LAPORAN JOBSHEET 5.2

BLADE VIEW, WEB TEMPLATING(ADMINLTE), DATATABLES

MATA KULIAH PEMROGRAMAN WEB LANJUT

Dosen Pengampu : Dimas Wahyu Wibowo, S.T., M.T.

Disusun oleh :

Dahniar Davina SIB-2A / 2341760023

JURUSAN TEKNOLOGI INFORMASI

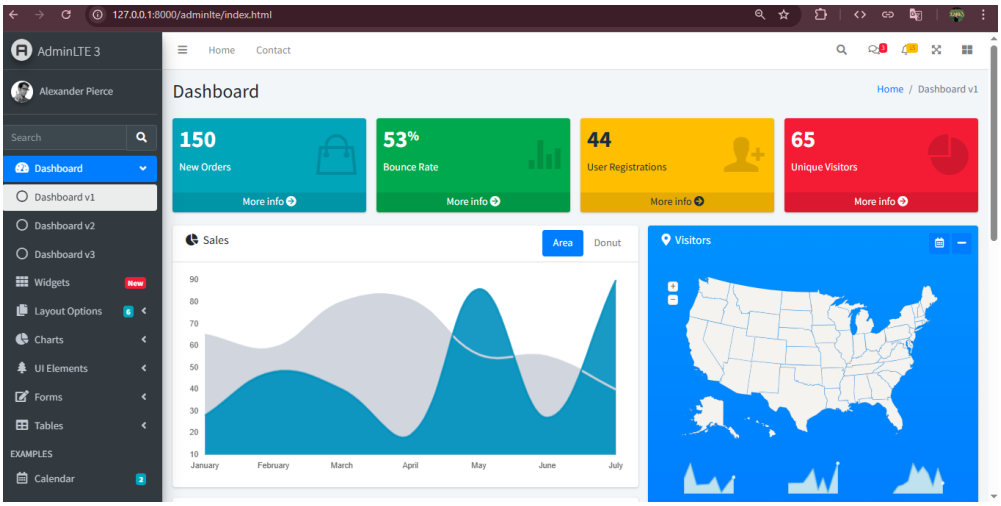
PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS

POLITEKNIK NEGERI MALANG

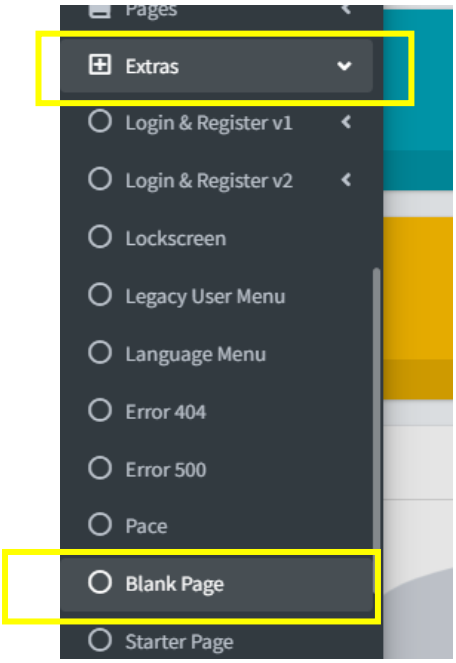
2025

A. PRAKTIKUM 1 : Layouting AdminLTE

1. Kita download AdminLTE v3.2.0 yang rilis pada 8 Feb 2022
2. Setelah kita berhasil download, kita ekstrak file yang sudah di download ke folder project PWL_POS/public, kemudian kita rename folder cukup menjadi adminlte
3. Selanjutnya kita buka di browser dengan alamat http://localhost/PWL_POS/public/adminlte maka akan muncul tampilan seperti berikut



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

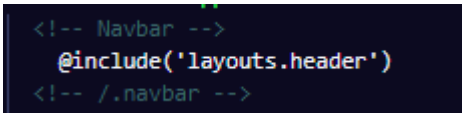


5. Dari sini kita bisa melakukan layouting halaman Blank Page ini menjadi 4 element seperti pada gambar berikut

6. Selanjutnya kita klik kanan halaman Blank Page dan klik view page source
7. 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)

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

10. 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.

11. Kita modifikasi baris 25 dan 26 pada template.blade.php

```
24  <!-- Brand Logo -->
25  a href="{{ url('/') }}" class="brand-link">
26   <img src="{{ assets('adminlte/dist/img/AdminLTELogo.png') }}" alt="AdminLTE L
27   <span class="brand-text font-weight-light">PWL - Starter Code</span>
```

12. 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

```
30      <!-- Sidebar -->
31      @include('layouts.sidebar')
32      <!-- /.sidebar -->
```

13. 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

```
86
87      @include('layouts.footer')
88  </div>
```

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

```
91  <!-- jQuery -->
92  <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
93  <!-- Bootstrap 4 -->
94  <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js')
95  <!-- AdminLTE App -->
96  <script src="{{ asset('adminlte/dist/js/adminlte.min.js') }}"></script>
97  </body>
98  </html>
```

15. Sekarang masuk pada bagian konten. Konten kita bagi menjadi 2, yaitu elemen untuk breadcrumb dan elemen untuk content.
16. Perhatikan file template.blade.php pada baris 38-52 kita jadikan sebagai elemen 4breadcrumb. 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

```
36    <div class="content-wrapper">
37      <!-- Content Header (Page header) -->
38       @include('layouts.breadcrumb')
39
```

17. 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.

18. Untuk content, kita akan menghapus baris 42-66 pada file template.blade.php. dan kita ganti dengan kode seperti ini @yield('content')

```
41      <section class="content">
42      @yield('content')
43
```

19. Hasil akhir pada file utama layouts/template.blade.php adalah seperti Berikut

```
18
19      @include('layouts.header')
20
21
22    <!-- Main Sidebar Container -->
23    <aside class="main-sidebar sidebar-dark-primary elevation-4">
24      <!-- Brand Logo -->
25      <a href="{{ url('/') }}" class="brand-link">
26        <img src="{{ assets('adminlte/dist/img/AdminLTELogo.png') }}" alt="Admin
27        <span class="brand-text font-weight-light">PWL - Starter Code</span>
28      </a>
29
30
31      @include('layouts.sidebar')
32
33    </aside>
34
35    <!-- Content Wrapper. Contains page content -->
36    <div class="content-wrapper">
37
38      @include('layouts.breadcrumb')
39
40      <!-- Main content -->
41
42      @yield('content')
43
44      </section>
45      <!-- /.content -->
46    </div>
47    <!-- /.content-wrapper -->
48
49      @include('layouts.footer')
50  </div>
```

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

B. PRAKTIKUM 2 : Penerapan Layouting

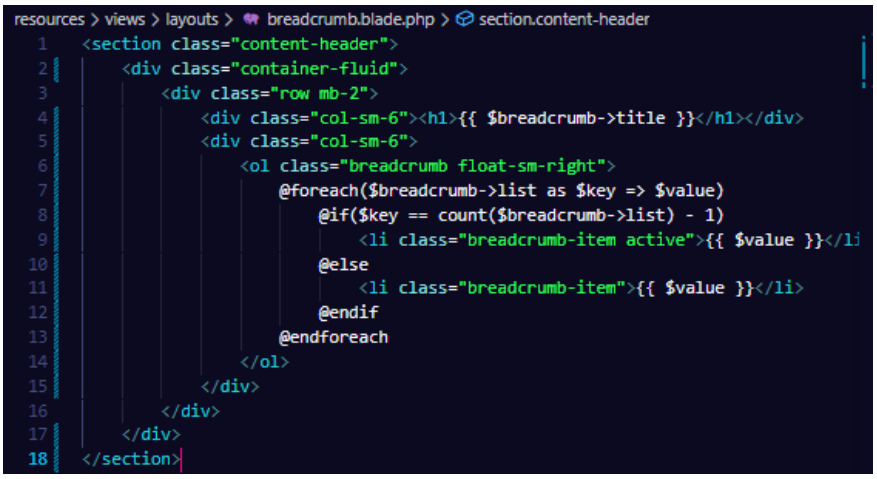
1. Kita buat file controller dengan nama WelcomeController.php

```
app > Http > Controllers > WelcomeController.php > ...
1   <?php
2   namespace App\Http\Controllers;
3
4   class WelcomeController extends Controller
5   {
6       public function index()
7       {
8           $breadcrumb = (object) [
9               'title' => 'Selamat Datang',
10              'list'  => ['Home', 'Welcome']
11          ];
12
13          $activeMenu = 'dashboard';
14
15          return view('welcome', ['breadcrumb' => $breadcrumb, 'activeMenu' => $a
16      }
17  }
```

2. Kita buat file pada PWL_POS/resources/views/welcome.blade.php

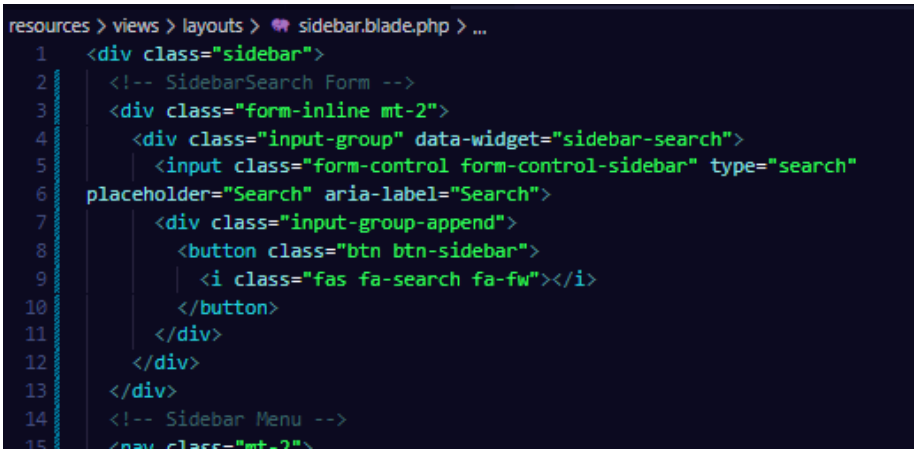
```
resources > views > welcome.blade.php > ...
1  @extends('layouts.template')
2
3  @section('content')
4
5  <div class="card">
6      <div class="card-header">
7          <h3 class="card-title">Halo, apakabar!!!</h3>
8          <div class="card-tools"></div>
9      </div>
10     <div class="card-body">
11         Selamat datang semua, ini adalah halaman utama dari
12     </div>
13 </div>
14 @endsection → c:\Laragon\www\PWL_2025\Minggu5.2\Jobsheet5.2\
```

3. Kita modifikasi file PWL_POS/resources/views/layouts/breadcrumb.blade.php



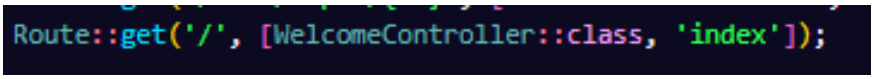
```
resources > views > layouts > breadcrumb.blade.php > section.content-header
1  <section class="content-header">
2      <div class="container-fluid">
3          <div class="row mb-2">
4              <div class="col-sm-6"><h1>{{ $breadcrumb->title }}</h1></div>
5              <div class="col-sm-6">
6                  <ol class="breadcrumb float-sm-right">
7                      @foreach($breadcrumb->list as $key => $value)
8                          @if($key == count($breadcrumb->list) - 1)
9                              <li class="breadcrumb-item active">{{ $value }}</li
10                         @else
11                             <li class="breadcrumb-item">{{ $value }}</li>
12                         @endif
13                     @endforeach
14                 </ol>
15             </div>
16         </div>
17     </div>
18 </section>
```

4. Kita modifikasi file PWL_POS/resources/views/layouts/sidebar.blade.php



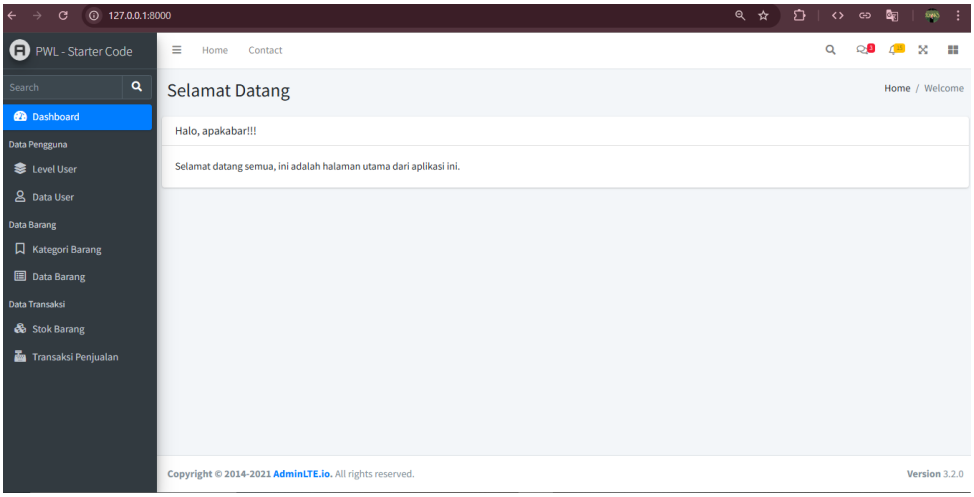
```
resources > views > layouts > sidebar.blade.php > ...
1  <div class="sidebar">
2    <!-- SidebarSearch Form -->
3    <div class="form-inline mt-2">
4      <div class="input-group" data-widget="sidebar-search">
5        <input class="form-control form-control-sidebar" type="search"
6  placeholder="Search" aria-label="Search">
7        <div class="input-group-append">
8          <button class="btn btn-sidebar">
9            <i class="fas fa-search fa-fw"></i>
10         </button>
11       </div>
12     </div>
13   </div>
14   <!-- Sidebar Menu -->
15   <nav class="mt-2">
```

5. Kita tambahkan kode berikut router web.php



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

6. Sekarang kita coba jalankan di browser dengan mengetikkan url http://localhost/PWL_POS/public



7. Jangan lupa commit dan push ke github PWL_POS kalian

C. 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

```
PS C:\laragon\www\PWL_2025\Minggu5.2\Jobsheet5.2> composer require yajra/laravel-
datatables:^10.0
The "10.0" constraint for "yajra/laravel-datatables" appears too strict and will
likely not match what you want. See https://getcomposer.org/constraints
./composer.json has been updated
Running composer update yajra/laravel-datatables
Loading composer repositories with package information
```

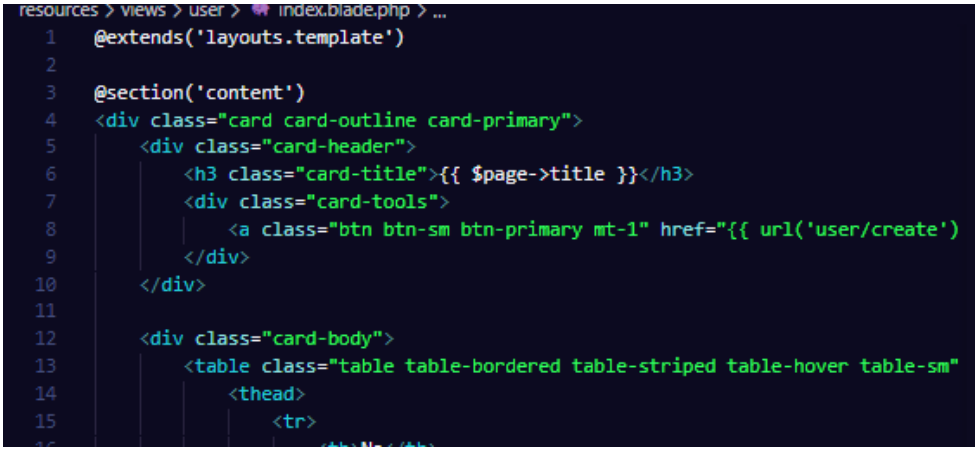
3. Kita modifikasi route web.php untuk proses CRUD user

```php
<?php

use App\Http\Controllers\UserController;
use App\Http\Controllers\WelcomeController;

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

Route::group(['prefix' => 'user'], function () {
    Route::get('/', [UserController::class, 'index']);          // menampil
    Route::post('/list', [UserController::class, 'list']);      // menampil
    Route::get('/create', [UserController::class, 'create']);   // menampi
    Route::post('/', [UserController::class, 'store']);         // menyimpa
    Route::get('/{id}', [UserController::class, 'show']);       // menampil
    Route::get('/{id}/edit', [UserController::class, 'edit']);  // menam
    Route::put('/{id}', [UserController::class, 'update']);     // menyimpa
    Route::delete('/{id}', [UserController::class, 'destroy']); // meng
});
```

4. Kita buat atau modifikasi penuh untuk UserController.php. Kita buat fungsi index() untuk menampilkan halaman awal user

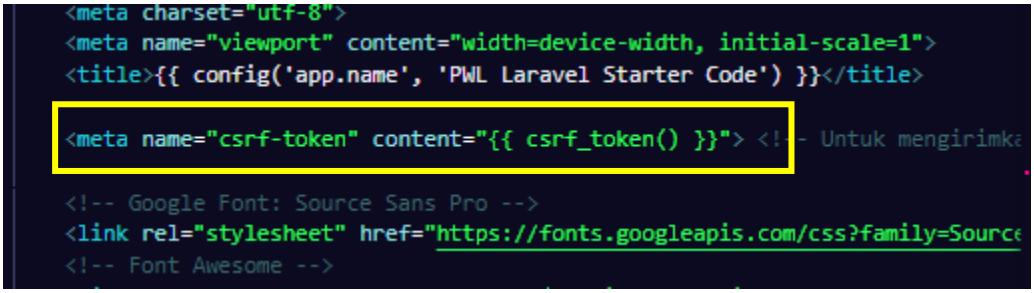
```php
<?php
namespace App\Http\Controllers;

use App\Models\LevelModel;
use App\Models\UserModel;
use Illuminate\Http\Request;
use Yajra\DataTables\Facades\DataTables;

class UserController extends Controller
{
    // Menampilkan halaman awal user
    public function index()
    {
        $breadcrumb = (object) [
            'title' => 'Daftar User',
            'list'  => ['Home', 'User']
        ];

        $page = (object) [
            'title' => 'Daftar user yang terdaftar dalam sistem'
        ];

        $activeMenu = 'user'; // set menu yang sedang aktif

        return view('user.index', ['breadcrumb' => $breadcrumb, 'page' => $page
    }
}
```

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



```blade
@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')
        </div>
    </div>

    <div class="card-body">
        <table class="table table-bordered table-striped table-hover table-sm"
            <thead>
                <tr>
```

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



```blade
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>{{ config('app.name', 'PWL Laravel Starter Code') }}</title>

<meta name="csrf-token" content="{{ csrf_token() }}"> <!- Untuk mengirimka

<!-- Google Font: Source Sans Pro -->
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source
<!-- Font Awesome -->
```

```
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/cs
    <!-- DataTables -->
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-bs4/css/
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-responsi
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-buttons/
    <!-- Theme style -->
    <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css')
    @stack('css') <!-- Digunakan untuk memanggil custom css dari perintah push(
</head>
<body class="hold-transition sidebar-mini">
```

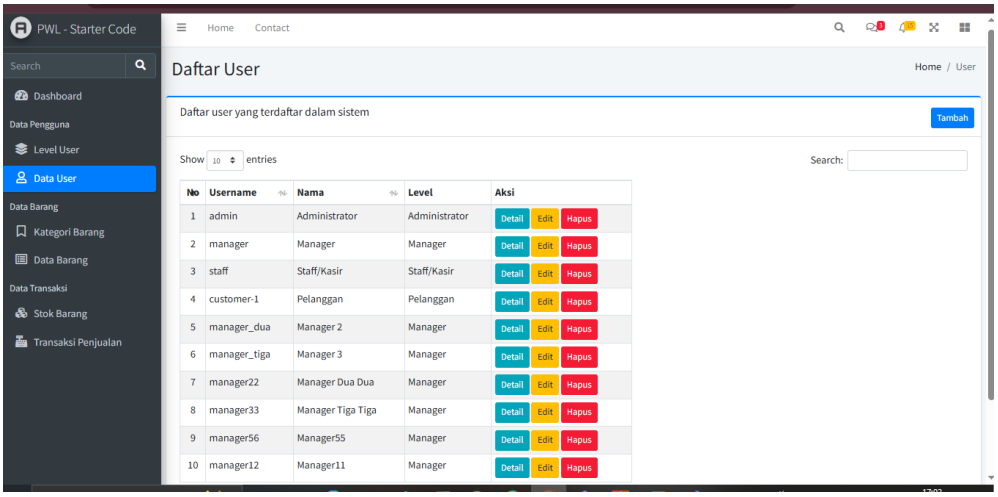
```
61  <!-- jQuery -->
62  <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
63  <!-- Bootstrap 4 -->
64  <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js')
65  <!-- DataTables & Plugins -->
66  <script src="{{ asset('adminlte/plugins/datatables/jquery.dataTables.min.js') }
67  <script src="{{ asset('adminlte/plugins/datatables-bs4/js/dataTables.bootstrap4
68  <script src="{{ asset('adminlte/plugins/datatables-responsive/js/dataTables.res
69  <script src="{{ asset('adminlte/plugins/datatables-responsive/js/responsive.boo
70  <script src="{{ asset('adminlte/plugins/datatables-buttons/js/dataTables.buttor
71  <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.bootstrap
72  <script src="{{ asset('adminlte/plugins/jszip/jszip.min.js') }}"></script>
73  <script src="{{ asset('adminlte/plugins/pdfmake/pdfmake.min.js') }}"></script>
74  <script src="{{ asset('adminlte/plugins/pdfmake/vfs_fonts.js') }}"></script>
75  <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.html5.min
76  <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.print.min
77  <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.colVis.mi
78
79  <!-- AdminLTE App -->
80  <script src="{{ asset('adminlte/dist/js/adminlte.min.js') }}"></script>
81
82  <script>
83      // Untuk mengirimkan token Laravel CSRF pada setiap request ajax
84      $.ajaxSetup({headers: {'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('
85  </script>
86
87  @stack('js') <!-- Digunakan untuk memanggil custom js dari perintah push('js')
```

7. Untuk bisa menangkap request data untuk datatable, kita buat fungsi list() pada UserController.php seperti berikut

```
class UserController extends Controller
    // 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 / nomor urut (default nama kolom: DT_Row
            ->addIndexColumn()

            // Menambahkan kolom aksi
            ->addColumn('aksi', function ($user) {
                $btn  = '<a href="'.url('/user/' . $user->user_id).'" class="bt
                $btn .= '<a href="'.url('/user/' . $user->user_id . '/edit').''
                $btn .= '<form class="d-inline-block" method="POST" action="'.u
```

```
                . csrf_field()
                . method_field('DELETE')
                . '<button type="submit" class="btn btn-danger btn-sm" onc
                . '</form>';
            return $btn;
        })

        // Memberitahu bahwa kolom aksi berisi HTML
        ->rawColumns(['aksi'])

        ->make(true);
```

8. Sekarang coba jalankan browser, dan klik menu Data User..!!! perhatikan dan amati apa yang terjadi.



9. Selanjutnya kita modifikasi UserController.php untuk form tambah data user

```
    // Menampilkan halaman form tambah user
    public function create()
    {
        $breadcrumb = (object) [
            'title' => 'Tambah User',
            'list'  => ['Home', 'User', 'Tambah']
        ];

        $page = (object) [
            'title' => 'Tambah user baru'
        ];

        $level = LevelModel::all(); // Ambil data level untuk ditampilkan di
        $activeMenu = 'user'; // Set menu yang sedang aktif

        return view('user.create', [
            'breadcrumb' => $breadcrumb,
            'page'       => $page,
            'level'      => $level,
            'activeMenu' => $activeMenu
        ]);
    }
}
```

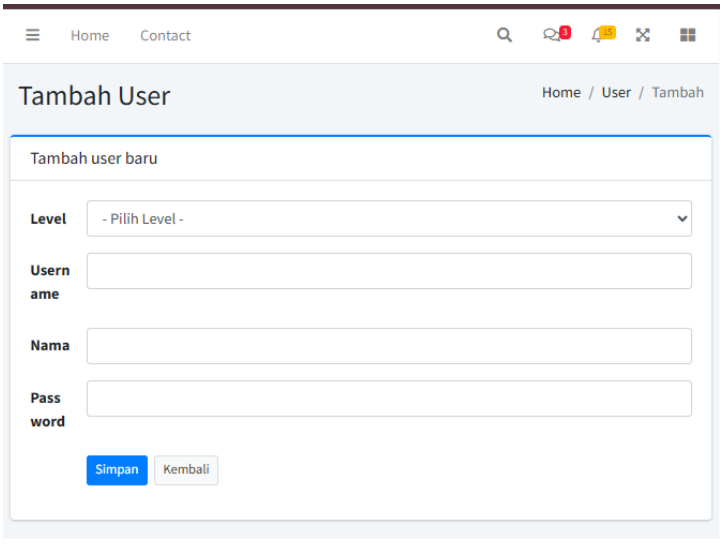
10. Sekarang kita buat form untuk menambah data, kita buat file PWL/resources/views/user/create.blade.php

```
resources > views > user > create.blade.php > ...
1   @extends('layouts.template')
2
3   @section('content')
4       <div class="card card-outline card-primary">
5           <div class="card-header">
6               <h3 class="card-title">{{ $page->title }}</h3>
7               <div class="card-tools"></div>
8           </div>
9
10          <div class="card-body">
11              <form method="POST" action="{{ url('user') }}" class="form-horizor
12                  @csrf
13
```

11. Kemudian untuk bisa menng-handle data yang akan disimpan ke database, kita buat fungsi store() di UserController.php

```
// Menyimpan data user baru
public function store(Request $request)
{
    $request->validate([
        // username harus diisi, berupa string, minimal 3 karakter, dan ber
        'username' => 'required|string|min:3|unique:m_user,username',
        // nama harus diisi, berupa string, dan maksimal 100 karakter
        'nama'     => 'required|string|max:100',
        // password harus diisi dan minimal 5 karakter
        'password' => 'required|min:5',
        // level_id harus diisi dan berupa angka
        'level_id' => 'required|integer'
    ]);

    UserModel::create([
        'username' => $request->username,
        'nama'     => $request->nama,
        'password' => bcrypt($request->password), // password dienkripsi se
        'level_id' => $request->level_id
    ]);

    return redirect('/user')->with('success', 'Data user berhasil disimpan'
```

12. Sekarang coba kalian buka form tambah data user dengan klik tombol tambah. Amati dan pelajari..!!!



13. Selanjutnya, kita masuk pada bagian menampilkan detail data user (klik tombol detail) pada halaman user. Route yang bertugas untuk menangkap request detail adalah

```php
Route::group(['prefix' => 'user'], function () {
    Route::get('/', [UserController::class, 'index']);       // menampilkan halaman awal user
    Route::post('/list', [UserController::class, 'list']);   // menampilkan data user dalam ber
    Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tamba
    Route::post('/', [UserController::class, 'store']);      // menyimpan data user baru
    Route::get('/{id}', [UserController::class, 'show']);    // menampilkan detail user
    Route::get('/{id}/edit', [UserController::class, 'edit']); // menampilkan halaman form edit
    Route::put('/{id}', [UserController::class, 'update']);  // menyimpan perubahan data user
    Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user
});
```

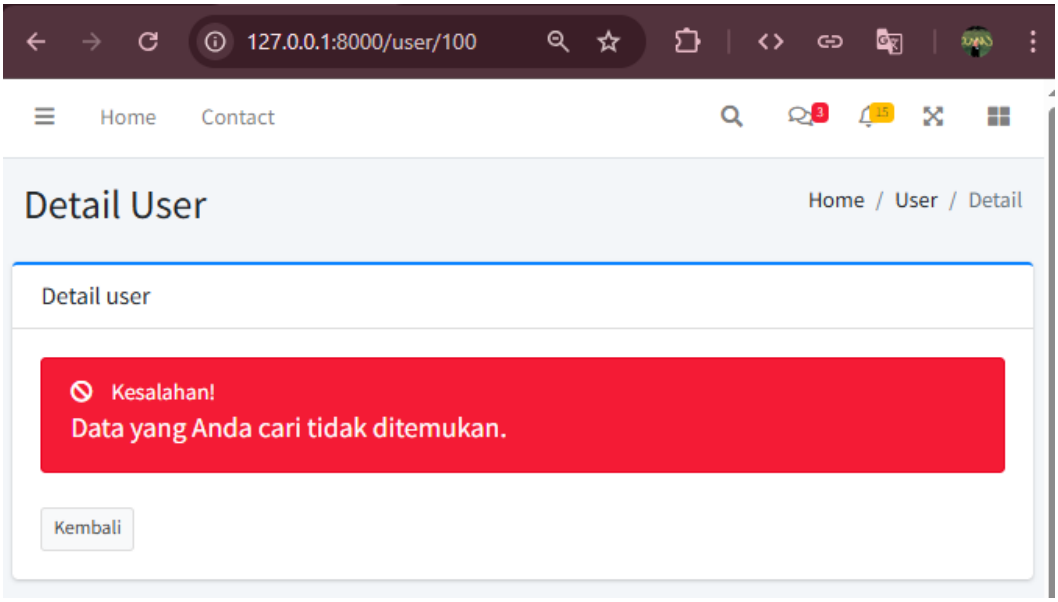
14. Jadi kita buat/modifikasi fungsi show() pada UserController.php seperti berikut

```php
100     // Menampilkan detail user
101     public function show(string $id)
102     {
103         $user = UserModel::with('level')->find($id);
104
105         $breadcrumb = (object) [
106             'title' => 'Detail User',
107             'list'  => ['Home', 'User', 'Detail']
108         ];
109
110         $page = (object) [
111             'title' => 'Detail user'
112         ];
113
114         $activeMenu = 'user'; // set menu yang sedang aktif
115
116         return view('user.show', [
117             'breadcrumb' => $breadcrumb,
118             'page'       => $page,
119             'user'       => $user,
120             'activeMenu' => $activeMenu
121         ]);
122     }
```

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

```
resources > views > user > show.blade.php > ...
1   @extends('layouts.template')
2
3   @section('content')
4       <div class="card card-outline card-primary">
5           <div class="card-header">
6               <h3 class="card-title">{{ $page->title }}</h3>
7               <div class="card-tools"></div>
8           </div>
9           <div class="card-body">
10              @empty($user)
11                  <div class="alert alert-danger alert-dismissible">
12                      <div class="icon fas fa-ban"></div> Kesalahan!
13                      <h5>Data yang Anda cari tidak ditemukan.</h5>
14                  </div>
15              @else
16                  <table class="table table-bordered table-striped table-hover ta
17                      <tr>
18                          <th>ID</th>
19                          <td>{{ $user->user_id }}</td>
```

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



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

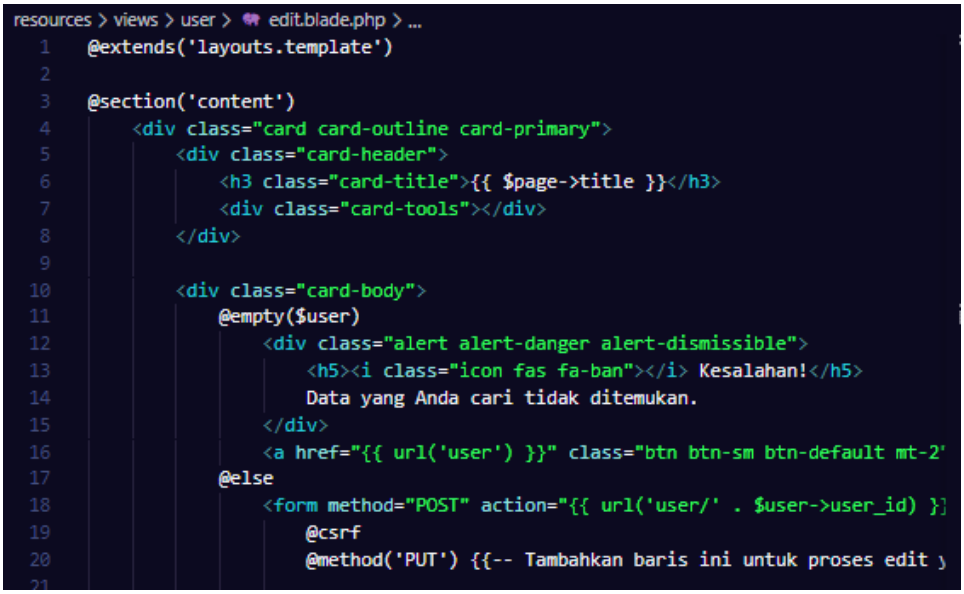
```
Route::get('/{id}/edit', [UserController::class, 'edit']); // menampilkan h
Route::put('/{id}', [UserController::class, 'update']);  // menyimpan perut
```

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

```
// Menampilkan halaman form edit user
public function edit(string $id)
{
    $user = UserModel::find($id);
    $level = LevelModel::all();

    $breadcrumb = (object) [
        'title' => 'Edit User',
        'list'  => ['Home', 'User', 'Edit']
    ];

    $page = (object) [
        'title' => 'Edit user'
    ];

    $activeMenu = 'user'; // set menu yang sedang aktif

    return view('user.edit', [
        'breadcrumb' => $breadcrumb,
        'page' => $page,
        'user' => $user,
        'level' => $level,
        'activeMenu' => $activeMenu
    ]);
}
```

```php
// Menyimpan perubahan data user
public function update(Request $request, string $id)
{
    $request->validate([
        // username harus diisi, berupa string, minimal 3 karakter,
        // dan bernilai unik di tabel m_user kolom username kecuali untuk user
        'username' => 'required|string|min:3|unique:m_user,username,' . $id .
        'nama' => 'required|string|max:100', // nama harus diisi, berupa stri
        'password' => 'nullable|min:5', // password bisa diisi (minimal 5 kar
        'level_id' => 'required|integer' // level_id harus diisi dan berupa a
    ]);

    UserModel::find($id)->update([
        'username' => $request->username,
        'nama' => $request->nama,
        'password' => $request->password ? bcrypt($request->password) : UserM
        'level_id' => $request->level_id
    ]);

    return redirect('/user')->with('success', 'Data user berhasil diubah');
}
```

19. Selanjutnya, kita buat view untuk melakukan proses edit data user di PWL/resources/views/user/edit.blade.php



20. Sekarang kalian coba untuk mengedit data user di browser, amati, pahami, dan laporkan!
21. 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']);

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

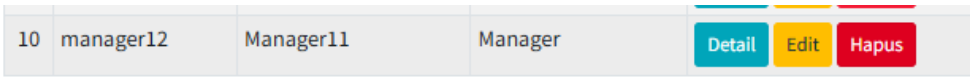
22. Jadi kita buat fungsi destroy() pada UserController.php

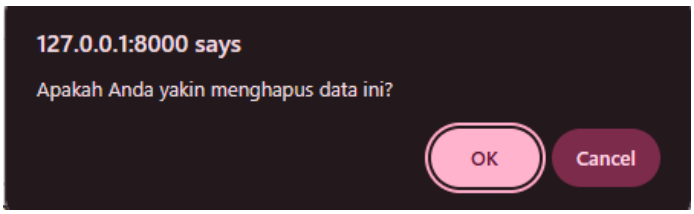
```php
// Menghapus data user
public function destroy(string $id)
{
    $check = UserModel::find($id);
    if (!$check) { // untuk mengecek apakah data user dengan id yang dimaksud
        return redirect('/user')->with('error', 'Data user tidak ditemukan');
    }

    try {
        UserModel::destroy($id); // Hapus data level

        return redirect('/user')->with('success', 'Data user berhasil dihapus');
    } catch (\Illuminate\Database\QueryException $e) {
        // Jika terjadi error ketika menghapus data, redirect kembali ke halaman
        return redirect('/user')->with('error', 'Data user gagal dihapus karena
    }
}
```

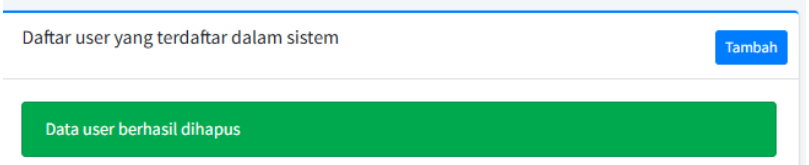
23. Selanjutnya kita modifikasi file PWL/resources/views/user/index.blade.php untuk menambahkan tampilan jika ada pesan error

```blade
11      <div class="card-body">
12      @if (session('success'))
13          <div class="alert alert-success">{{ session('success') }}</div>
14      @endif
15
16      @if (session('error'))
17          <div class="alert alert-danger">{{ session('error') }}</div>
18      @endif
19
20      <table class="table table-bordered table-striped table-hover table-sm" id='
21          <thead>
22              <tr>
23                  <th>ID</th>
24                  <th>Username</th>
25                  <th>Nama</th>
26                  <th>Level Pengguna</th>
27                  <th>Aksi</th>
28              </tr>
29          </thead>
30      </table>
31  </div>
```

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







25. Selamat, kalian sudah membuat Laravel Starter Code untuk proses CRUD dengan menggunakan template AdminLTE dan plugin jQuery Datatables.
26. Jangan lupa commit dan push ke github PWL_POS kalian

D. PRAKTIKUM 4 : Implementasi Filtering Datatables

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

```
$level = LevelModel::all(); // Ambil data level untuk ditampilkan di fo
```

```
ge, 'level' => $level, 'activ
```

2. Kemudian kita modifikasi view untuk menampilkan data filtering pada PWL/resources/views/user/index.blade.php

```
<div class="row">
<div class="col-md-12">
    <div class="form-group row">
        <label class="col-1 control-label col-form-label">Filter:</label>
        <div class="col-3">
            <select class="form-control" id="level_id" name="level_id" requ
                <option value="">-- Semua --</option>
                @foreach($level as $item)
                    <option value="{{ $item->level_id }}">{{ $item->level_n
                @endforeach
            </select>
            <small class="form-text text-muted">Level Pengguna</small>
        </div>
    </div>
</div>
```

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

```
"data": function (d) {
d.level_id = $('#level_id').val();
}
```

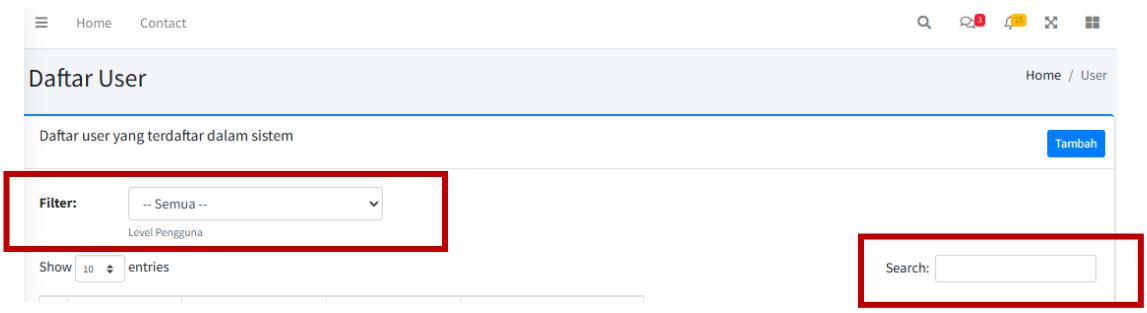
4. Kemudian kita edit pada bagian akhir script @push('js') untuk menambahkan listener jika data filtering dipilih

```
$('#level_id').on('change', function() {
    dataUser.ajax.reload();
});
```

5. Tahapan akhir adalah memodifikasi fungsi list() pada UserController.php yang digunakan untuk menampilkan data pada datatable

```php
// Filter data user berdasarkan level_id
if ($request->level_id) {
    $users->where('level_id', $request->level_id);
}
```

6. Bagian akhir adalah kita coba jalankan di browser dengan akses menu user, maka akan tampil seperti berikut

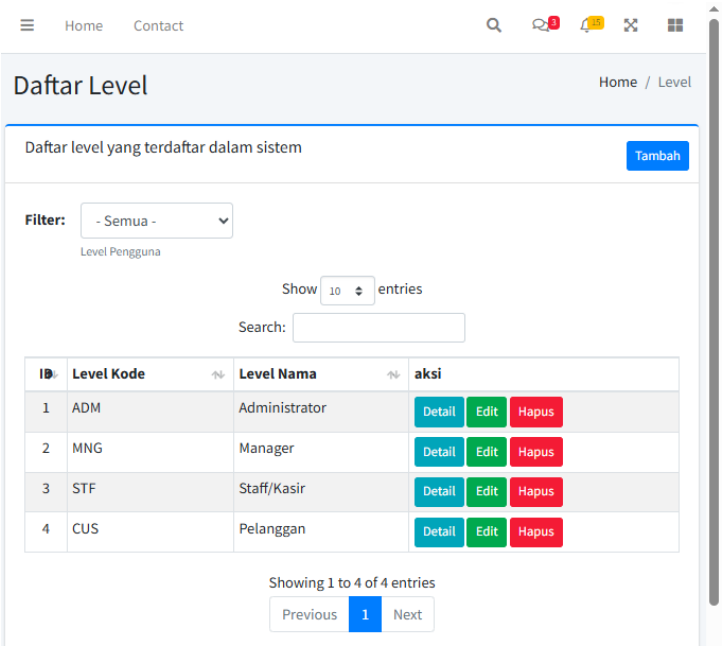


7. Selamat, sekarang Laravel Starter Code sudah ada filtering dan searching data. Starter Code sudah bisa digunakan dalam membangun sebuah sistem berbasis website.
8. Jangan lupa commit dan push ke github PWL_POS kalian

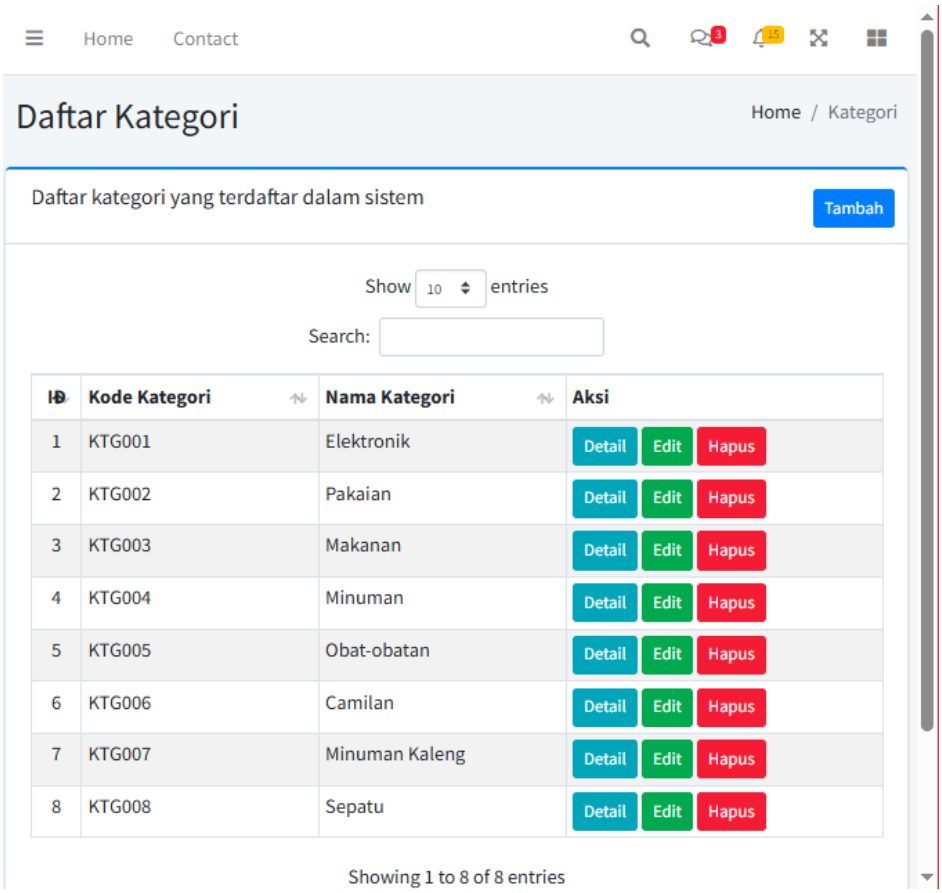
E. TUGAS

1. Implementasikan web layout dan datatables, pada menu berikut ini

   ✓ Tabel m_level



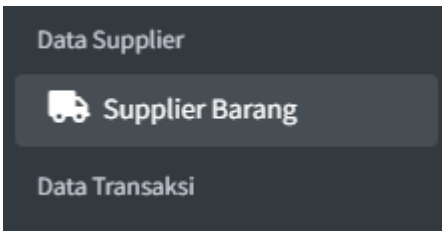
✓ Tabel m_kategori

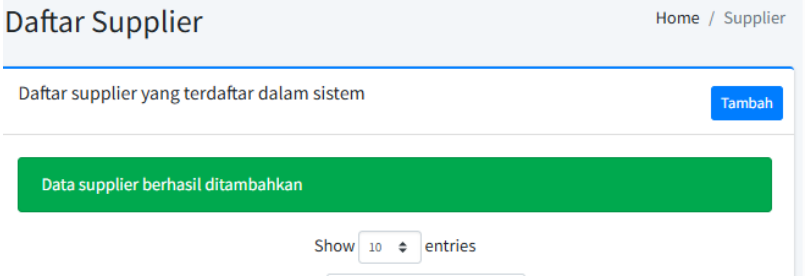


✓ Tabel m_supplier

```
PS C:\laragon\www\PWL_2025\Minggu5.2\Jobsheet5.2> php artisan make:migration crea
te_m_supplier_table.php

  INFO  Migration [C:\laragon\www\PWL_2025\Minggu5.2\Jobsheet5.2\database\migrat
ions/2025_03_26_144400_create_m_supplier_table.php.php] created successfully.
```

```
database > seeders > SupplierSeeder.php > SupplierSeeder
  1   <?php
```





ID	Kode Supplier	Nama Supplier	Alamat Supplier	Aksi
1	SP1	Toko Pakaian Indah	JL. Maju Jaya Sentosa 58, Jakarta	Detail Edit Hapus
2	SP2	Gadget University	JL. Gajah Mada No. 88 , Surabaya	Detail Edit Hapus
3	SP3	Grosir SuperMalang	JL. Komplek Ruko WAH No. 21 , Malang	Detail Edit Hapus

Showing 1 to 3 of 3 entries

✓ Tabel m_barang