|  |  |  |
| --- | --- | --- |
| Mata Kuliah | **:** | Pemrograman Web Lanjut (PWL) |
| Program Studi | **:** | D4 – Teknik Informatika **/** D4 – Sistem Informasi Bisnis |
| Semester | **:** | 4 (empat) / 5 (lima) |
| Pertemuan ke- | **:** | 5 (lima) |

# JOBSHEET 05

**Blade View, Web Templating(AdminLTE), Datatables**

Sesuai dengan **Studi Kasus PWL.pdf**.

Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL\_POS.**

*Project* **PWL\_POS** akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajarai

**A. Blade View**

Laravel Blade adalah template engine yang digunakan secara default oleh Laravel untuk mempermudah pembuatan tampilan (view) dalam aplikasi web. Blade menawarkan fitur-fitur yang powerful dan mudah digunakan untuk memisahkan logika bisnis dari tampilan antarmuka pengguna (UI). Berikut adalah penjelasan lengkap tentang Laravel Blade View:

## Apa Itu Blade?

Blade adalah template engine yang disertakan dengan Laravel. Dengan Blade, Anda bisa menulis sintaks HTML yang terintegrasi dengan PHP dengan cara yang lebih bersih dan terstruktur. Meskipun menggunakan Blade, semua file template Blade di Laravel tetap memiliki ekstensi *.blade.php.*

## Keuntungan Menggunakan Blade

* + **Inheritance (Pewarisan Template)**: Blade memungkinkan Anda untuk membuat layout dasar yang bisa di-extend oleh halaman lain. Ini mempermudah pembuatan struktur halaman yang konsisten.
  + **Komponen**: Anda dapat membuat komponen yang reusable untuk elemen UI yang sering digunakan.
  + **Control Structures**: Blade mendukung struktur kontrol seperti if, for, while, dan foreach yang mirip dengan sintaks PHP, namun dengan tampilan yang lebih bersih.
  + **Tidak Ada Overhead**: Semua kode Blade dikompilasi menjadi PHP biasa, jadi tidak ada overhead tambahan.

## Sintaks Dasar Blade

Blade memiliki sintaks yang sangat sederhana dan intuitif. Berikut adalah beberapa contoh penggunaan sintaks Blade:

## Menampilkan Variabel:



Blade akan secara otomatis melindungi (escape) output untuk mencegah serangan XSS (Cross-Site Scripting).

## Struktur Kontrol:



* + **Komentar**

****

**B. Web Template**

Merupakan suatu file yang sudah memiliki desain tertentu sehingga, kita tinggal memodifikasi bagian-bagian tertentu saja untuk mendapatkan hasil yang sudah bagus. Dengan menggunakan template yang sesuai dengan kebutuhan, kita bisa mengurangi waktu pembuatan suatu website secara drastis.

Secara umum kita bisa menggunakan template website yang sudah ada di internet untuk mempermudah kita dalam membuat website. Ada banyak sekali template yang ada di internet dan kita ambil satu contoh template untuk halaman administrator, seperti **AdminLTE** template. **AdminLTE** adalah salah satu template yang sering digunakan oleh web developer sebagai

template backend/admin pada proyek yang sering dikerjakan. Template ini dibuat menggunakan framework bootsrap yang merupakan framework CSS yang paling banyak digunakan di kalangan web desainer

Template AdminLTE dapat di download di [**https://adminlte.io**](https://adminlte.io/)

Contoh Web template

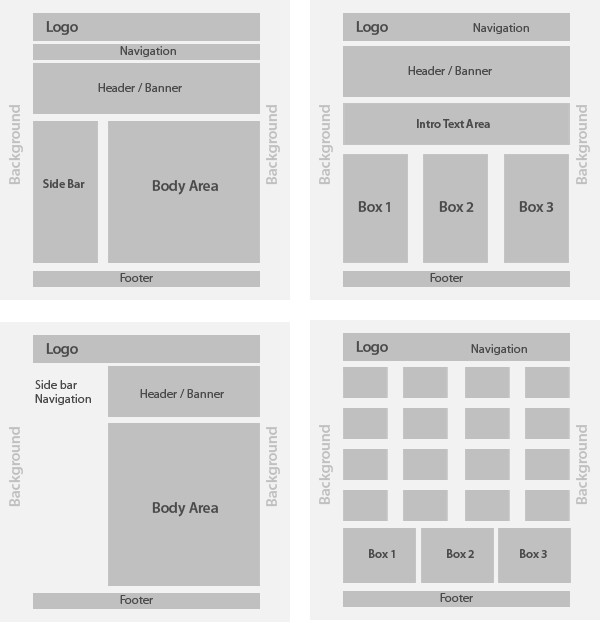
|  |  |
| --- | --- |
| **Frontend Template** | **Backend Template** |
| Desain untuk penasihat keuangan | Limitless - Responsive Web Application Kit |

|  |  |
| --- | --- |
| Template pemodelan ulang arsitektur | Fily - Responsive Bootstrap Admin Template |

**A. Web Layout**

Merupakan sekumpulan elemen atau variabel desain yang terkait dengan media tertentu untuk mendukung konsep pembuatan website. Pada dasarnya tujuan penggunaan layout website adalah untuk menggabungkan elemen yang digunakan untuk membangun website. Selain elemen di dalamnya, layout website juga harus memuat beberapa elemen, mulai dari header, navigasi, sidebar, hingga footer. Untuk memahami perbedaan elemen sebuah layout website, berikut penjelasan singkatnya

* + **Header**  Pada bagian header, layout designer dapat mengisi bagian ini dengan logo website, navigasi situs, ikon media sosial, dan menu pencarian.
  + **Navigasi**  Navigasi dapat dipahami sebagai panduan. Di website, navigasi dapat berkisar dari menu yang muncul di bagian atas halaman web hingga menu pendukung lainnya yang biasa ditemukan di bawah situs.
  + **Body/Konten**  Selain itu, terdapat elemen body/konten yang biasanya diisi dengan informasi produk, fitur produk, dan deskripsi produk yang dijual.
  + **Sidebar**  elemen ini merupakan elemen yang berada di samping kanan/kiri sebuah website. Element ini kadang sudah tidak dipakai, terutama pada Web Frontend. Namun, sidebar masih digunakan di banyak halaman artikel, yang kemudian akan diisi dengan informasi produk, produk terpopuler, dan navigasi tambahan.
  + **Footer**  Elemen terakhir adalah footer. Footer merupakan elemen yang berada dibawah sendiri sebagai menu pendukung atau sebagai identitas dari sebuah website.



**C. Layouting AdminLTE**

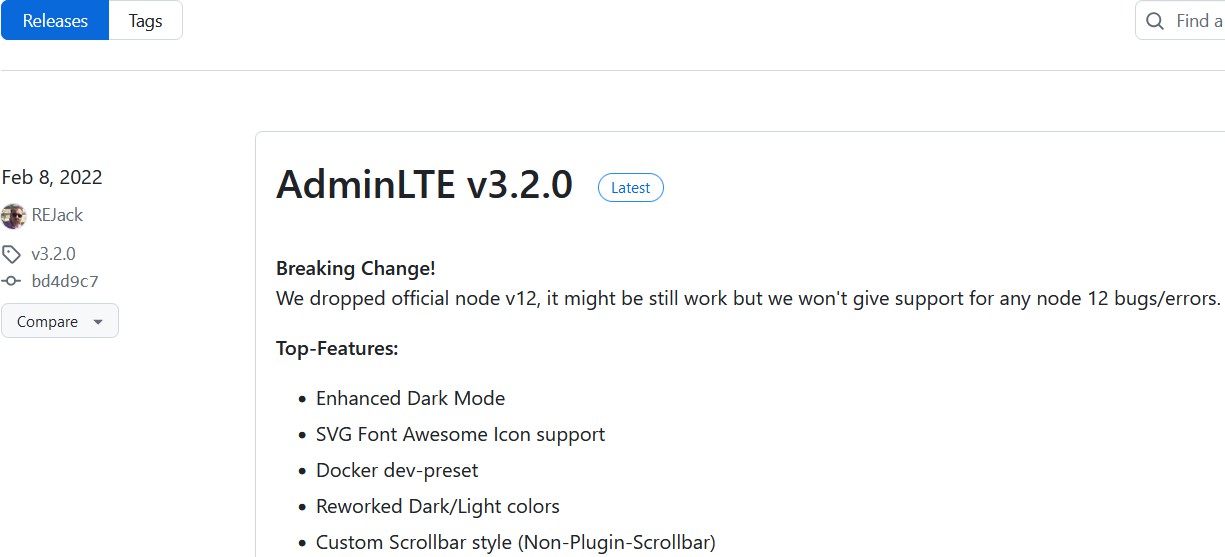
Pada kali ini kita akan melakukan *layouting* template AdminLTE. Tujuan dari *layouting* ini adalah untuk memecah bagian-bagian dari template adminLTE menjadi berbagai elemen. Hal ini dapat mempermudah kita dalam membuat website karena kita bisa menggunakan berulang-ulang elemen yang sudah kita buat nantinya.

Pada layouting kali ini, kita membutuhkan **Blade Template Engine** dari Laravel untuk memecah web menjadi beberapa elemen. Komponen blade yang kita butuhkan untuk proses layouting pada pertemuan kali ini adalah

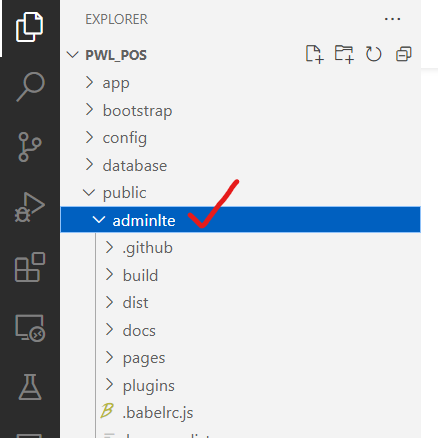
* + @include()
  + @extend()
  + @section()
  + @push()
  + @yield()
  + @stack()

**Praktikum 1 - Layouting AdminLTE:**

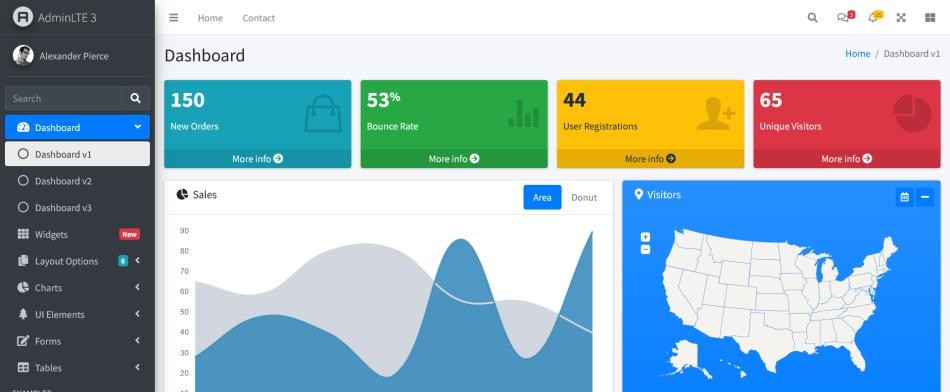
1. Kita download **AdminLTE v3.2.0** yang rilis pada 8 Feb 2022



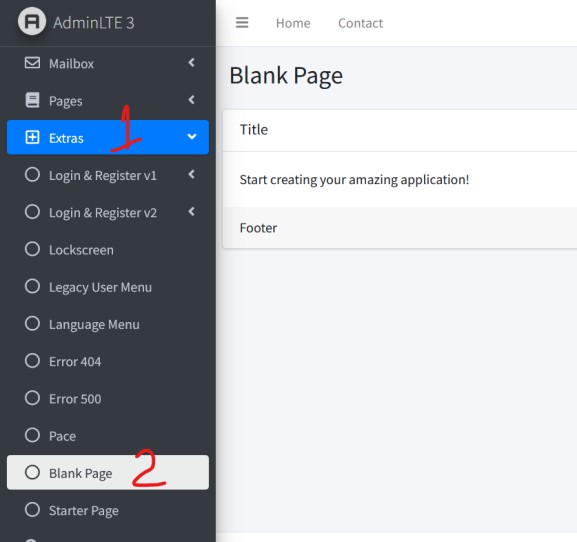
1. Seleteh kita berhasil download, kita ekstrak file yang sudah di download ke folder project PWL\_POS/public, kemudian kita ***rename*** folder cukup menjadi adminlte



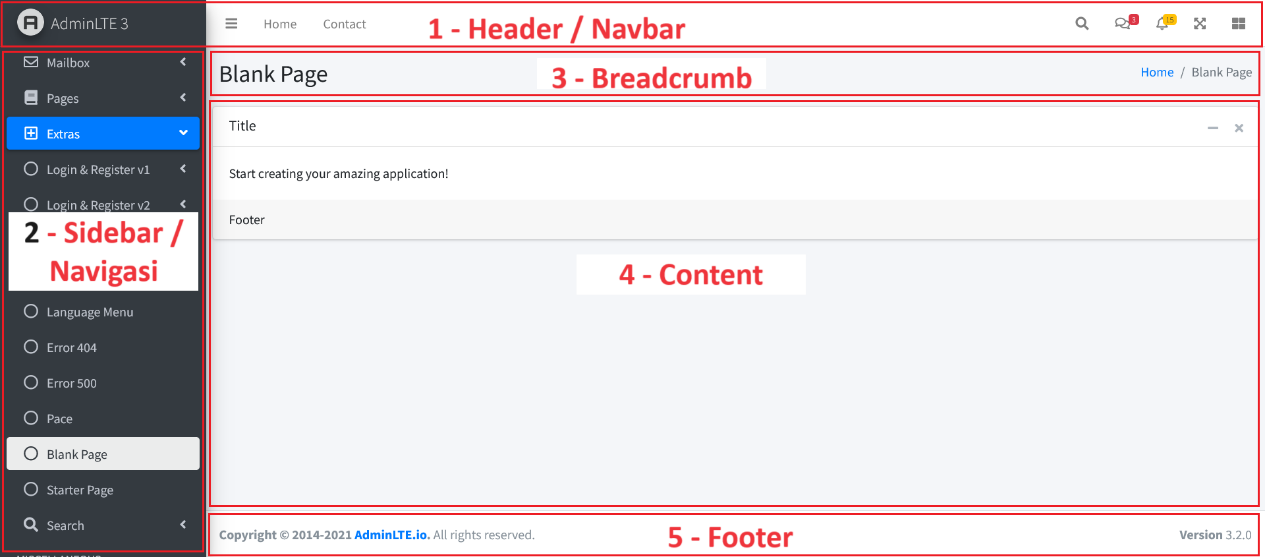
1. Selanjutnya kita buka di browser dengan alamat <http://localhost/PWL_POS/public/adminlte> maka akan muncul tampilan seperti berikut



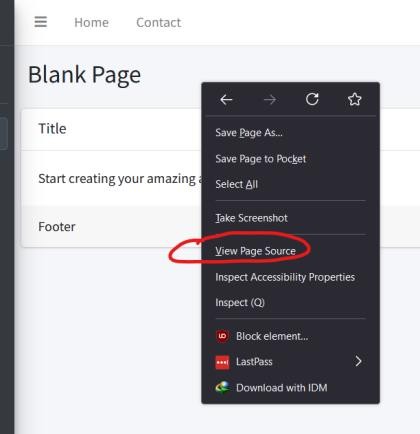
1. Kita klik menu Extras > Blank Page, page inilah yang akan menjadi dasar web template



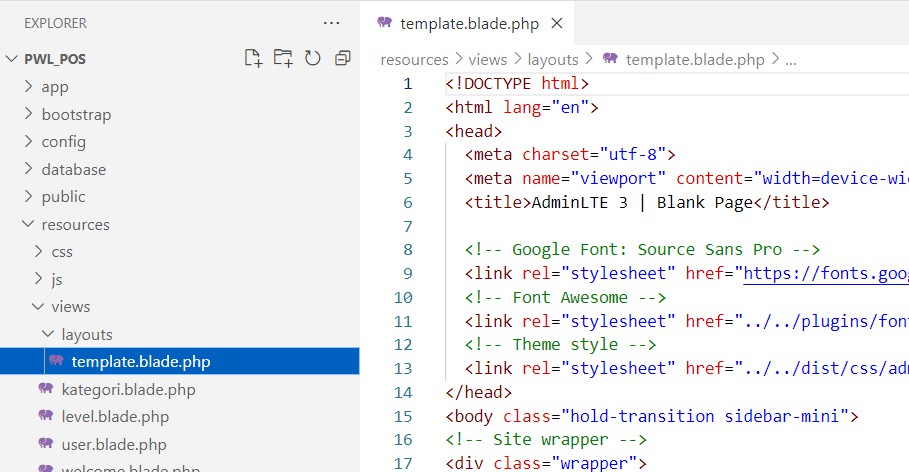
1. Dari sini kita bisa melakukan layouting halaman Blank Page ini menjadi 4 element seperti pada gambar berikut



1. Selanjutnya kita klik kanan halaman Blank Page dan klik *view page source*



1. Selanjutnya kita copy page source dari halaman Blank Page, kemudia kita *paste* pada **PWL\_POS/resource/view/layouts/template.blade.php** (buat dulu folder layouts dan file template.blade.php)

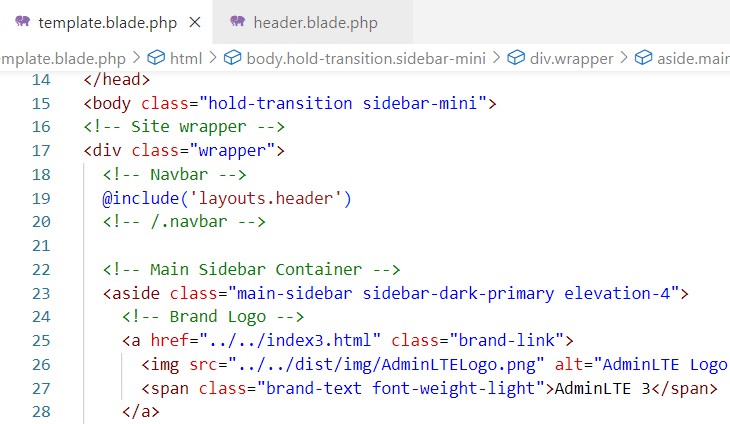


1. File layouts/template.blade.php adalah file utama untuk templating website
2. Pada baris **1-14** file template.blade.php, kita modifikasi

Menjadi



1. Kemudian kita blok baris **19-153** (baris untuk *element* ***1-header***), lalu kita ***cut***, dan ***paste***-kan di file **PWL\_POS/resource/view/layouts/header.blade.php** (buat dulu file header.blade.php jika belum ada). Sehingga tampilan dari file template.blade.php menjadi seperti berikut



Baris **19** adalah komponen Blade untuk memanggil elemen layouts/header.blade.php

agar menjadi satu dengan template.blade.php saat di-render nanti.

1. Kita modifikasi baris **25** dan **26** pada template.blade.php

Menjadi



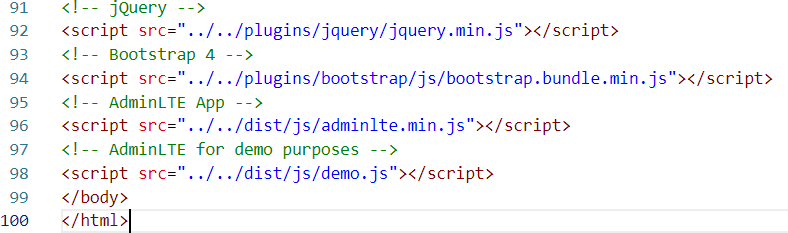
1. Selanjutnya kita blok baris **31-693** (baris untuk *element* ***2-sidebar***), lalu kita ***cut***, dan ***paste***-kan di file **PWL\_POS/resource/view/layouts/sidebar.blade.php** (buat dulu file sidebar.blade.php jika belum ada). Sehingga tampilan dari file template.blade.php menjadi seperti berikut



1. Selanjutnya perhatikan baris **87-98** (baris untuk *element* ***5-footer***), lalu kita ***cut***, dan ***paste***-kan di file **PWL\_POS/resource/view/layouts/footer.blade.php** (buat file footer.blade.php jika belum ada). Sehingga tampilan dari file template.blade.php menjadi seperti berikut



1. Kemudian kita modifikasi file template.blade.php baris **91-100**



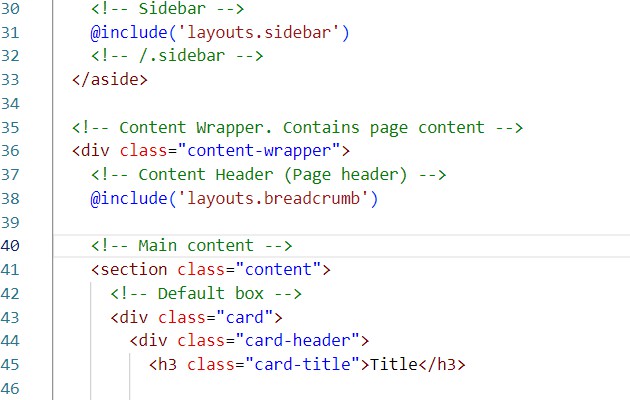
Menjadi



1. Sekarang masuk pada bagian konten. Konten kita bagi menjadi 2, yaitu elemen untuk

***breadcrumb*** dan elemen untuk ***content*.**

1. Perhatikan file template.blade.php pada baris **38-52** kita jadikan sebagai elemen ***4- breadcrumb***. Kita blok baris **38-52** lalu kita ***cut***, dan ***paste***-kan di file **PWL\_POS/resource/view/layouts/breadcrumb.blade.php** (buat file breadcrumb.blade.php jika belum ada). Sehingga tampilan dari file template.blade.php menjadi seperti berikut



1. Layout terakhir adalah pada bagian konten. Layout untuk konten bisa kita buat dinamis, sesuai dengan apa yang ingin kita sajikan pada web yang kita bangun.
2. Untuk ***content***, kita akan menghapus baris **42-66** pada file template.blade.php. dan kita ganti dengan kode seperti ini @yield('content')
3. Hasil akhir pada file utama layouts/template.blade.php adalah seperti berikut

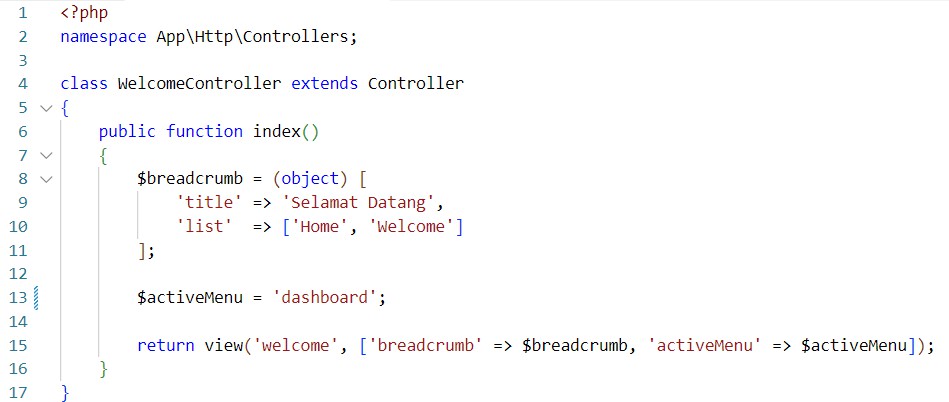


1. Selamat kalian sudah selesai dalam melakukan layouting website di laravel.
2. Jangan lupa commit dan push ke github untuk praktikum 1 ini

**Praktikum 2 - Penerapan Layouting:**

Sekarang kita akan mencoba melakukan penerapan terhadap layouting yang sudah kita lakukan.

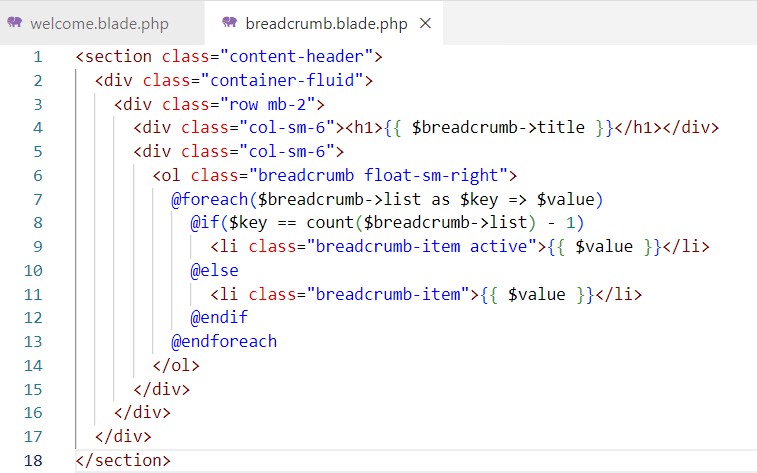
1. Kita buat file controller dengan nama WelcomeController.php



1. Kita buat file pada PWL\_POS/resources/views/welcome.blade.php



1. Kita modifikasi file PWL\_POS/resources/views/layouts/breadcrumb.blade.php



1. Kita modifikasi file PWL\_POS/resources/views/layouts/sidebar.blade.php

<div class="sidebar">

<!-- SidebarSearch Form -->

<div class="form-inline mt-2">

<div class="input-group" data-widget="sidebar-search">

<input class="form-control form-control-sidebar" type="search" placeholder="Search" aria-label="Search">

<div class="input-group-append">

<button class="btn btn-sidebar">

<i class="fas fa-search fa-fw"></i>

</button>

</div>

</div>

</div>

<!-- Sidebar Menu -->

<nav class="mt-2">

<ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview" role="menu" data-accordion="false">

<li class="nav-item">

<a href="{{ url('/') }}" class="nav-link {{ ($activeMenu == 'dashboard')? 'active' : '' }} ">

<i class="nav-icon fas fa-tachometer-alt"></i>

<p>Dashboard</p>

</a>

</li>

<li class="nav-header">Data Pengguna</li>

<li class="nav-item">

<a href="{{ url('/level') }}" class="nav-link {{ ($activeMenu == 'level')? 'active' : '' }} ">

<i class="nav-icon fas fa-layer-group"></i>

<p>Level User</p>

</a>

</li>

<li class="nav-item">

<a href="{{ url('/user') }}" class="nav-link {{ ($activeMenu == 'user')? 'active' : '' }}">

<i class="nav-icon far fa-user"></i>

<p>Data User</p>

</a>

</li>

<li class="nav-header">Data Barang</li>

<li class="nav-item">

<a href="{{ url('/kategori') }}" class="nav-link {{ ($activeMenu == 'kategori')? 'active' : '' }} ">

<i class="nav-icon far fa-bookmark"></i>

<p>Kategori Barang</p>

</a>

</li>

<li class="nav-item">

<a href="{{ url('/barang') }}" class="nav-link {{ ($activeMenu == 'barang')? 'active' : '' }} ">

<i class="nav-icon far fa-list-alt"></i>

<p>Data Barang</p>

</a>

</li>

<li class="nav-header">Data Transaksi</li>

<li class="nav-item">

<a href="{{ url('/stok') }}" class="nav-link {{ ($activeMenu == 'stok')? 'active' : '' }} ">

<i class="nav-icon fas fa-cubes"></i>

<p>Stok Barang</p>

</a>

</li>

<li class="nav-item">

<a href="{{ url('/barang') }}" class="nav-link {{ ($activeMenu == 'penjualan')? 'active' : '' }} ">

<i class="nav-icon fas fa-cash-register"></i>

<p>Transaksi Penjualan</p>

</a>

</li>

</ul>

</nav>

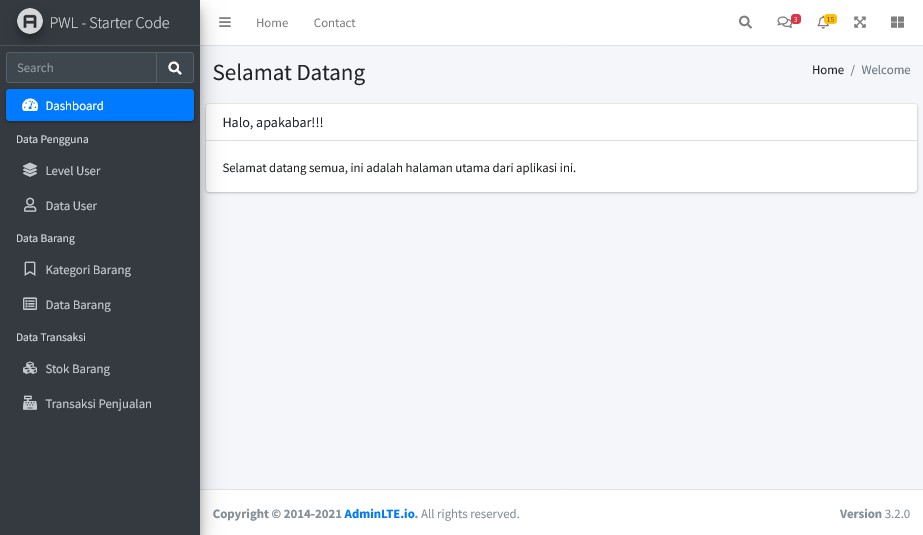
</div>

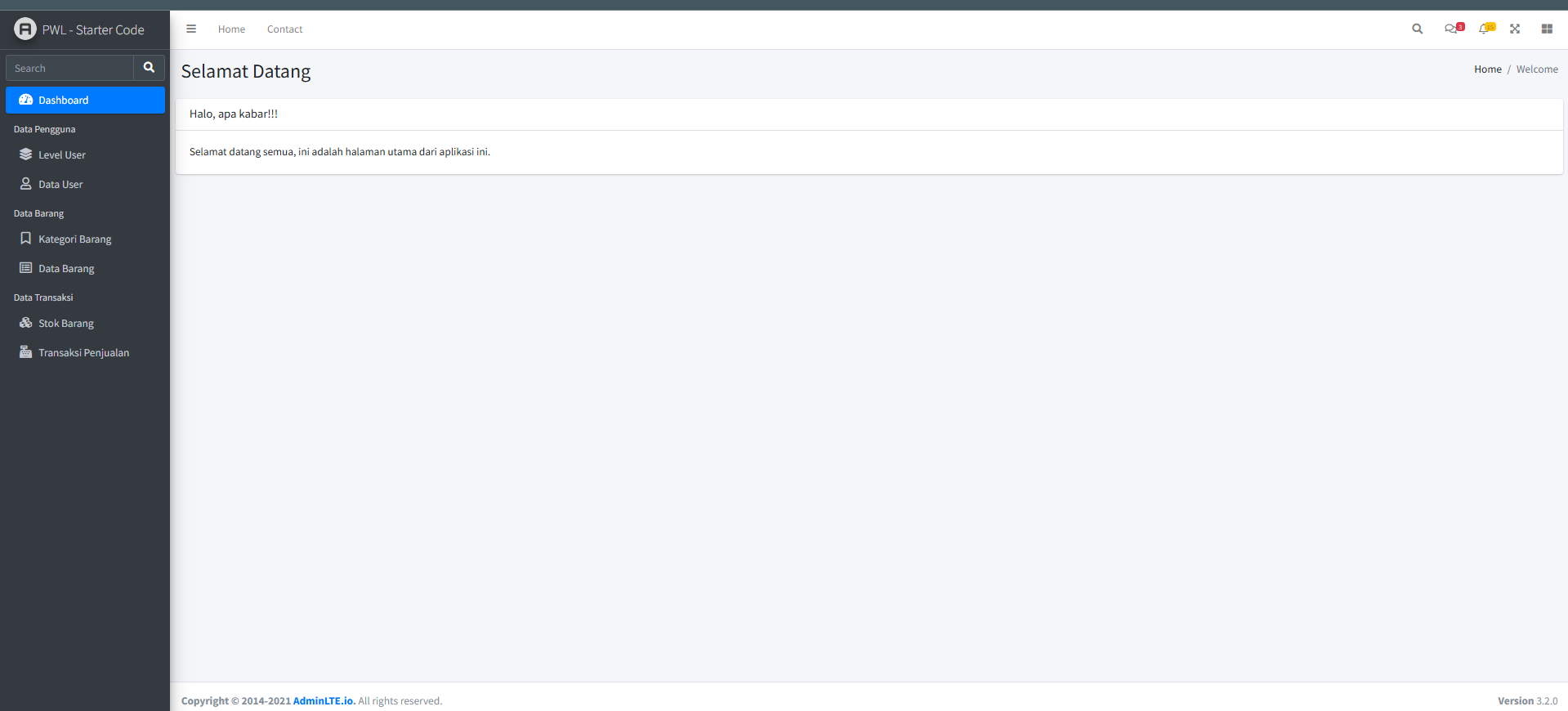
1. Kita tambahkan kode berikut router web.php



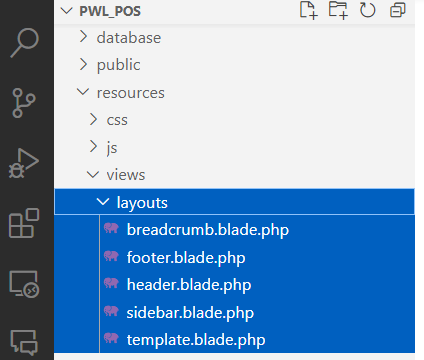
1. Sekarang kita coba jalankan di browser dengan mengetikkan url

<http://localhost/PWL_POS/public>





1. Jangan lupa commit dan push ke github PWL\_POS kalian



**INFO**

Terdapat 5 file utama pada folder *layouts* yang digunakan dalam membangun sebuah website, yaitu ***template***, ***header***, ***sidebar***, ***breadcrumb***, ***footer****.* Kalian bisa memodifikasi ke-5 file tersebut dalam membangun website.

**D. jQuery Datatables di Laravel**

jQuery DataTables adalah sebuah plugin jQuery yang sangat populer dan powerful untuk menampilkan dan mengelola data dalam bentuk tabel di halaman web. Untuk menampilkan banyak data, kita bisa menampilkan data tersebut dalam format tabel. DataTable ini merupakan plugin **jQuery** yang dibuat untuk mengelola data informasi dalam bentuk grid / table.

**INFO**

AdminLTE juga sudah menerapkan library Datatable dalam template nya. Datatable pada AdminLTE dapat kalian cek di

<http://localhost/PWL/public/adminlte/pages/tables/data.html>

Pada praktikum kali ini kita **tidak menggunakan Vite/NodeJS** dalam mengelola datatables, kita cukup menggunakan jQuery datatables bawaan AdminLTE dan library Yajra *laravel-datatables.*

**E. Implementasi jQuery Datatables di Laravel**

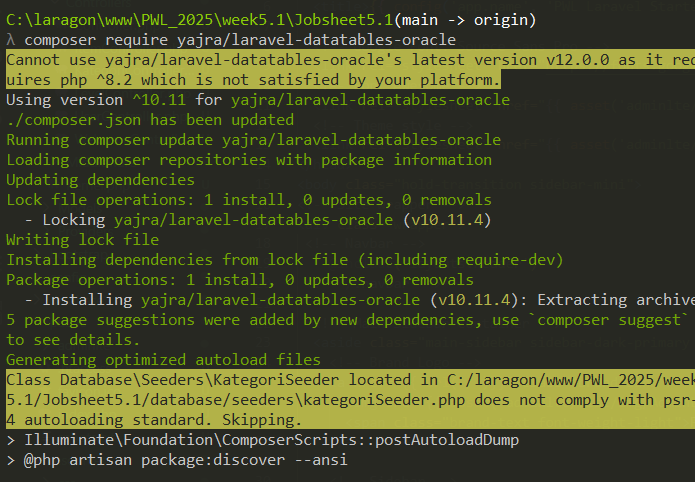
Kita akan menerapkan penggunakan jQuery datatable dari AdminLTE dengan Laravel. Dalam penerapan ini, kita menggunakan library Yajra laravel-datatables.

**Praktikum 3 – Implementasi jQuery Datatable di AdminLTE :**

1. Kita modifikasi proses CRUD pada tabel m\_user pada praktikum ini
2. Kita gunakan library Yajra-datatable dengan mengetikkan perintah pada CMD

composer require yajra/laravel-datatables:^10.0 atau

composer require yajra/laravel-datatables-oracle

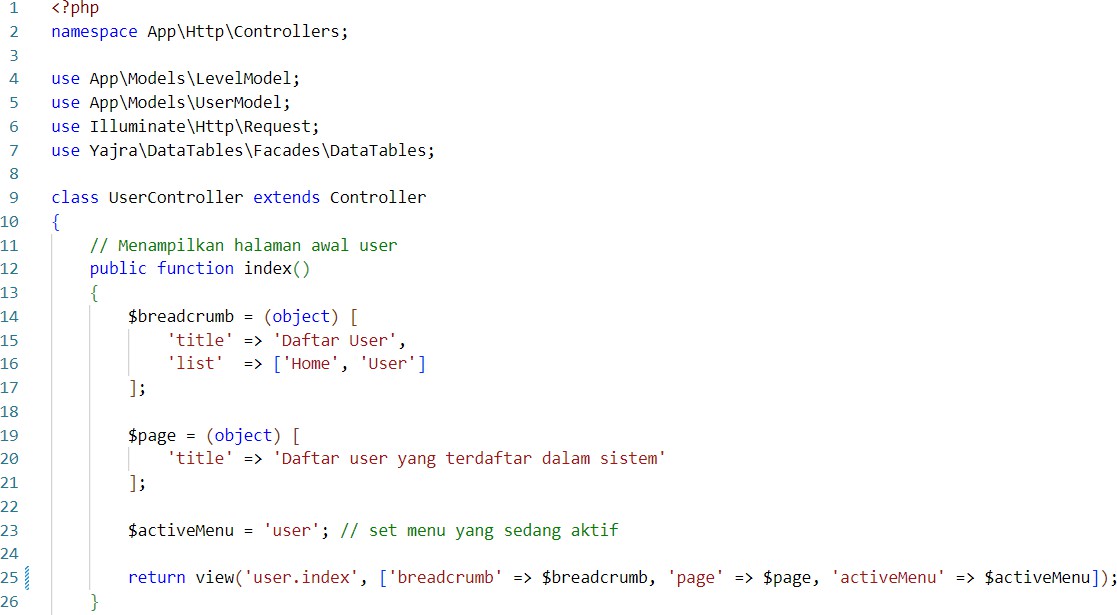


1. Kita modifikasi route web.php untuk proses CRUD user



1. Kita buat atau modifikasi penuh untuk UserController.php. Kita buat fungsi **index()**

untuk menampilkan halaman awal user



1. Lalu kita buat view pada PWL/resources/views/user/index.blade.php

@extends('layouts.template')

@section('content')

<div class="card card-outline card-primary">

<div class="card-header">

<h3 class="card-title">{{ $page->title }}</h3>

<div class="card-tools">

<a class="btn btn-sm btn-primary mt-1" href="{{ url('user/create') }}">Tambah</a>

</div>

</div>

<div class="card-body">

<table class="table table-bordered table-striped table-hover table-sm" id="table\_user">

<thead>

<tr><th>ID</th><th>Username</th><th>Nama</th><th>Level Pengguna</th><th>Aksi</th></tr>

</thead>

</table>

</div>

</div> @endsection

@push('css') @endpush

@push('js')

<script>

$(document).ready(function() {

var dataUser = $('#table\_user').DataTable({

// serverSide: true, jika ingin menggunakan server side processing serverSide: true,

ajax: {

"url": "{{ url('user/list') }}", "dataType": "json",

"type": "POST"

},

columns: [

{

// nomor urut dari laravel datatable addIndexColumn() data: "DT\_RowIndex",

className: "text-center", orderable: false, searchable: false

},{

data: "username", className: "",

// orderable: true, jika ingin kolom ini bisa diurutkan orderable: true,

// searchable: true, jika ingin kolom ini bisa dicari searchable: true

},{

data: "nama",

className: "", orderable: true, searchable: true

},{

// mengambil data level hasil dari ORM berelasi data: "level.level\_nama",

className: "", orderable: false, searchable: false

},{

data: "aksi",

className: "", orderable: false, searchable: false

}

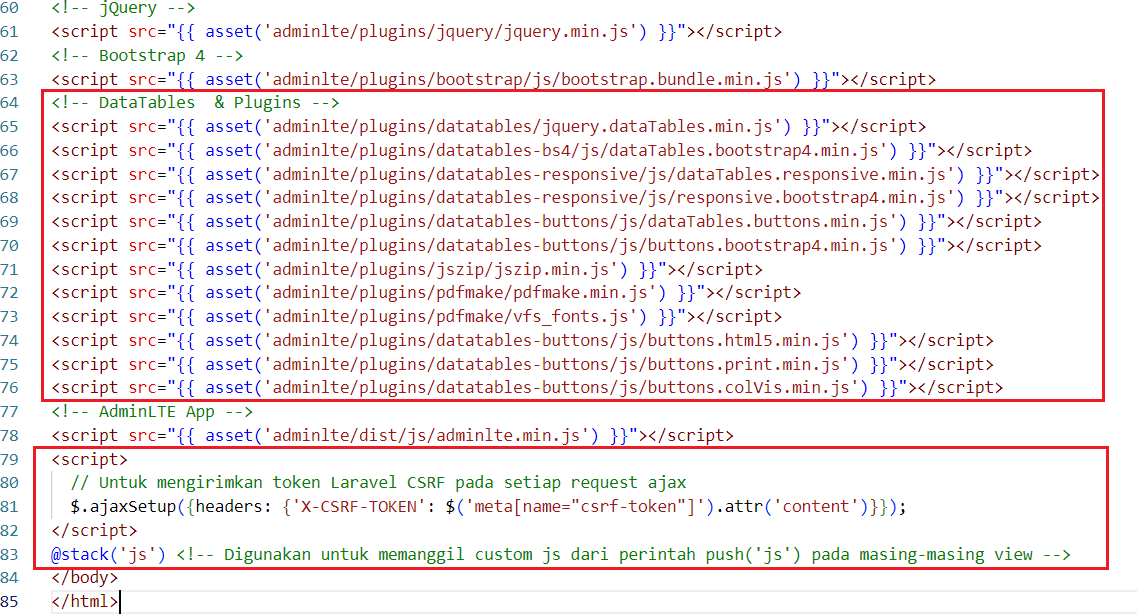
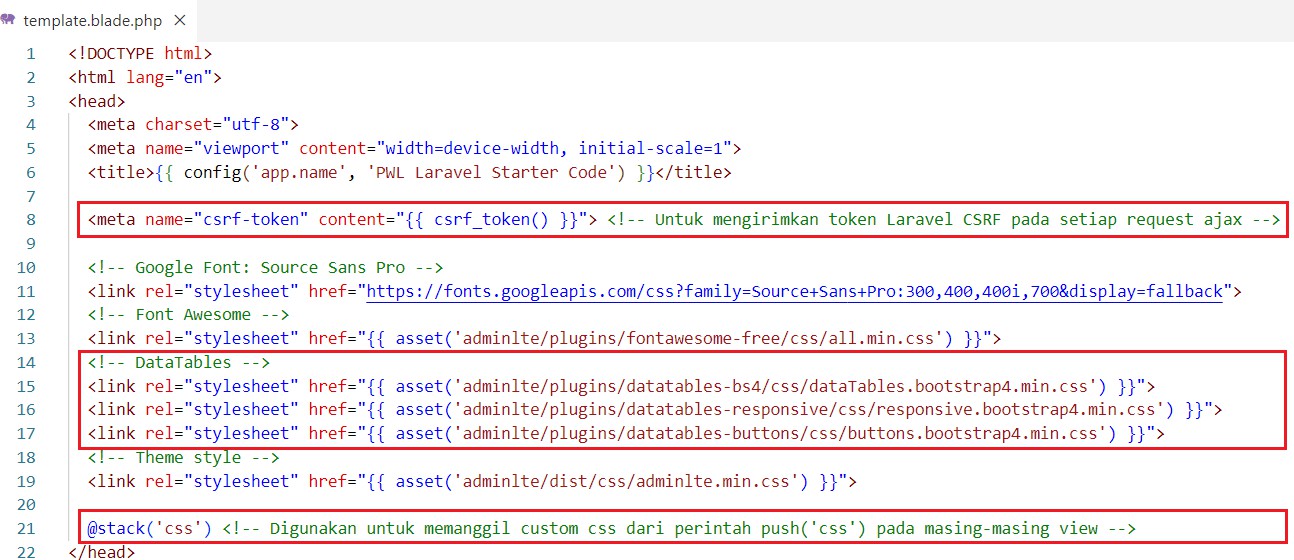
]

});

});

</script> @endpush

1. Kemudian kita modifikasi file template.blade.php untuk menambahkan library jquery datatables dari template AdminLTE yang kita download dan berada di folder public



1. Untuk bisa menangkap request data untuk datatable, kita buat fungsi **list()** pada

UserController.php seperti berikut

// Ambil data user dalam bentuk json untuk datatables public function list(Request $request)

{

$users = UserModel::select('user\_id', 'username', 'nama', 'level\_id')

->with('level');

return DataTables::of($users)

// menambahkan kolom index / no urut (default nama kolom: DT\_RowIndex)

->addIndexColumn()

->addColumn('aksi', function ($user) { // menambahkan kolom aksi

$btn = '<a href="'.url('/user/' . $user->user\_id).'" class="btn btn-info btn- sm">Detail</a> ';

$btn .= '<a href="'.url('/user/' . $user->user\_id . '/edit').'" class="btn btn- warning btn-sm">Edit</a> ';

$btn .= '<form class="d-inline-block" method="POST" action="'. url('/user/'.$user->user\_id).'">'

. csrf\_field() . method\_field('DELETE') .

'<button type="submit" class="btn btn-danger btn-sm" onclick="return confirm(\'Apakah Anda yakit menghapus data ini?\');">Hapus</button></form>';

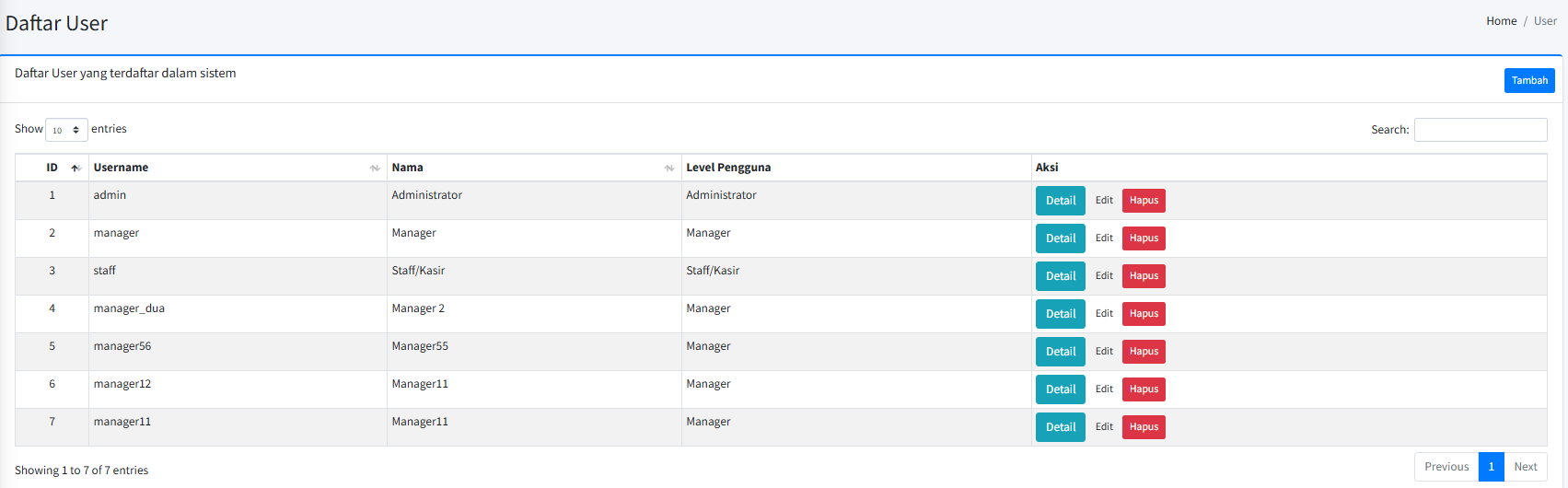
return $btn;

})

->rawColumns(['aksi']) // memberitahu bahwa kolom aksi adalah html

->make(true);

}

1. Sekarang coba jalankan browser, dan klik menu **Data User..!!!** perhatikan dan amati apa yang terjadi.  
    output sesuai dengan yang dikerjakan pada jobsheet sebelumnya
2. Selanjutnya kita modifikasi UserController.php untuk form tambah data user



1. Sekarang kita buat form untuk menambah data, kita buat file

PWL/resources/views/user/create.blade.php

@extends('layouts.template')

@section('content')

<div class="card card-outline card-primary">

<div class="card-header">

<h3 class="card-title">{{ $page->title }}</h3>

<div class="card-tools"></div>

</div>

<div class="card-body">

<form method="POST" action="{{ url('user') }}" class="form-horizontal"> @csrf

<div class="form-group row">

<label class="col-1 control-label col-form-label">Level</label>

<div class="col-11">

<select class="form-control" id="level\_id" name="level\_id" required>

<option value="">- Pilih Level -</option> @foreach($level as $item)

<option value="{{ $item->level\_id }}">{{ $item->level\_nama }}</option> @endforeach

</select> @error('level\_id')

<small class="form-text text-danger">{{ $message }}</small> @enderror

</div>

</div>

<div class="form-group row">

<label class="col-1 control-label col-form-label">Username</label>

<div class="col-11">

<input type="text" class="form-control" id="username" name="username" value="{{ old('username') }}" required>

@error('username')

<small class="form-text text-danger">{{ $message }}</small> @enderror

</div>

</div>

<div class="form-group row">

<label class="col-1 control-label col-form-label">Nama</label>

<div class="col-11">

<input type="text" class="form-control" id="nama" name="nama" value="{{ old('nama') }}" required>

@error('nama')

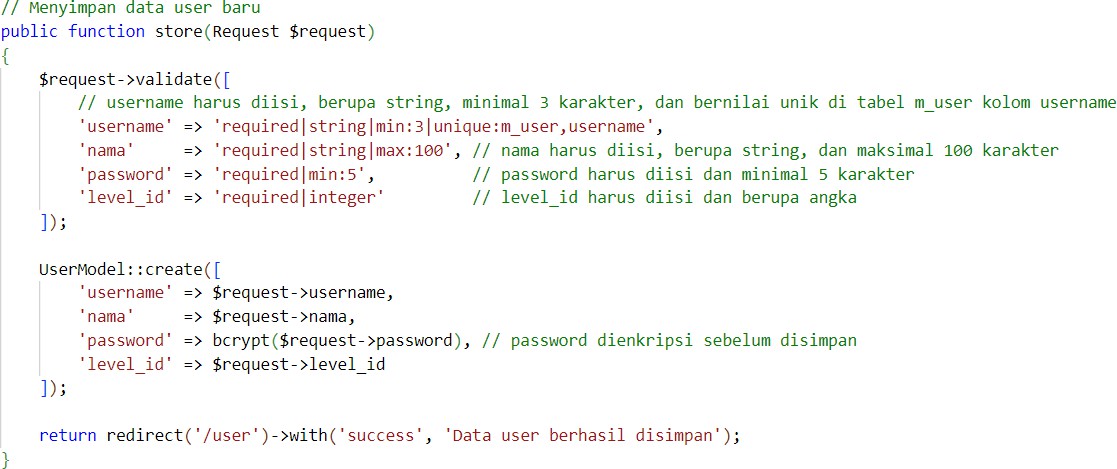
<small class="form-text text-danger">{{ $message }}</small> @enderror

</div>

</div>

<div class="form-group row">

<label class="col-1 control-label col-form-label">Password</label>



<div class="col-11">

<input type="password" class="form-control" id="password" name="password"

required>

@error('password')

<small class="form-text text-danger">{{ $message }}</small> @enderror

</div>

</div>

<div class="form-group row">

<label class="col-1 control-label col-form-label"></label>

<div class="col-11">

<button type="submit" class="btn btn-primary btn-sm">Simpan</button>

<a class="btn btn-sm btn-default ml-1" href="{{ url('user') }}">Kembali</a>

</div>

</div>

</form>

</div>

</div> @endsection @push('css') @endpush @push('js') @endpush

1. Kemudian untuk bisa menng-*handle* data yang akan disimpan ke database, kita buat fungsi **store()** di UserController.php
2. Sekarang coba kalian buka form tambah data user dengan klik tombol tambah. Amati dan pelajari..!!!
3. Selanjutnya, kita masuk pada bagian menampilkan detail data user (klik tombol  ) pada halaman user. Route yang bertugas untuk menangkap request detail adalah
4. Jadi kita buat/modifikasi fungsi **show()** pada UserController.php seperti berikut



1. Kemudian kita buat *view* di PWL/resources/views/user/show.blade.php

@extends('layouts.template')

@section('content')

<div class="card card-outline card-primary">

<div class="card-header">

<h3 class="card-title">{{ $page->title }}</h3>

<div class="card-tools"></div>

</div>

<div class="card-body"> @empty($user)

<div class="alert alert-danger alert-dismissible">

<h5><i class="icon fas fa-ban"></i> Kesalahan!</h5> Data yang Anda cari tidak ditemukan.

</div> @else

<table class="table table-bordered table-striped table-hover table-sm">

<tr>

<th>ID</th>

<td>{{ $user->user\_id }}</td>

</tr>

<tr>

<th>Level</th>

<td>{{ $user->level->level\_nama }}</td>

</tr>

<tr>

<th>Username</th>

<td>{{ $user->username }}</td>

</tr>

<tr>

<th>Nama</th>

<td>{{ $user->nama }}</td>

</tr>

<tr>

<th>Password</th>

<td>\*\*\*\*\*\*\*\*</td>

</tr>

</table> @endempty

<a href="{{ url('user') }}" class="btn btn-sm btn-default mt-2">Kembali</a>

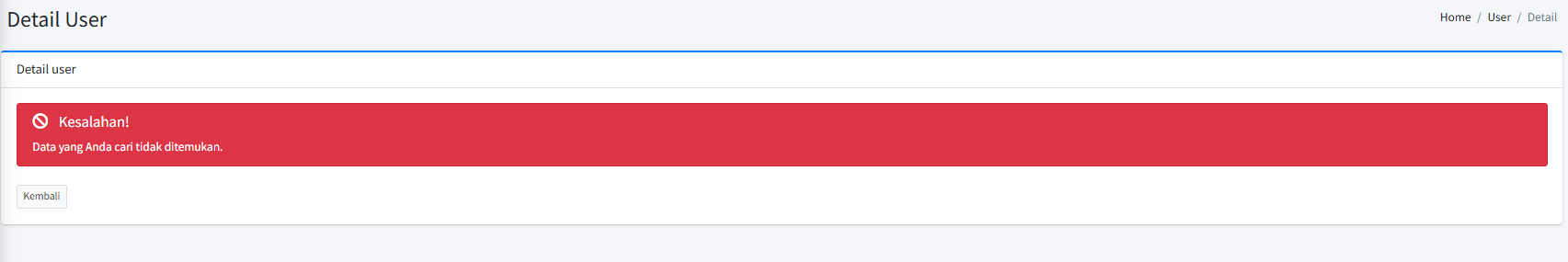
</div>

</div> @endsection

@push('css') @endpush

@push('js') @endpush

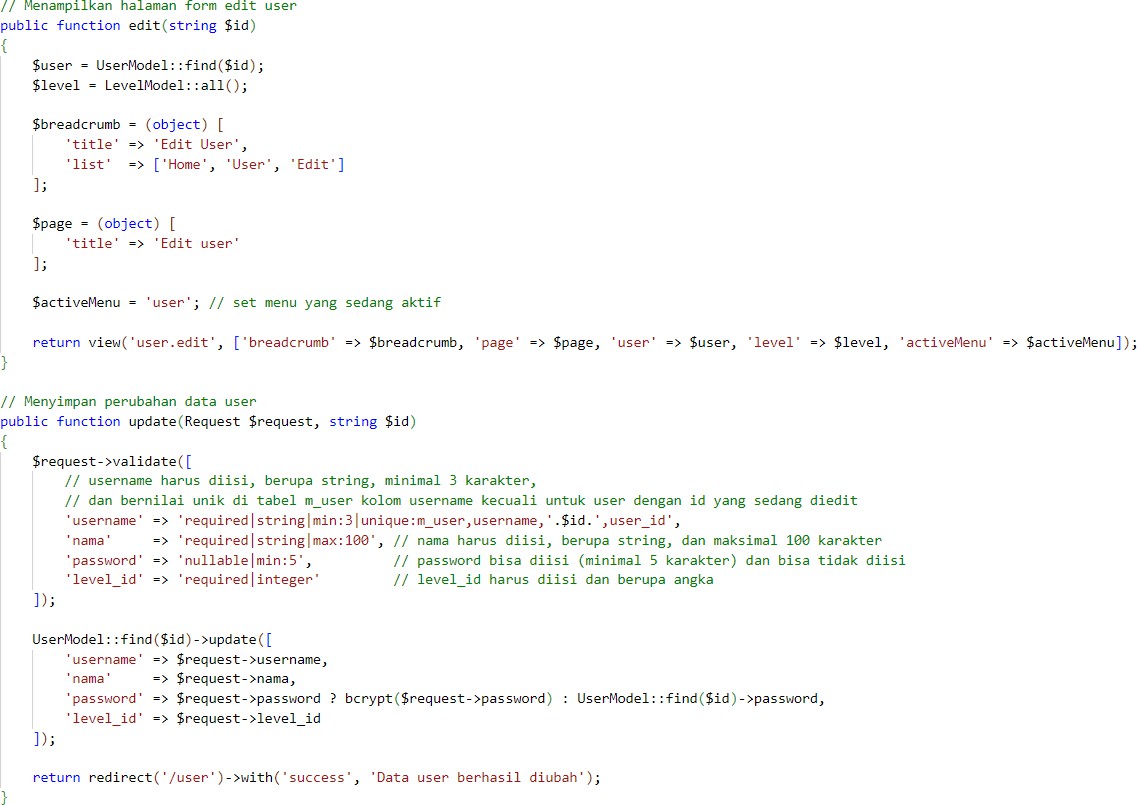
1. Sekarang kalian coba untuk melihat detail data user di browser, dan coba untuk mengetikkan id yang salah contoh [http://localhost/PWL/public/user/100](http://localhost/PWL_POS/public/user/100) amati apa yang terjadi, dan laporkan!!!

 terjadi eror karena tidak ada user 100

1. Selanjutnya, kita masuk pada bagian untuk memodifikasi data user. Route yang bertugas untuk menangkap request edit adalah



1. Jadi kita buat fungsi **edit()** dan **update()** pada UserController.php



1. Selanjutnya, kita buat *view* untuk melakukan proses edit data user di

PWL/resources/views/user/edit.blade.php

@extends('layouts.template')

@section('content')

<div class="card card-outline card-primary">

<div class="card-header">

<h3 class="card-title">{{ $page->title }}</h3>

<div class="card-tools"></div>

</div>

<div class="card-body"> @empty($user)

<div class="alert alert-danger alert-dismissible">

<h5><i class="icon fas fa-ban"></i> Kesalahan!</h5> Data yang Anda cari tidak ditemukan.

</div>

<a href="{{ url('user') }}" class="btn btn-sm btn-default mt-2">Kembali</a> @else

<form method="POST" action="{{ url('/user/'.$user->user\_id) }}" class="form- horizontal">

@csrf

{!! method\_field('PUT') !!} <!-- tambahkan baris ini untuk proses edit yang butuh method PUT -->

<div class="form-group row">

<label class="col-1 control-label col-form-label">Level</label>

<div class="col-11">

<select class="form-control" id="level\_id" name="level\_id" required>

<option value="">- Pilih Level -</option> @foreach($level as $item)

<option value="{{ $item->level\_id }}" @if($item->level\_id == $user-

>level\_id) selected @endif>{{ $item->level\_nama }}</option> @endforeach

</select> @error('level\_id')

<small class="form-text text-danger">{{ $message }}</small> @enderror

</div>

</div>

<div class="form-group row">

<label class="col-1 control-label col-form-label">Username</label>

<div class="col-11">

<input type="text" class="form-control" id="username" name="username" value="{{ old('username', $user->username) }}" required>

@error('username')

<small class="form-text text-danger">{{ $message }}</small> @enderror

</div>

</div>

<div class="form-group row">

<label class="col-1 control-label col-form-label">Nama</label>

<div class="col-11">

<input type="text" class="form-control" id="nama" name="nama" value="{{ old('nama', $user->nama) }}" required>

@error('nama')

<small class="form-text text-danger">{{ $message }}</small> @enderror

</div>

</div>

<div class="form-group row">

<label class="col-1 control-label col-form-label">Password</label>

<div class="col-11">

<input type="password" class="form-control" id="password" name="password"> @error('password')

<small class="form-text text-danger">{{ $message }}</small> @else

<small class="form-text text-muted">Abaikan (jangan diisi) jika tidak ingin mengganti password user.</small>

@enderror

</div>

</div>

<div class="form-group row">

<label class="col-1 control-label col-form-label"></label>

<div class="col-11">

<button type="submit" class="btn btn-primary btn-sm">Simpan</button>

<a class="btn btn-sm btn-default ml-1" href="{{ url('user') }}">Kembali</a>

</div>

</div>

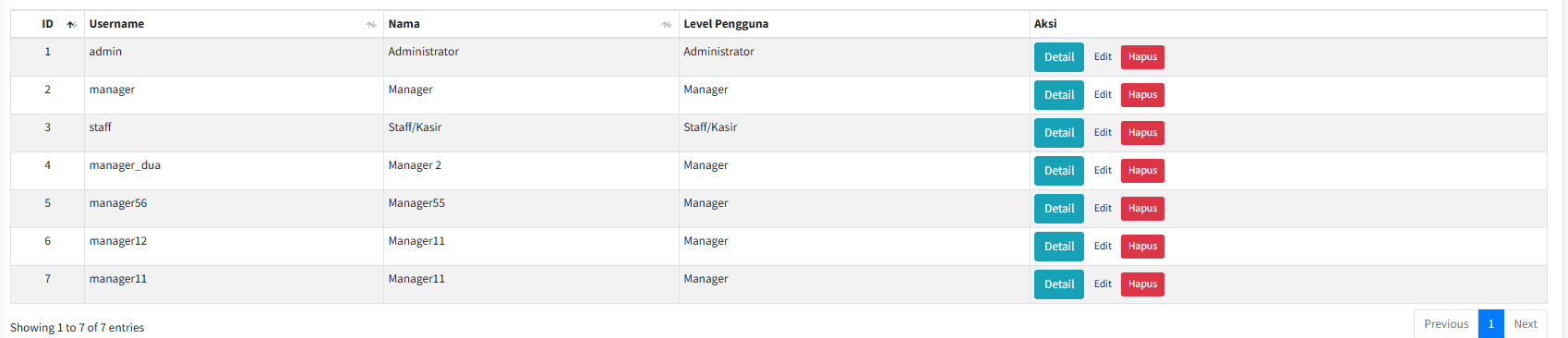
</form> @endempty

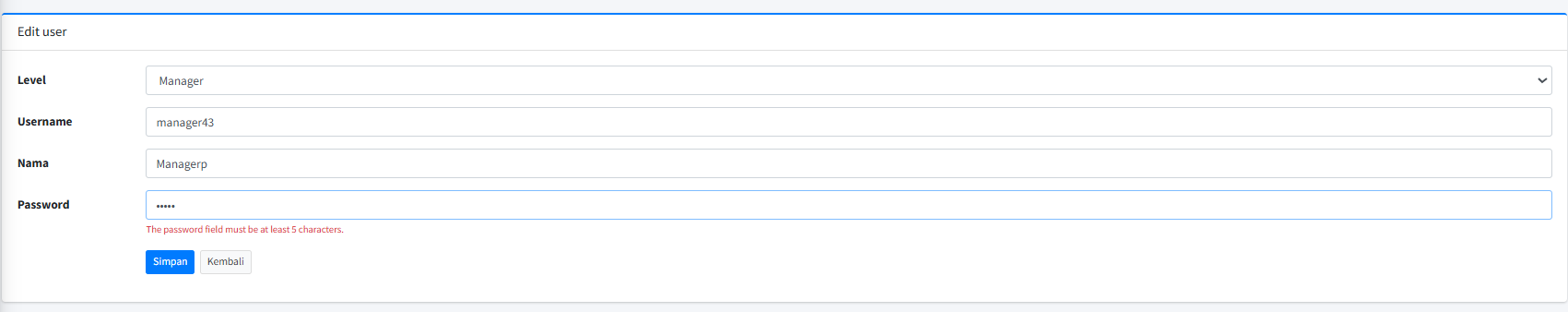
</div>

</div> @endsection

@push('css') @endpush @push('js') @endpush

1. Sekarang kalian coba untuk mengedit data user di browser, amati, pahami, dan laporkan!



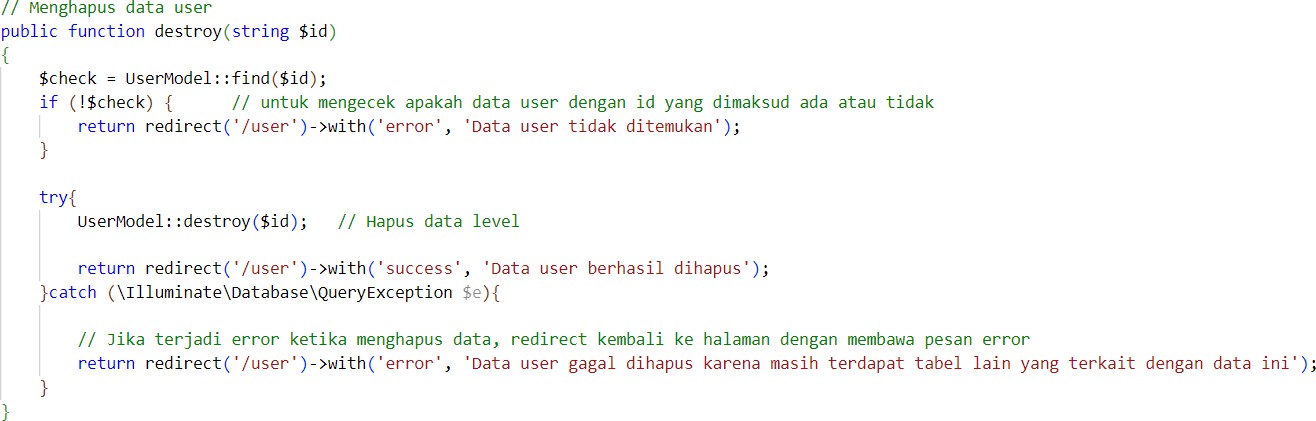




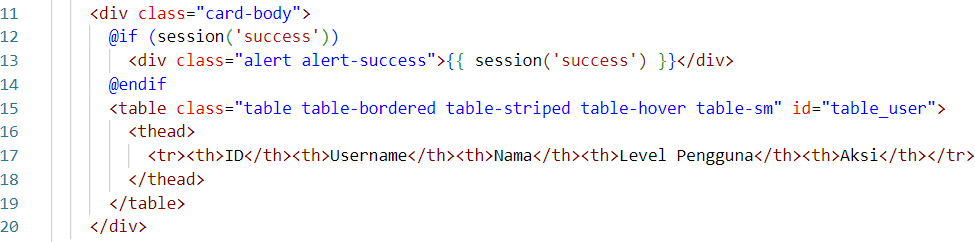
1. Selanjutnya kita akan membuat penanganan untuk tombol hapus. Router web.php yang berfungsi untuk menangkap request hapus dengan method DETELE adalah

Route::delete('/{id}', [UserController::class, 'destroy']);

1. Jadi kita buat fungsi **destroy()** pada UserController.php



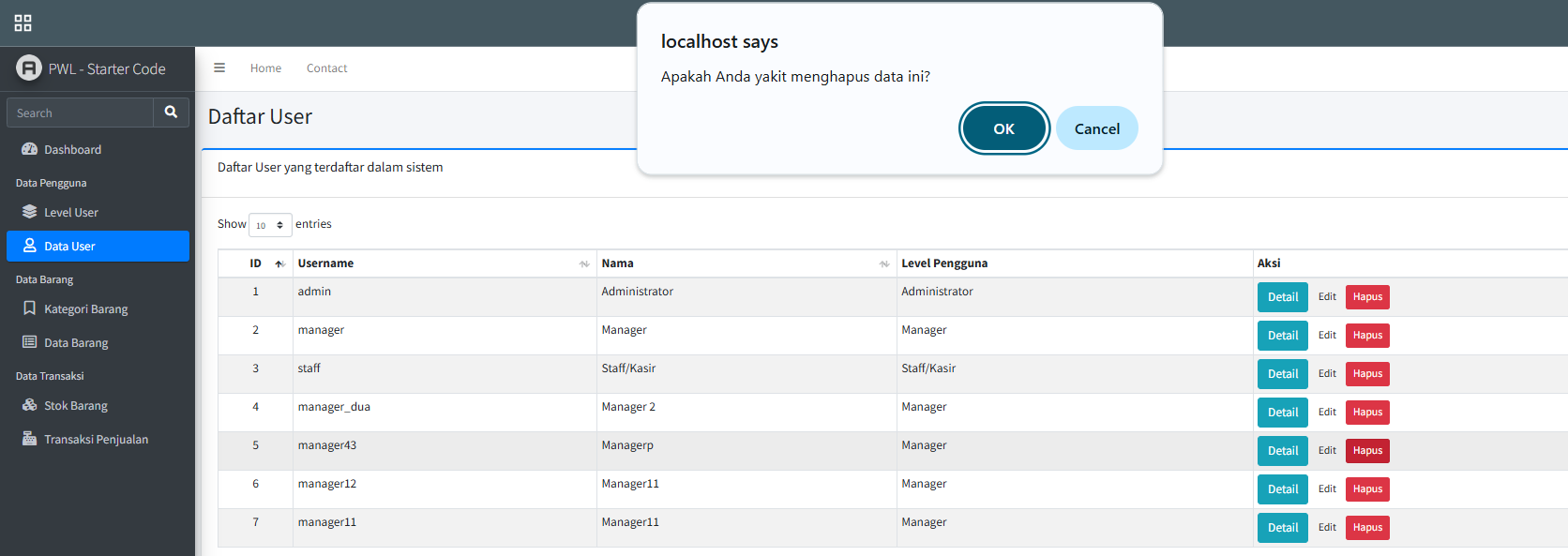
1. Selanjutnya kita modifikasi file PWL/resources/views/user/index.blade.php untuk menambahkan tampilan jika ada pesan error

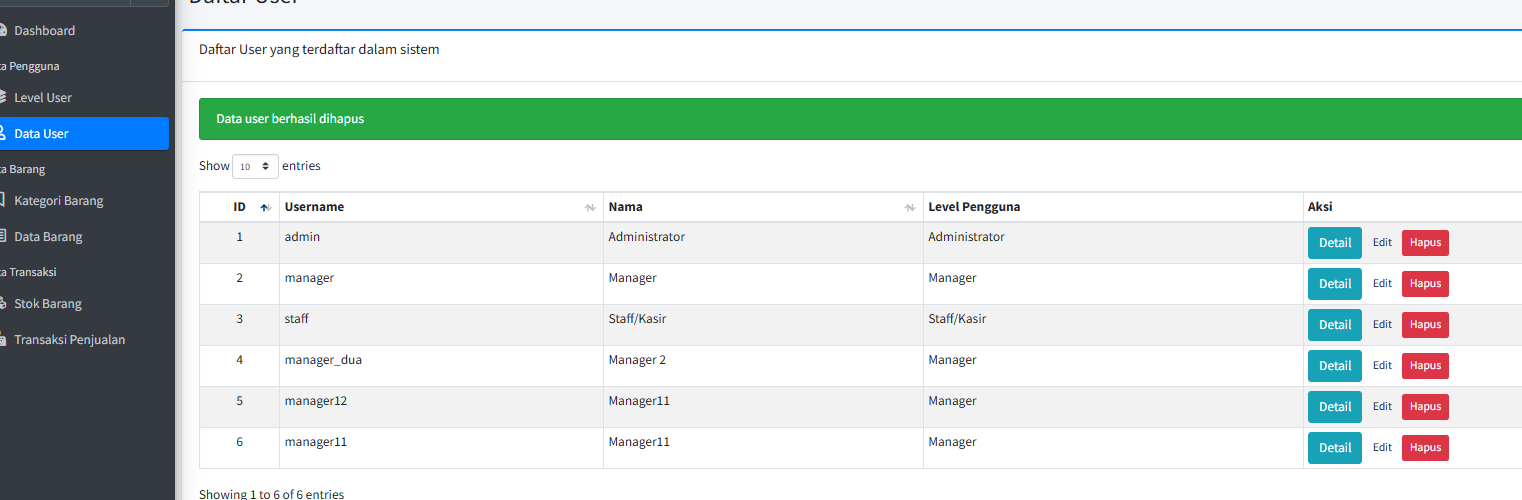


Menjadi



1. Kemudian jalankan browser untuk menghapus salah satu data user. Amati dan laporkan!





1. Selamat, kalian sudah membuat Laravel Starter Code untuk proses CRUD dengan menggunakan template AdminLTE dan plugin jQuery Datatables.
2. Jangan lupa commit dan push ke github PWL\_POS kalian

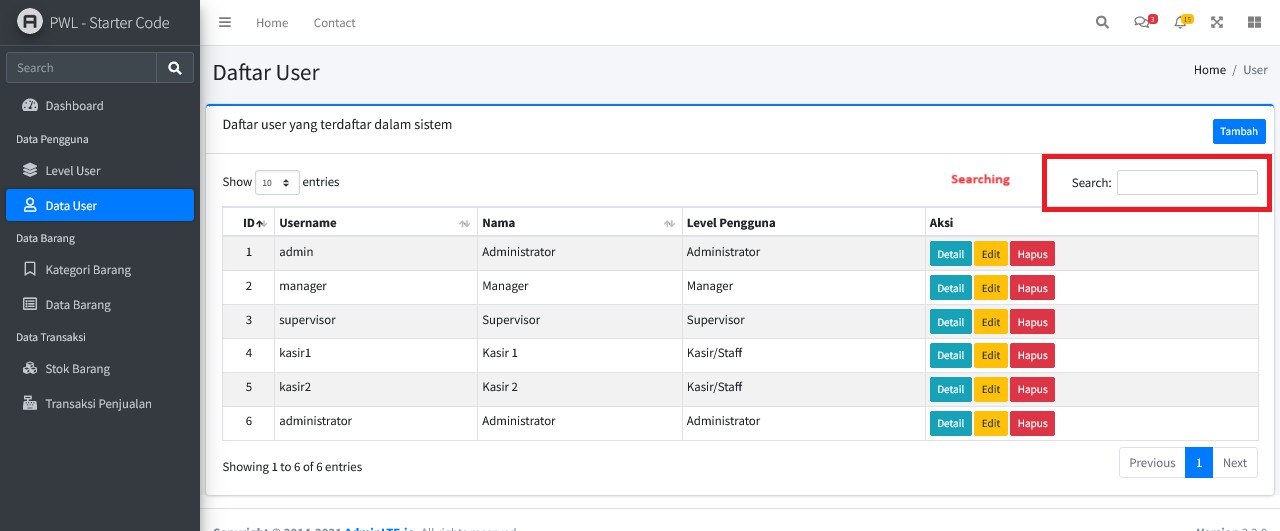
**F. Data Searching dan Filtering**

Dalam menampilkan sebuah data, kadang kita perlu yang namanya melakukan pencarian (*searching*) berdasarkan kata-kunci (*keyword*) tertentu ataupun penyaringan data berdasarkan suatu kategori (*filtering*).

1. ***Searching*** (Pencarian):
   * Searching adalah proses mencari informasi atau data tertentu yang sesuai dengan kriteria yang diberikan.
   * Biasanya, pencarian dilakukan dengan menggunakan kata kunci atau frasa tertentu untuk mencocokkan dengan data yang ada.
   * Tujuan dari pencarian adalah untuk menemukan entitas yang cocok dengan kriteria pencarian, baik itu dalam basis data, dalam dokumen teks, atau dalam konteks lainnya.
   * Pencarian sering kali melibatkan pencocokan pola atau kata kunci dalam teks atau data yang ada, dan hasilnya mungkin mencakup entitas yang sebagian cocok atau mirip dengan kriteria pencarian.
2. ***Filtering*** (Penyaringan):
   * Filtering adalah proses memilih atau membatasi sekumpulan data atau entitas berdasarkan kriteria tertentu.
   * Biasanya, filtering dilakukan untuk menyaring data yang sudah ada berdasarkan atribut atau karakteristik tertentu, seperti tanggal, kategori, atau atribut lainnya.
   * Tujuan dari penyaringan adalah untuk menyajikan data yang relevan atau relevan dengan kebutuhan pengguna atau kriteria tertentu.
   * Filtering dapat dilakukan dengan memilih data berdasarkan nilai yang sesuai dengan kriteria tertentu atau dengan mengecualikan data yang tidak memenuhi kriteria tersebut.

Perbedaan utama antara ***searching*** dan ***filtering*** adalah bahwa ***searching*** berkaitan dengan pencarian data yang sesuai dengan kriteria tertentu, sementara ***filtering*** berkaitan dengan penyaringan data yang sudah ada berdasarkan kriteria tertentu. Meskipun keduanya sering digunakan bersama-sama dalam aplikasi atau sistem informasi, mereka melayani tujuan yang berbeda dalam proses pengelolaan dan analisis data.

Secara default, jQuery datatables sudah memiliki fitur searching yang sudah terintegrasi dengan laravel yajra-datatables. Akan tetapi, belum ada untuk proses ***filtering*** datanya. Untuk itu kita akan membuat filtering data pada Laravel Starter Code kita.



**Praktikum 4 – Implementasi *Filtering* Datatables:**

Kita akan menerapkan filtering pada datatable yang sudah kita buat. Hal ini akan mempermudah kita dalam mengelompokkan suatu data sesuai kategori tertentu. Langkah- langkah yang kita kerjakan sebagai berikut

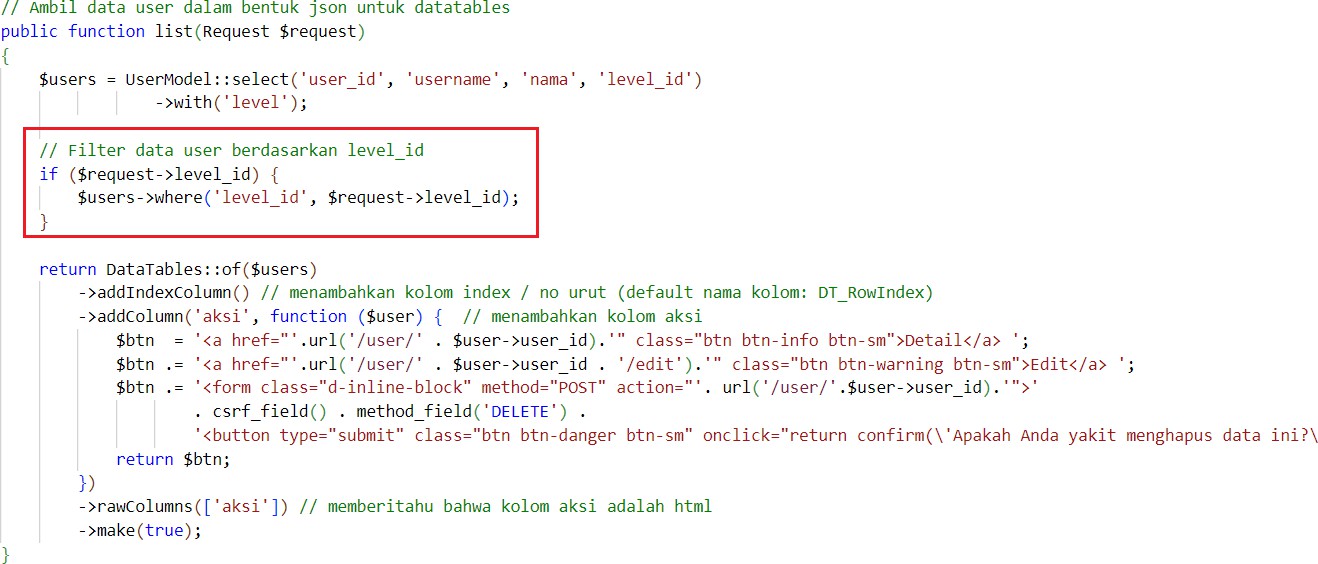
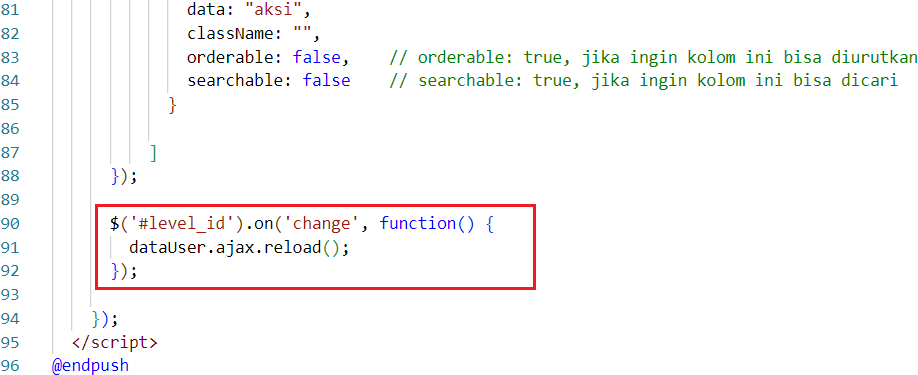
1. Kita modifikasi fungsi **index()** di UserController.php untuk menambahkan data yang ingin dijadikan kategori untuk data ***filtering***



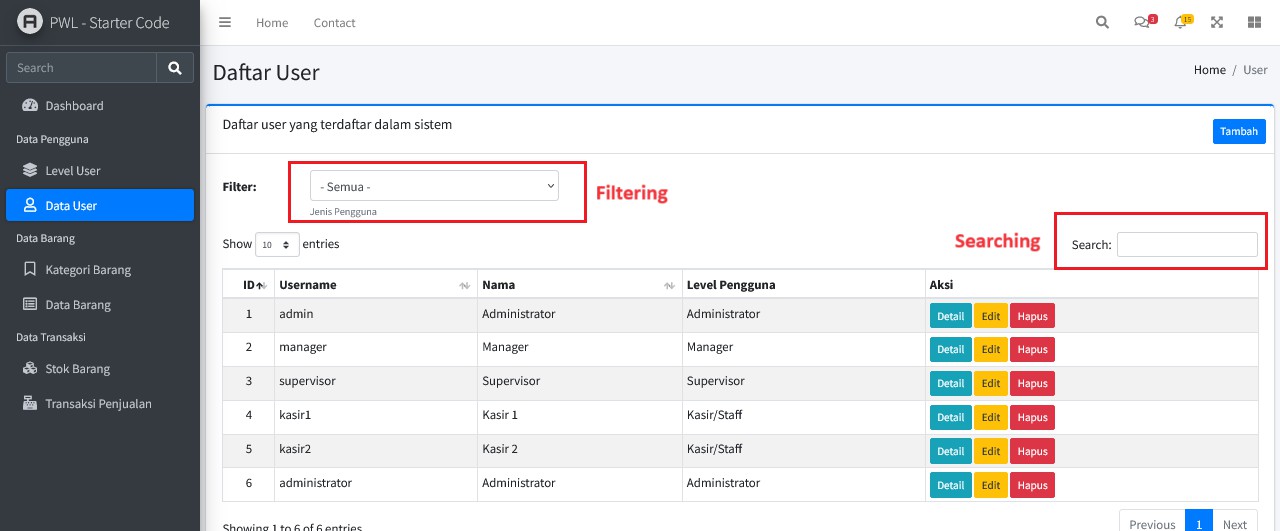
1. Kemudian kita modifikasi view untuk menampilkan data filtering pada

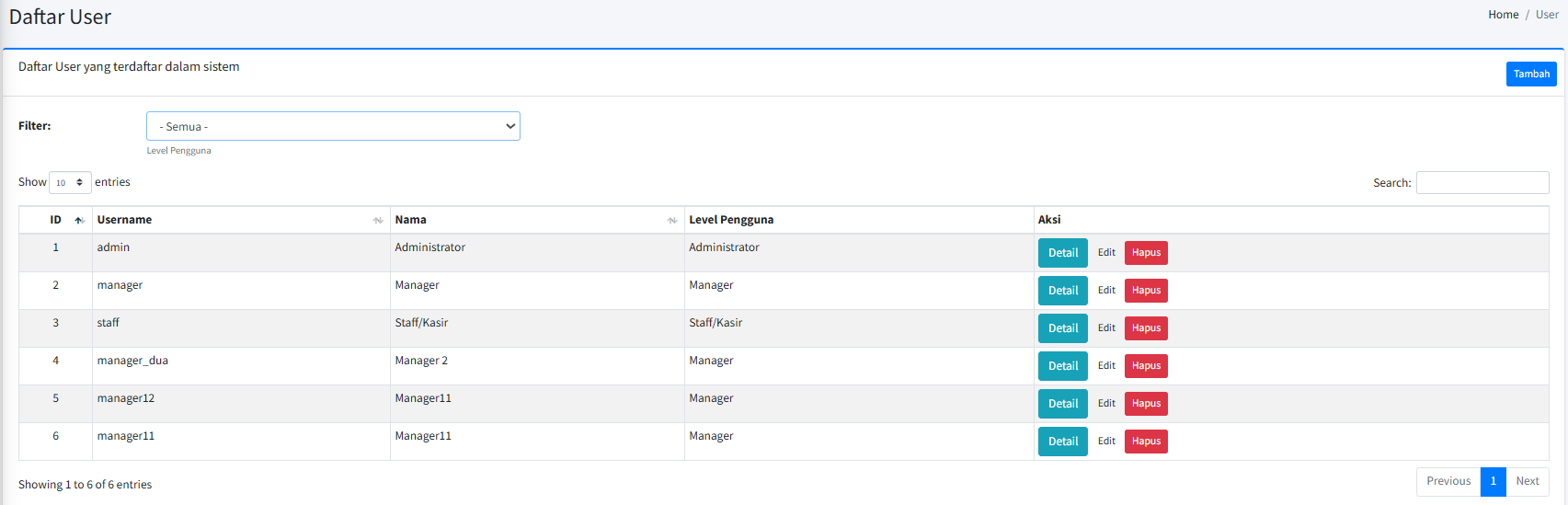
PWL/resources/views/user/index.blade.php

1. Selanjutnya, tetap pada view index.blade.php, kita tambahkan kode berikut pada deklarasi ajax di datatable. Kode ini digunakan untuk mengirimkan data untuk filtering



1. Kemudian kita edit pada bagian akhir script @push(‘js’) untuk menambahkan listener jika data filtering dipilih
2. Tahapan akhir adalah memodifikasi fungsi **list()** pada UserController.php yang digunakan untuk menampilkan data pada datatable
3. Bagian akhir adalah kita coba jalankan di browser dengan akses menu user, maka akan tampil seperti berikut

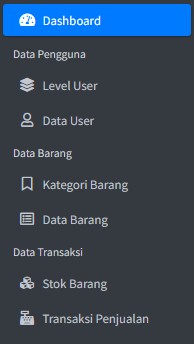


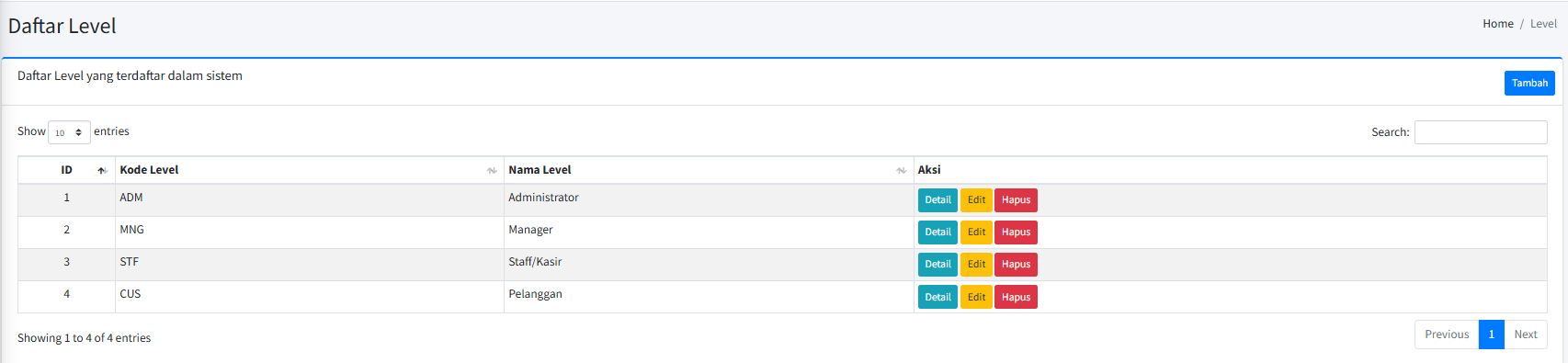


1. Selamat, sekarang Laravel Starter Code sudah ada filtering dan searching data. Starter Code sudah bisa digunakan dalam membangun sebuah sistem berbasis website.
2. Jangan lupa commit dan push ke github PWL\_POS kalian

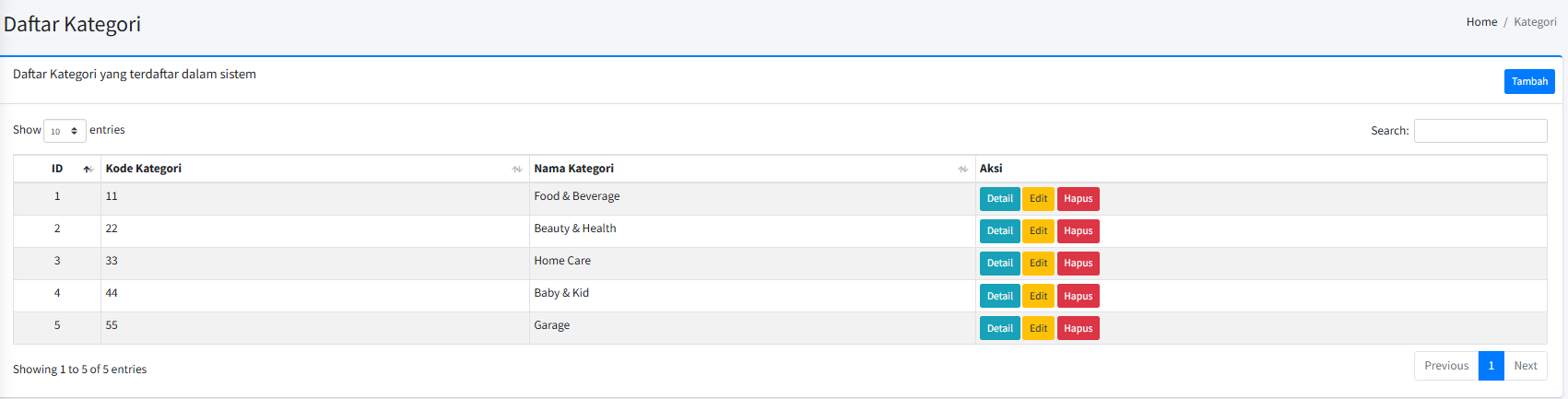
**G. Tugas**

Implementasikan web layout dan datatables, pada menu berikut ini

* + Tabel m\_level



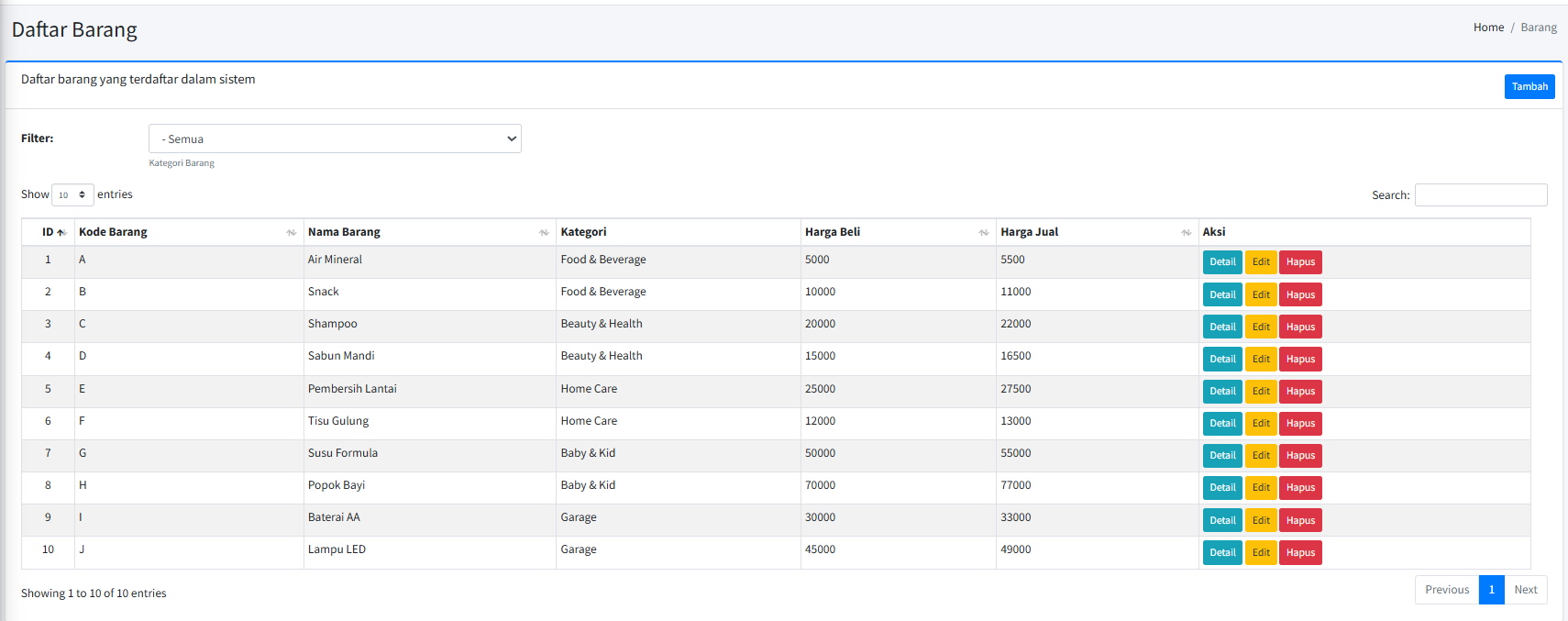
* + Tabel m\_kategori



* + Tabel m\_supplier

Tidak ada tabel m\_supplier

* + Tabel m\_barang



*\*\*\* Sekian, dan selamat belajar \*\*\**