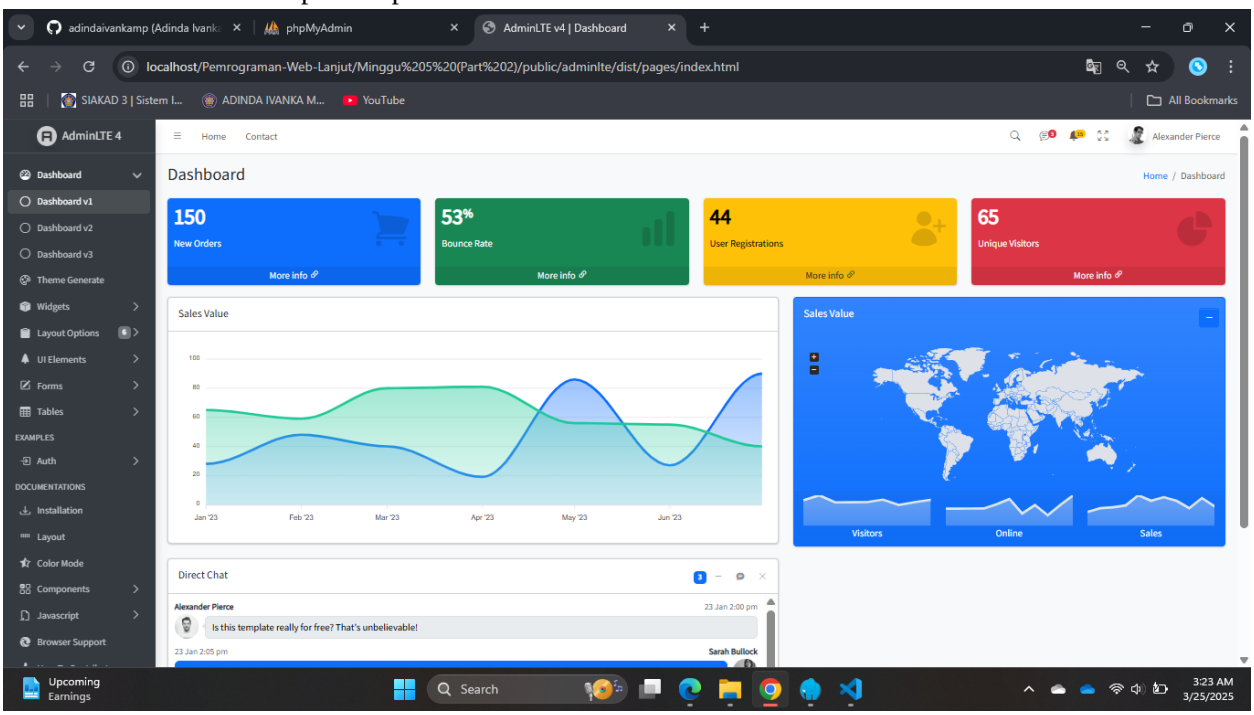
Nama : Adinda Ivanka Maysanda Putri
Kelas : SIB 2B
NIM : 2341760058

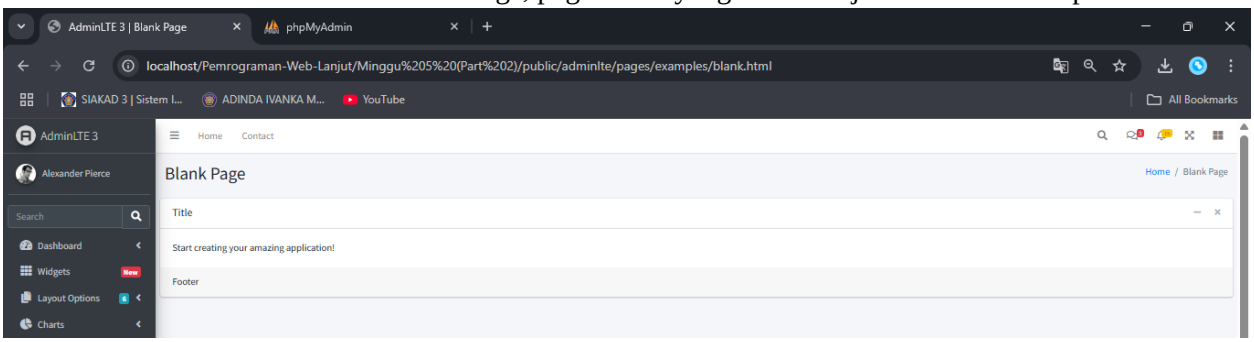
JOBSHEET 5
Blade View, Web Templating(AdminLTE), Datatables

Praktikum 1 - Layouting AdminLTE:

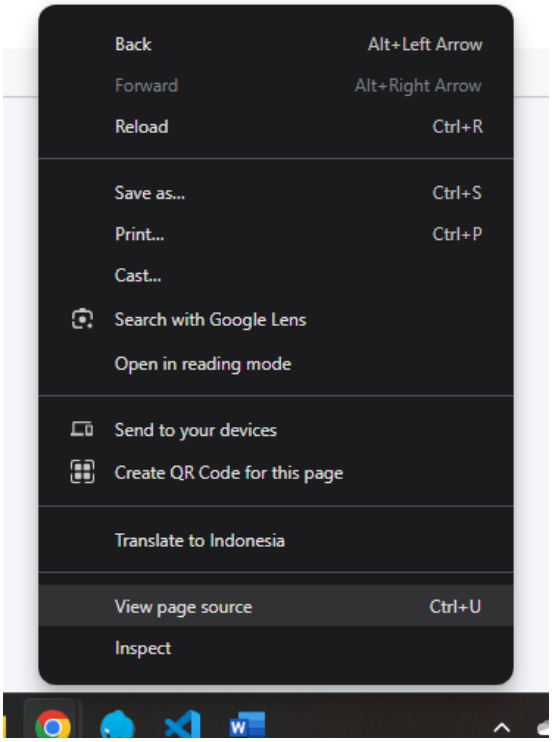
1. Kita download AdminLTE v3.2.0 yang rilis pada 8 Feb 2022
2. Seleteh kita berhasil download, kita ekstrak file yang sudah di download ke folder project PWL_POS/public, kemudian kita rename folder cukup menjadi adminlte
3. Selanjutnya kita buka di browser dengan alamat http://localhost/PWL_POS/public/adminlte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



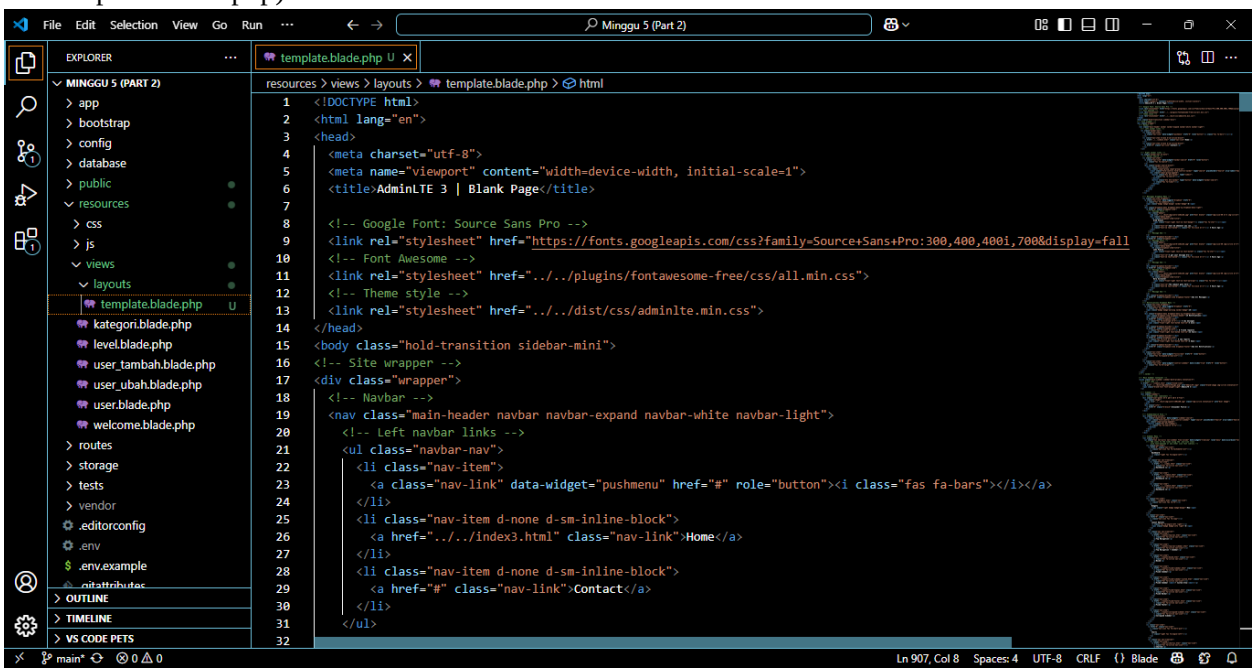
4. Kita klik menu Extras > Blank Page, page inilah yang akan menjadi dasar web template



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

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

  <!-- Google Font: Source Sans Pro -->
  <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallbac
k">
  <!-- Font Awesome -->
  <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
  <!-- Theme style -->
  <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
```

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

```
</head>
<body class="hold-transition sidebar-mini">
<!-- Site wrapper -->
<div class="wrapper">
  <!-- Navbar -->
  @include('layouts.header')
  {{-- /.navbar --}}

  {{-- Main Sidebar Container --}}
  <aside class="main-sidebar sidebar-dark-primary elevation-4">
    {{-- Brand Logo --}}
    <a href="../../index3.html" class="brand-link">
      <img src="../../dist/img/AdminLTELogo.png" alt="AdminLTE Logo" class="brand-image img-
circle elevation-3" style="opacity: .8">
      <span class="brand-text font-weight-light">AdminLTE 3</span>
    </a>
```

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

```
  {{-- Main Sidebar Container --}}
  <aside class="main-sidebar sidebar-dark-primary elevation-4">
    {{-- Brand Logo --}}
    <a href="{{url('/')}}" class="brand-link">
      <img src="{{('adminlte/dist/img/AdminLTELogo.png')}}" alt="AdminLTE Logo"
class="brand-image img-circle elevation-3" style="opacity: .8">
      <span class="brand-text font-weight-light">PWL - Stater Code</span>
```

```
    </a>

{{-- Sidebar --}}
<div class="sidebar">
```

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

```
<div class="sidebar">

    <div class="user-panel mt-3 pb-3 mb-3 d-flex">
        <div class="image">
            <img src="../../dist/img/user2-160x160.jpg" class="img-circle elevation-2" alt="User Image">
        </div>
        <div class="info">
            <a href="#" class="d-block">Alexander Pierce</a>
        </div>
    </div>

    <div class="form-inline">
        <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>
</div>
```

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

```
<footer class="main-footer">
    <div class="float-right d-none d-sm-block">
        <b>Version</b> 3.2.0
    </div>
    <strong>Copyright &copy; 2014-2021 <a href="https://adminlte.io">AdminLTE.io</a>.</strong>
All rights reserved.
</footer>

<aside class="control-sidebar control-sidebar-dark">
    </aside>
```

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

```
<script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
<script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
<script src="../../dist/js/adminlte.min.js"></script>
<script src="../../dist/js/demo.js"></script>
</body>
</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 4 breadcrumb. Kita blok baris 38-52 lalu kita cut, dan PWL_POS/resource/view/layouts/breadcrumb.blade.php breadcrumb.blade.php jika paste-kan di file (buat file belum ada). Sehingga tampilan dari file template.blade.php menjadi seperti berikut \

```
<section class="content-header">
   <div class="container-fluid">
      <div class="row mb-2">
         <div class="col-sm-6">
            <h1>Blank Page</h1>
         </div>
         <div class="col-sm-6">
            <ol class="breadcrumb float-sm-right">
               <li class="breadcrumb-item"><a href="#">Home</a></li>
               <li class="breadcrumb-item active">Blank Page</li>
            </ol>
         </div>
      </div>
   </div></section>
```

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.

```
@yield('content')

</section>
</div>
```

18. Untuk content, kita akan menghapus baris 42-66 pada file template.blade.php. dan kita ganti dengan kode seperti ini @yield('content')
19. Hasil akhir pada file utama layouts/template.blade.php adalah seperti berikut

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1">
   <title>{{ config('app.name', 'Pwl Laravel Starter Code') }}</title>
```

```
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallbac
k">
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/dist/css/adminlte.min.css') }}">
</head>
<body class="hold-transition sidebar-mini">
    <div class="wrapper">

        @include('layouts.header')
        <aside class="main-sidebar sidebar-dark-primary elevation-4">
            <a href="{{ url('/') }}" class="brand-link">
                <img src="{{ asset('adminlte/plugins/dist/img/AdminLTELogo.png') }}" alt="AdminLTE
Logo" class="brand-image img-circle elevation-3" style="opacity: .8">
                <span class="brand-text font-weight-light">Starter Code</span>
            </a>

            @include('layouts.sidebar')
            </aside>

        <div class="content-wrapper">
            @include('layouts.breadcrumb')
            <section class="content">
                @yield('content')
            </section>
            </div>
        @include('layouts.footer')
        </div>
    </body>
</html>
```

20. Selamat kalian sudah selesai dalam melakukan layouting website di laravel.
21. 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
2. Kita buat file pada PWL_POS/resources/views/welcome.blade.php

```
<?php

namespace App\Http\Controllers;

class WelcomeController extends Controller
{
```

```
    public function index()
    {
        $breadcrumb = (object) [
            'title' => 'Selamat Datang',
            'list' => ['Home', 'Welcome']
        ];

        $activeMenu = 'dashboard';

        return view('welcome', ['breadcrumb' => $breadcrumb, 'activeMenu' => $activeMenu]);
    }
}
```

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

```
<section class="content-header">
    <div class="container-fluid">
        <div class="row mb-2">
            <div class="col-sm-6"><h1>{{ $breadcrumb->title }}</h1></div>
            <div class="col-sm-6">
                <ol class="breadcrumb float-sm-right">
                    @foreach($breadcrumb->list as $key => $value)
                        @if($key == count($breadcrumb->list) - 1)
                            <li class="breadcrumb-item active">{{ $value }}</li>
                        @else
                            <li class="breadcrumb-item">{{ $value }}</li>
                        @endif
                    @endforeach
                </ol>
            </div>
        </div>
    </div>
</section>
```

4. 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>
```

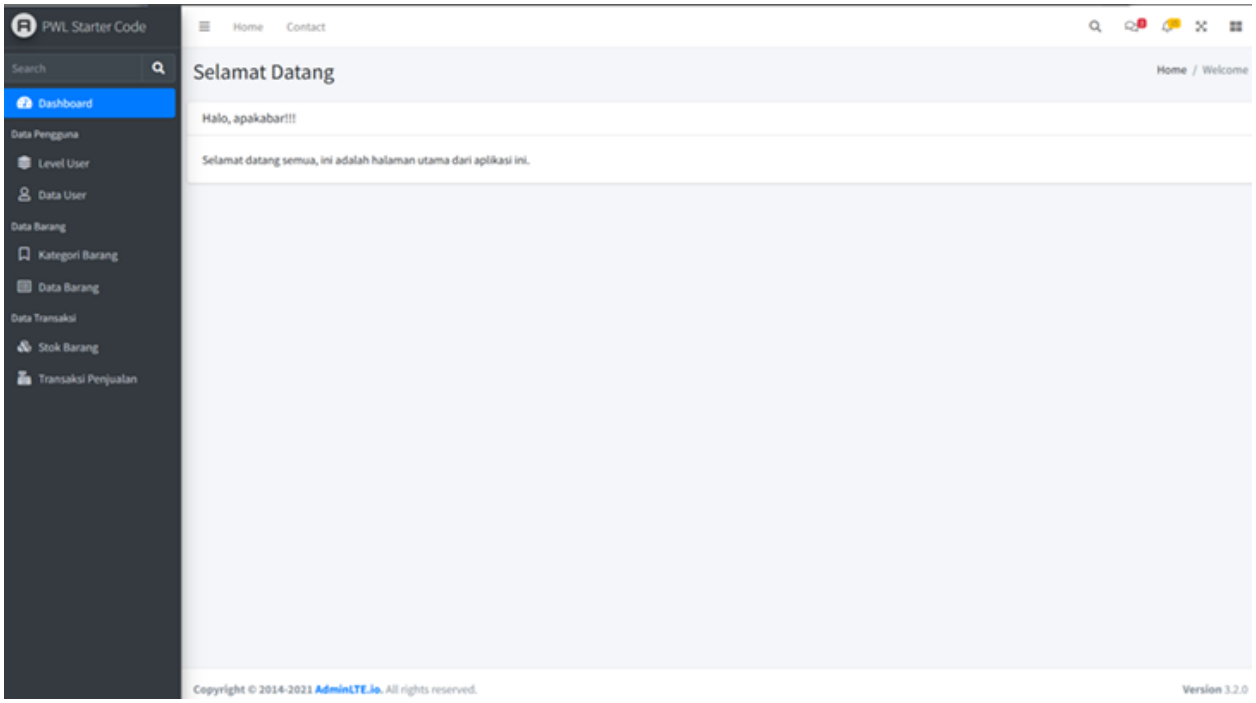
```
    </div>
    <!-- Sidebar Menu -->
    <nav class="mt-2">
      <ul class="nav nav-pills nav-sidebar flex-column" data-widget="treeview" role="menu" data-
accordion="false">
        <li class="nav-item">
          <a href="{{ url('/') }}" class="nav-link {{ ($activeMenu == 'dashboard')? 'active' : '' }} ">
            <i class="nav-icon fas fa-tachometer-alt"></i>
            <p>Dashboard</p>
          </a>
        </li>
        <li class="nav-header">Data Pengguna</li>
        <li class="nav-item">
          <a href="{{ url('/level') }}" class="nav-link {{ ($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('/supplier') }}" class="nav-link {{ ($activeMenu == 'supplier')? 'active' : '' }} ">
            <i class="nav-icon far fa-building"></i>
            <p>Data Supplier</p>
          </a>
        </li>
        <li class="nav-item">
          <a href="{{ url('/barang') }}" class="nav-link {{ ($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>
  <!-- /.sidebar-menu -->
 </div>
```

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

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\Pemrograman-Web-Lanjut\Minggu 5 (Part 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://getcompo
er.org/constraints
./composer.json has been updated
Running composer update yajra/laravel-datatables
Loading composer repositories with package information
Updating dependencies
Lock file operations: 7 installs, 0 updates, 0 removals
  - Locking league/fractal (0.20.2)
  - Locking yajra/laravel-datatables (v10.0.0)
  - Locking yajra/laravel-datatables-buttons (v10.0.9)
  - Locking yajra/laravel-datatables-editor (v1.25.4)
  - Locking yajra/laravel-datatables-fractal (v10.0.0)
  - Locking yajra/laravel-datatables-html (v10.12.0)
  - Locking yajra/laravel-datatables-oracle (v10.11.4)
les-fractal (v10.0.0): Extracting archive
  - Installing yajra/laravel-datatables-editor (v1.25.4): Extracting archive
  - Installing yajra/laravel-datatables-buttons (v10.0.9): Extracting archive
  - Installing yajra/laravel-datatables (v10.0.0): Extracting archive
8 package suggestions were added by new dependencies, use `composer suggest` to see details.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

   INFO  Discovering packages.

  laravel/sail ..... DONE
  laravel/sanctum ..... DONE
  laravel/tinker ..... DONE
```

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

```php
<?php

use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers\UserController;
use App\Http\Controllers\WelcomeController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

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

Route::group(['prefix' => 'user'], function () {
```

```
    Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user

    Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk
datatables
    Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user
    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 user
    Route::put('/{id}', [UserController::class, 'update']); // menyimpan perubahan data user
    Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user

});
```

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

```
<?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, 'activeMenu' =>
$activeMenu]);
    }
}
```

5. 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:
```

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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <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() }}"> <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallbac
k">
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-
bs4/css/dataTables.bootstrap4.min.css') }}">
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-
responsive/css/responsive.bootstrap4.min.css') }}">
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-
buttons/css/buttons.bootstrap4.min.css') }}">
    <link rel="stylesheet" href="{{ asset('adminlte/plugins/dist/css/adminlte.min.css') }}">
    @stack('css') </head>
<body class="hold-transition sidebar-mini">
    <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/datatables/jquery.dataTables.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/datatables-bs4/js/dataTables.bootstrap4.min.js')
}}"></script>
```

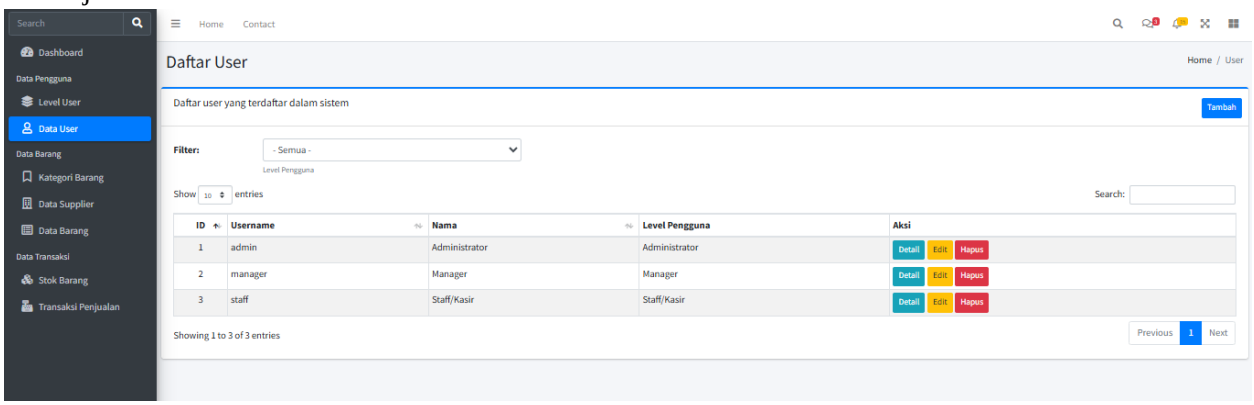
```
    <script src="{{ asset('adminlte/plugins/datatables-responsive/js/dataTables.responsive.min.js')
}}"></script>
    <script src="{{ asset('adminlte/plugins/datatables-responsive/js/responsive.bootstrap4.min.js')
}}"></script>
    <script src="{{ asset('adminlte/plugins/datatables-buttons/js/dataTables.buttons.min.js')
}}"></script>
    <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.bootstrap4.min.js')
}}"></script>
    <script src="{{ asset('adminlte/plugins/jszip/jszip.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/pdfmake/pdfmake.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/pdfmake/vfs_fonts.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.html5.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.print.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.colvis.min.js') }}"></script>
    <script src="{{ asset('adminlte/plugins/dist/js/adminlte.min.js') }}"></script>
    <script>
        // Untuk mengirimkan token Laravel CSRF pada setiap request ajax
        $.ajaxSetup({ headers: { 'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content') } });
    </script>
    @stack('js') </body>
</html>
```

7. 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)
. '">';
            $btn .= csrf_field();
            $btn .= method_field('DELETE');
            $btn .= '<button type="submit" class="btn btn-danger btn-sm" onclick="return
confirm(\'Apakah Anda yakin menghapus data ini?\')">Hapus</button>';
            return $btn;
        })
```

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

    $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

```
@extends('layouts.template')

@section('content')
```

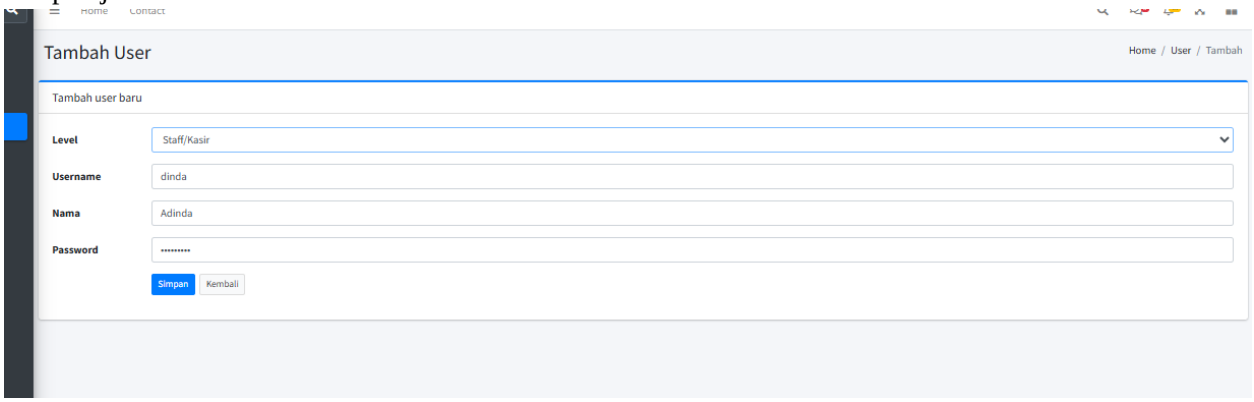
```
    <div class="card card-outline card-primary">
        <div class="card-header">
            <h3 class="card-title">{{ $page->title }}</h3>
            <div class="card-tools"></div>
        </div>
        <div class="card-body">
            <form method="POST" action="{{ url('user') }}" class="form-horizontal">
                @csrf
                <div class="form-group row">
                    <label class="col-1 control-label col-form-label">Level</label>
                    <div class="col-11">
                        <select class="form-control" id="level_id" name="level_id" required>
                            <option value="">Pilih Level</option>
                            @foreach($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>
            </form>
        </div>
    </div>
@endsection
```

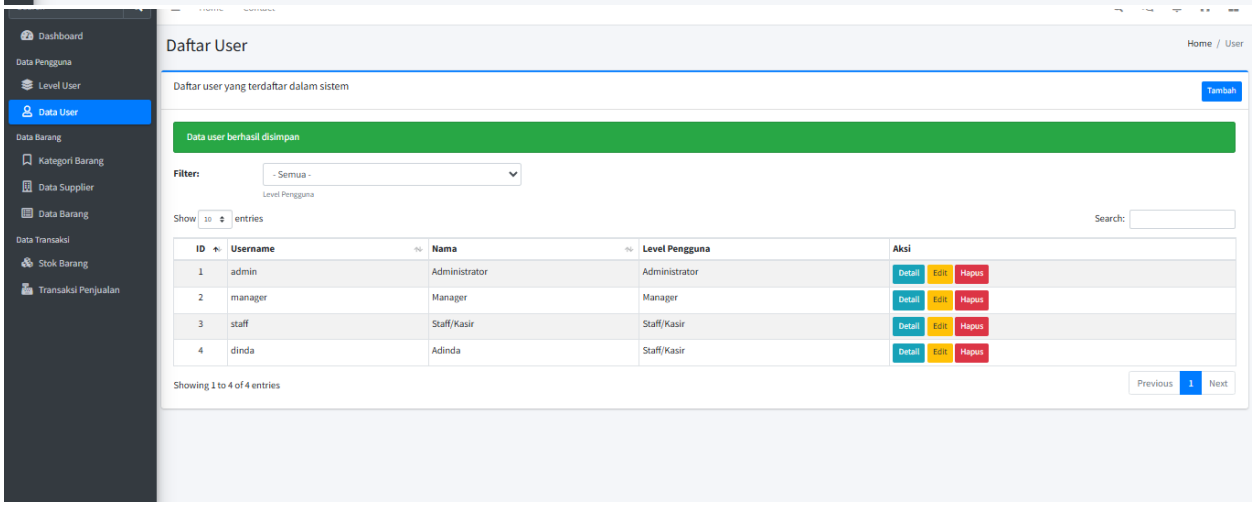
11. Kemudian untuk bisa menng-handle data yang akan disimpan ke database, kita buat fungsi store() di UserController.php

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

    UserModel::create([
        'username' => $request->username,
```

```
        'nama' => $request->nama,
        'password' => bcrypt($request->password), // password dienkripsi sebelum disimpan
        '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) pada halaman user. Route yang bertugas untuk menangkap request detail adalah

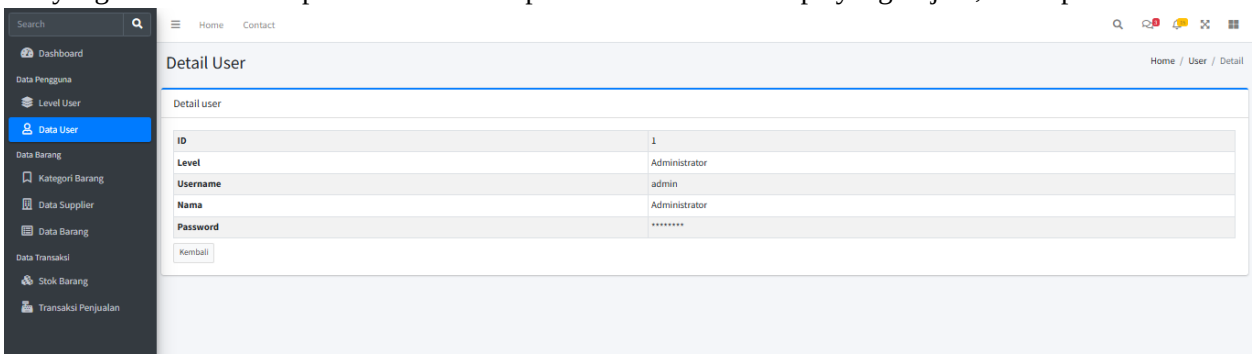
```
Route::group(['prefix' => 'user'], function () {
    Route::get('/', [UserController::class, 'index']);          // menampilkan halaman awal user
    Route::post('/list', [UserController::class, 'list']);      // menampilkan data user dalam bentuk json untuk datatables
    Route::get('/create', [UserController::class, 'create']);   // menampilkan halaman form tambah user
    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 user
    Route::put('/{id}', [UserController::class, 'update']);     // menyimpan perubahan data user
    Route::delete('/{id}', [UserController::class, 'destroy']); // menghapus data user
```

```
    Route::post('/data', [UserController::class, 'getUsers'])->name('user.data');
});
```

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

```
  // Menampilkan detail user
    public function show(string $id)
    {
        $user = UserModel::with('level')->find($id);

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

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

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

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

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

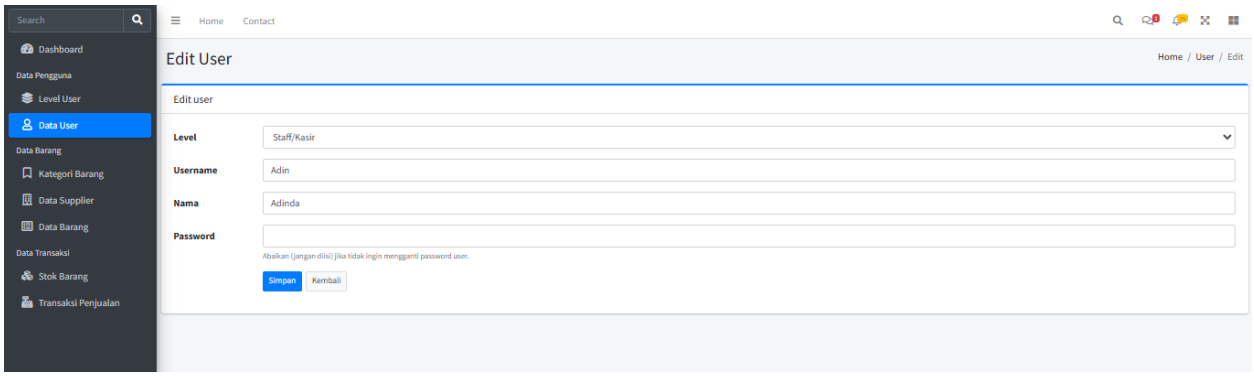


17. Selanjutnya, kita masuk pada bagian untuk memodifikasi data user. Route yang bertugas untuk menangkap request edit adalah
18. Jadi kita buat fungsi edit() dan update() pada UserController.php

```
    // 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]);
    }

    // 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 dengan id yang sedang diedit
            'username' => 'required|string|min:3|unique:m_user,username,' . $id . ',user_id',
            'nama' => 'required|string|max:100', // nama harus diisi, berupa string, dan maksimal 100 karakter
            'password' => 'nullable|min:5', // password bisa diisi (minimal 5 karakter) dan bisa tidak diisi
            'level_id' => 'required|integer' // level_id harus diisi dan berupa angka
        ]);

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

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

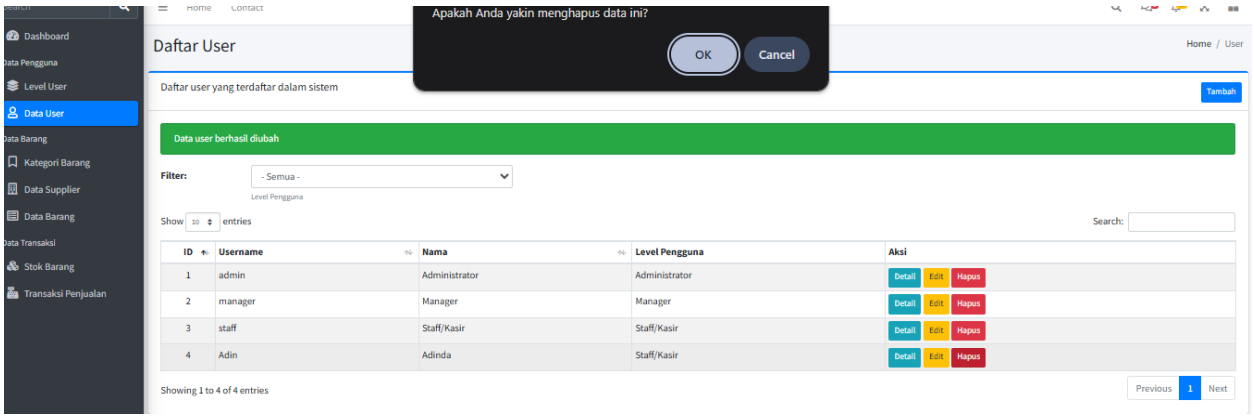
19. Selanjutnya, kita buat view untuk melakukan proses edit data user di PWL/resources/views/user/edit.blade.php
20. Sekarang kalian coba untuk mengedit data user di browser, amati, pahami, dan laporkan!



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

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

      try {
          UserModel::destroy($id); // Hapus data user
          return redirect('/user')->with('success', 'Data user berhasil dihapus');
      } catch (\Illuminate\Database\QueryException $e) {
          // Jika terjadi error ketika menghapus data, redirect kembali ke halaman dengan membawa
pesan error
          return redirect('/user')->with('error', 'Data user gagal dihapus karena masih terdapat tabel lain
yang terkait dengan data ini');
      }
  }
```

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

Praktikum 4 – Implementasi Filtering Datatables:

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

```
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

    $level = LevelModel::all(); // ambil data Level untuk filter Level

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

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" required>
                <option value="">- Semua -</option>
                @foreach($level as $item)
                    <option value="{{ $item->level_id }}">{{ $item->level_nama }}</option>
                @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

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

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

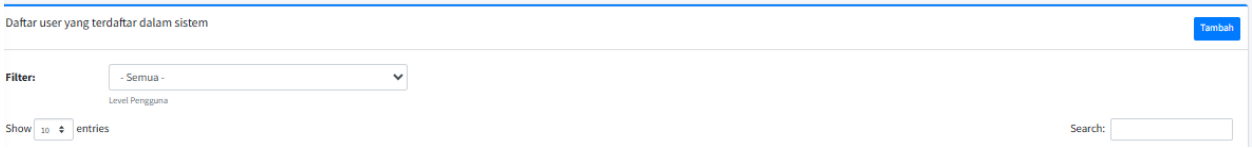
```
$('#level_id').on('change', function() {
    dataUser.ajax.reload();
});
```

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

```
// Ambil data user dalam bentuk json untuk datatables
public function list(Request $request)
{
    $users = UserModel::select('user_id', 'username', 'nama', 'level_id')
        ->with('level');

    // Filter data user berdasarkan level_id
    if ($request->level_id) {
```

```
        $users->where('level_id', $request->level_id);
    }

    return DataTables::of($users)
        // menambahkan kolom index / no urut (default nama kolom: DT_RowIndex)
```

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

TUGAS

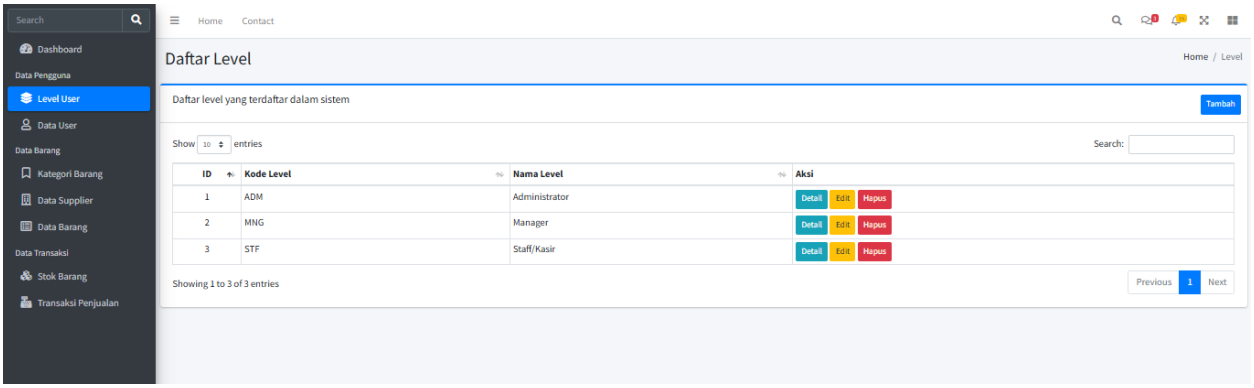
Implementasikan web layout dan datatables, pada menu berikut ini

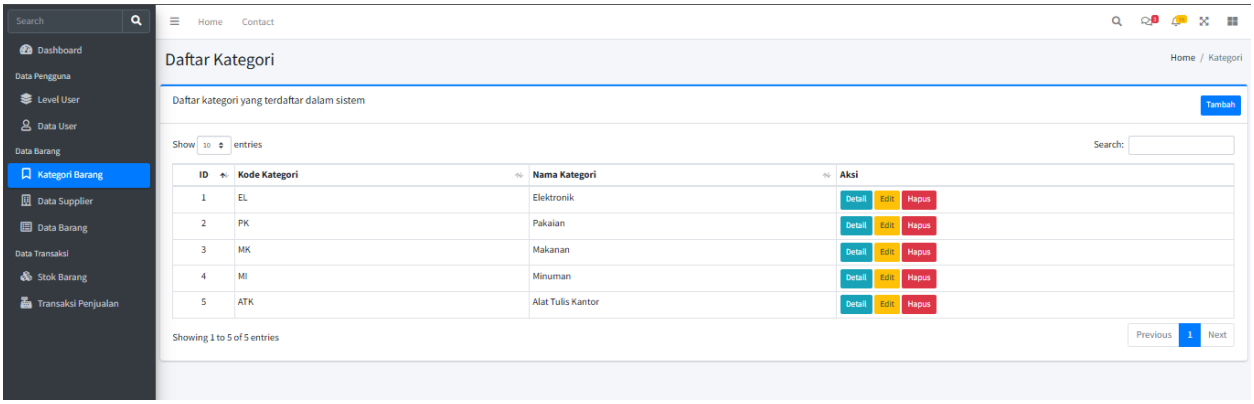
✓Tabel m_level

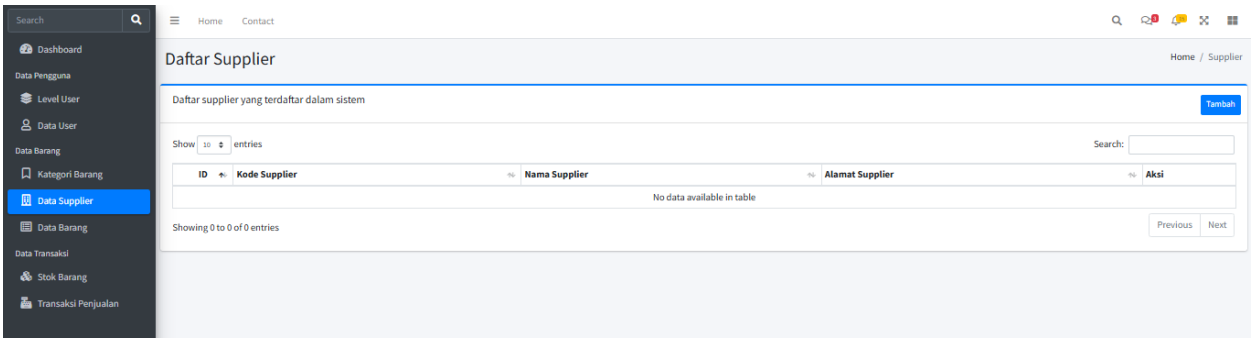
✓ Tabel m_kategori

✓ Tabel m_supplier

✓ Tabel m_barang







(tidak dapat tersambung ke database)