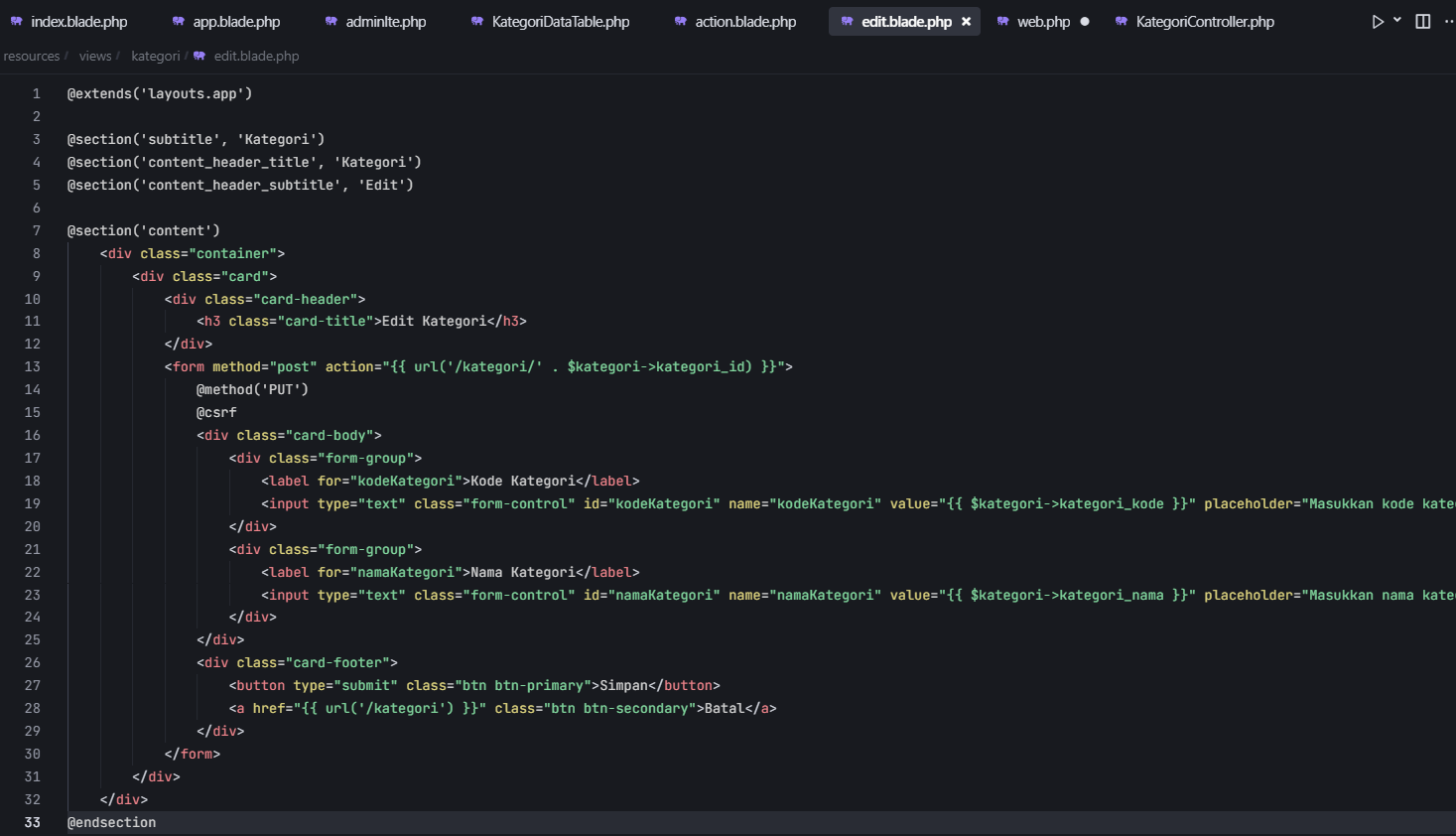
1. Menambahkan routing



1. Menambahkan method edit dan update pada KategoriController untuk menampilkan form edit, dan update untuk menyimpan perubahan.

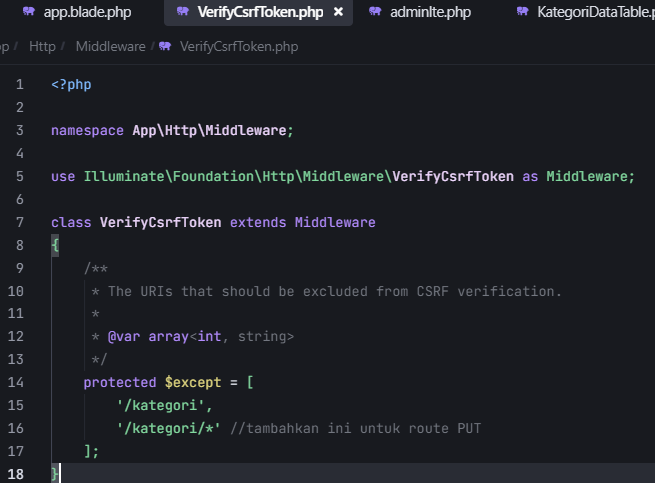


1. Buat view edit.blade.php yang berisi form untuk mengedit data kategori

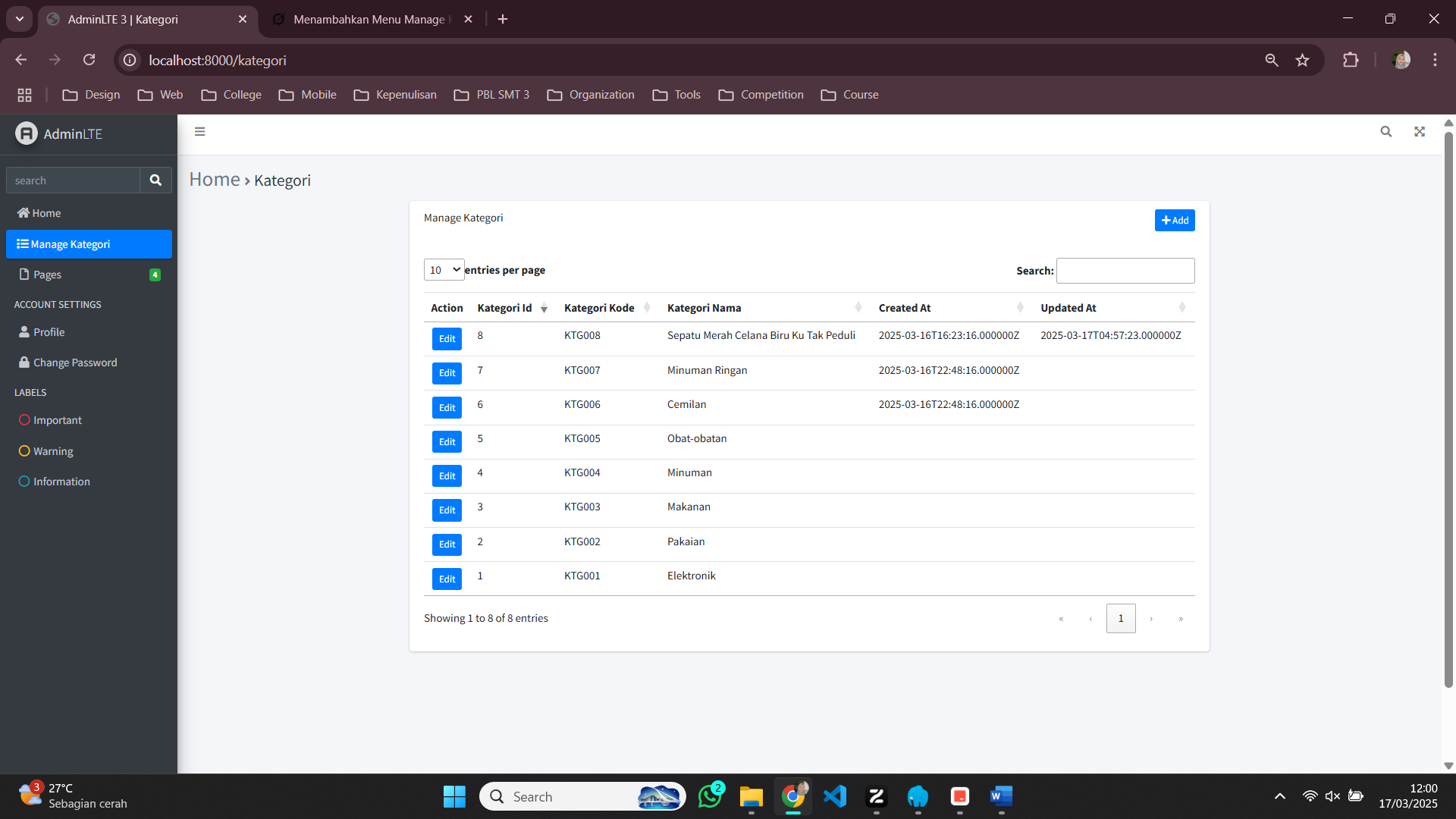


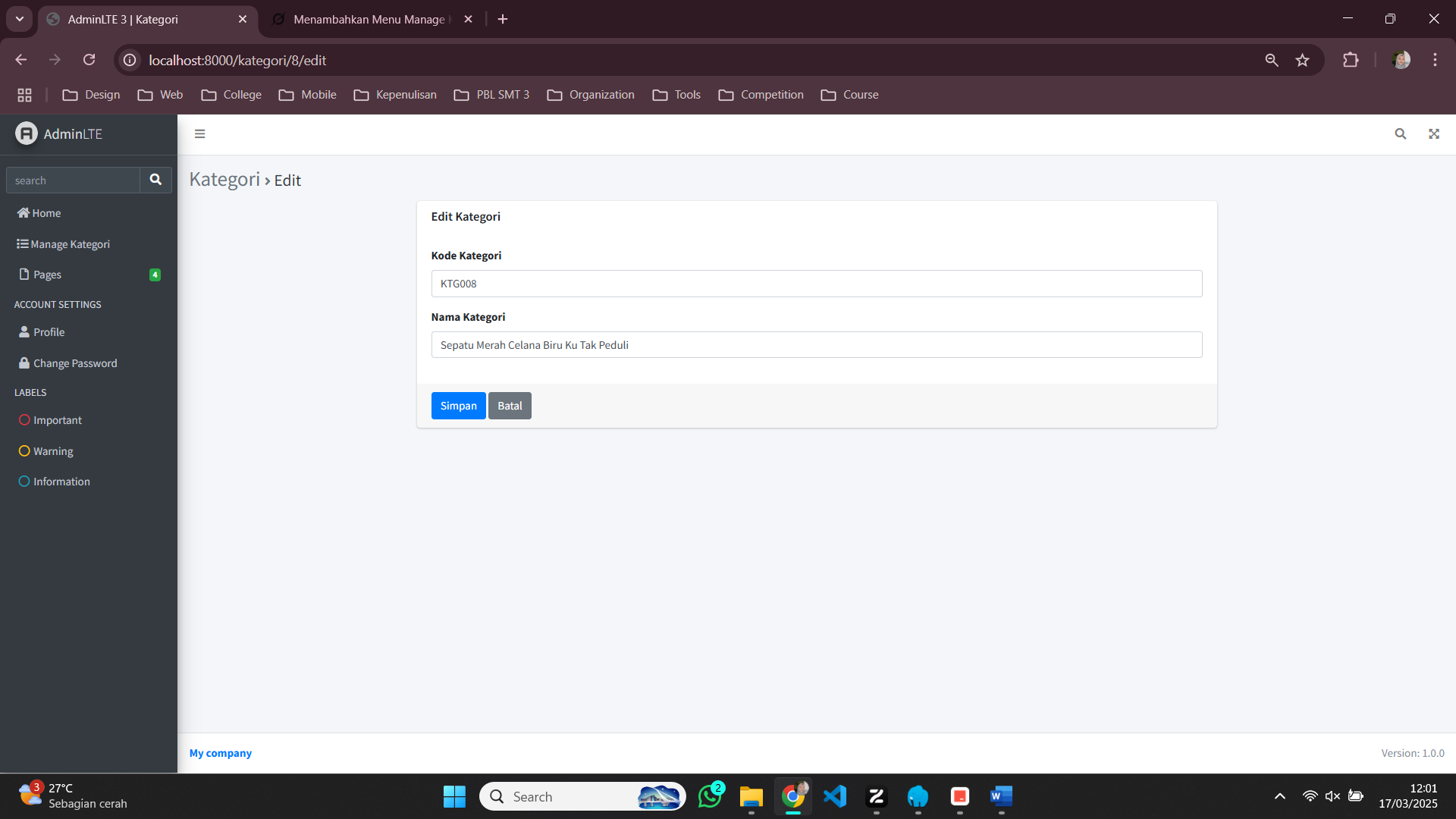
1. Karena kita menggunakan method PUT, kita perlu memastikan route ini dikecualikan dari verifikasi CSRF (sementara, karena belum ada autentikasi).

Kita akan menambahkan /kategori/\* ke array $except:



1. Hasilnya

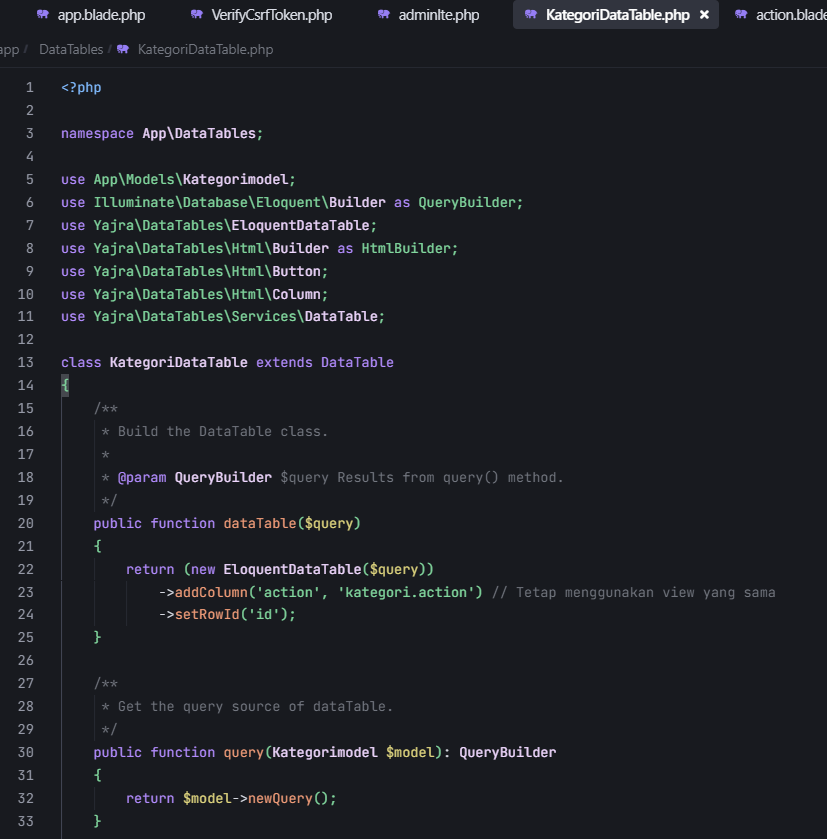




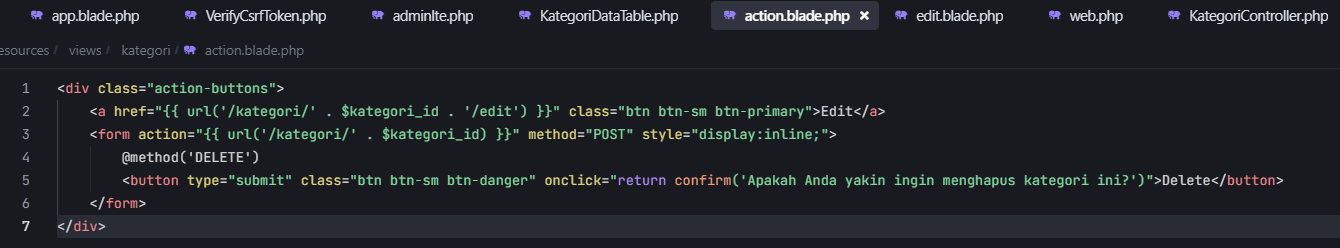
1. Tambahkan action delete di datatables serta controllernya

**Jawab:**

1. Kita akan memperbarui kolom "Action" di DataTables untuk menambahkan tombol "Delete" di samping tombol "Edit" yang sudah ada.



1. Tambahkan tombol "Delete" di samping tombol "Edit" dalam file view. Karena link <a> tidak mendukung method DELETE, kita akan menggantinya dengan form kecil yang menggunakan method DELETE



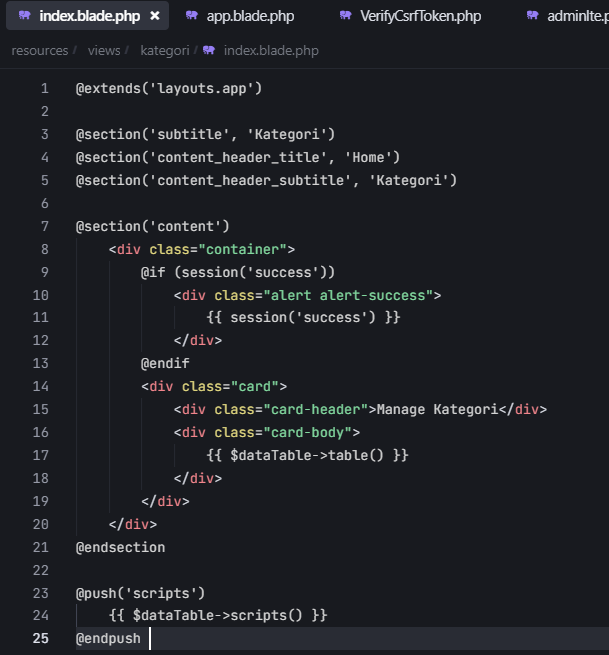
1. Tambahkan route untuk menghapus data kategori menggunakan method DELETE



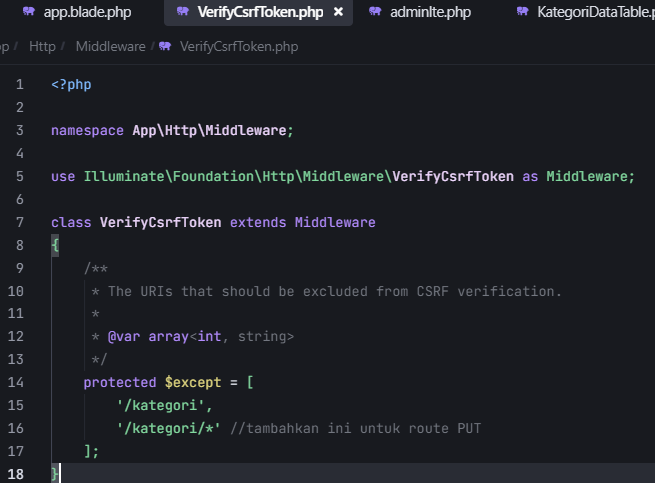
1. Tambahkan method destroy di KategoriController untuk menangani penghapusan data



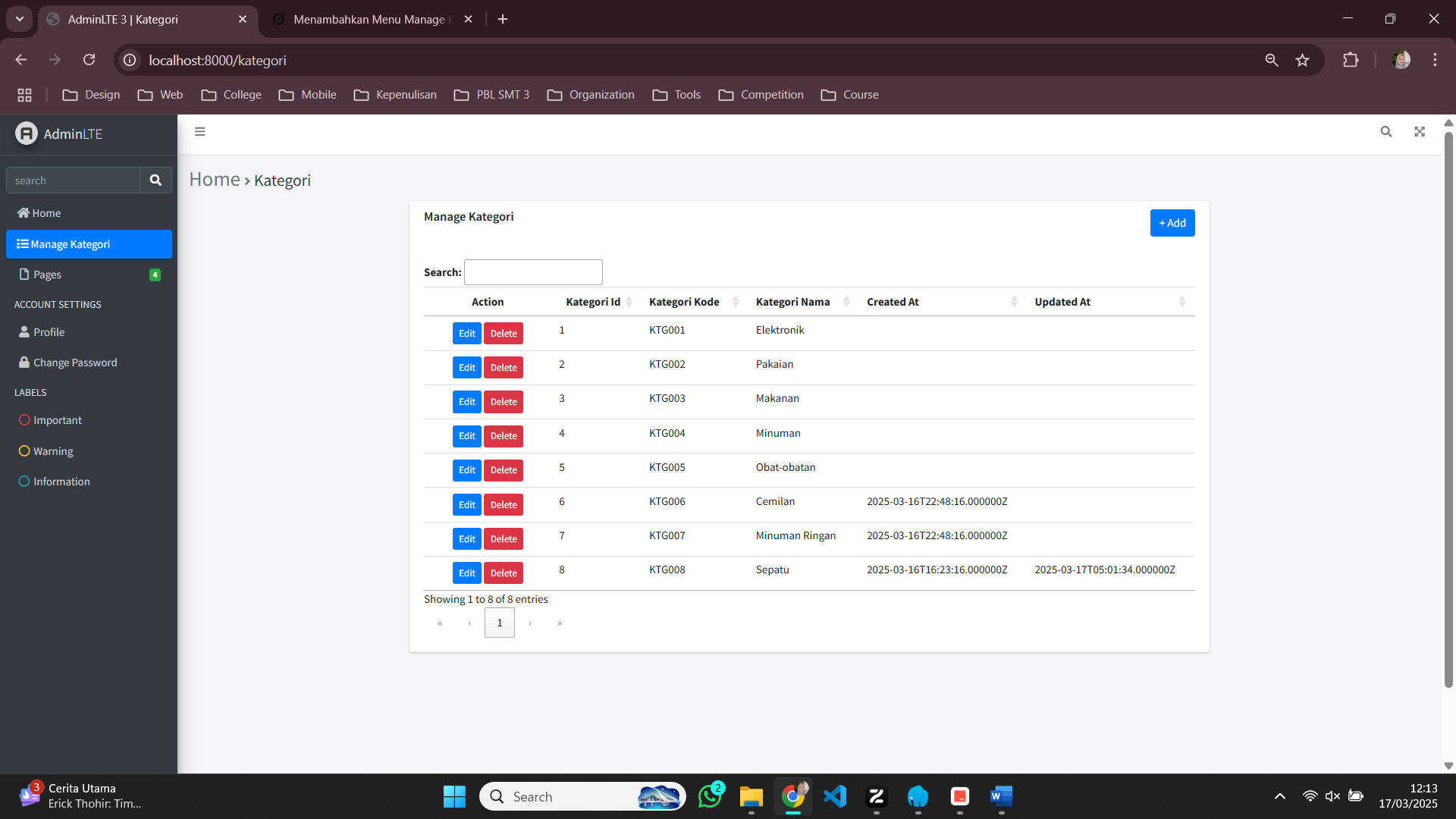
1. Untuk memberikan feedback kepada pengguna setelah penghapusan berhasil, tambahkan pesan sukses di halaman index.



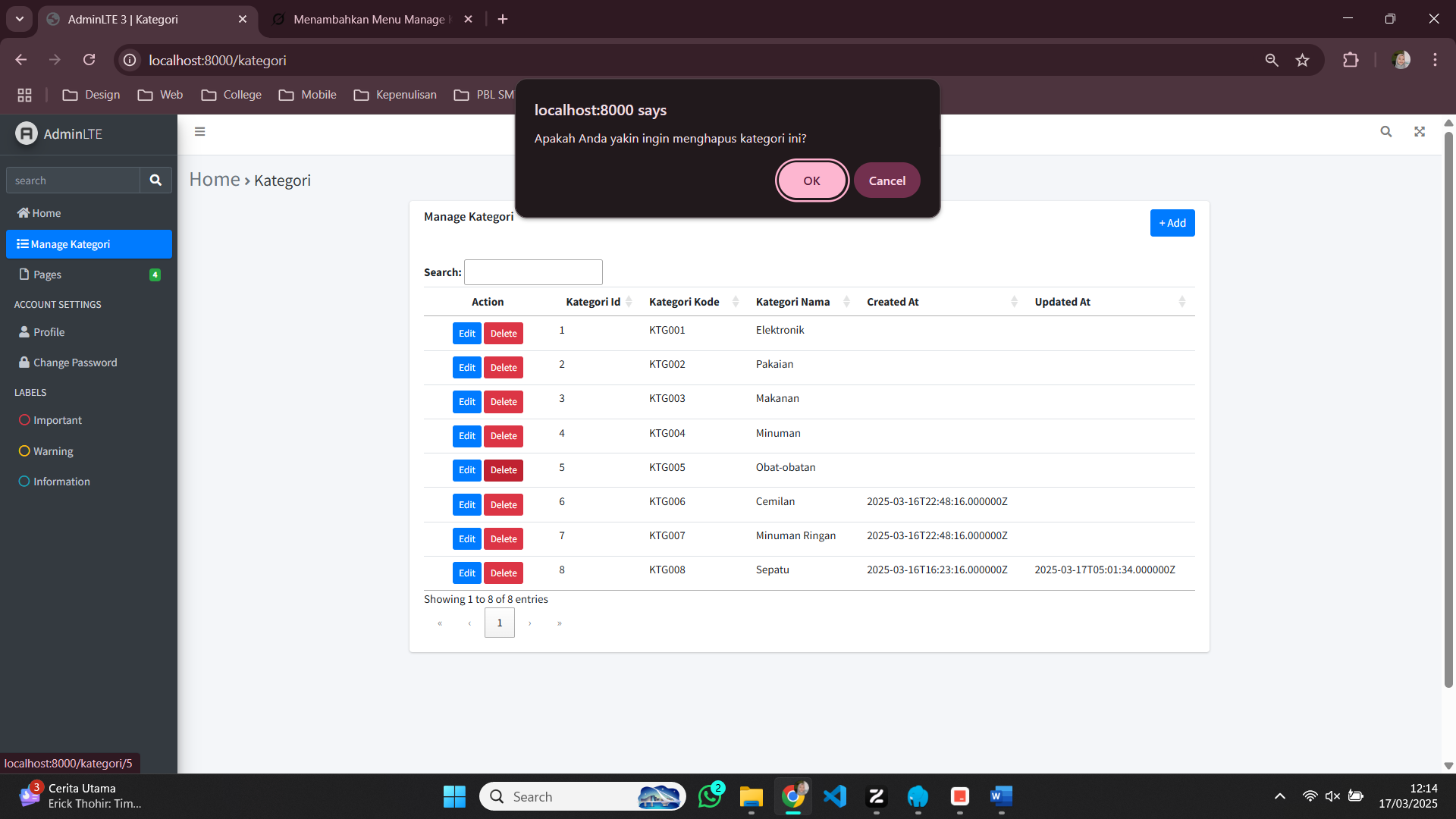
1. Karena form HTML tidak mendukung method DELETE secara langsung, kita akan menggunakan @method('DELETE') di dalam link penghapusan. Namun, karena tombol "Delete" adalah link (<a>), kita perlu menangani ini melalui JavaScript atau memastikan route dikecualikan dari verifikasi CSRF sementara



1. Hasilnya



Saya ingin menghapus kategori id 8



Maka tampilannya akan seperti ini

