



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 pelajari*

#### 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:

##### 1. 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*.

##### 2. 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.

### 3. Sintaks Dasar Blade

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

- **Menampilkan Variabel:**

```
{{ $name }}
```

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

- **Struktur Kontrol:**

```
@if($condition)
    <p>Condition is true</p>
@endif
```

```
@foreach($users as $user)
    <li>{{ $user->name }}</li>
@endforeach
```

- **Komentar**

```
{{-- Ini adalah komentar yang tidak akan terlihat dalam output HTML --}}
```





## B. Web Template

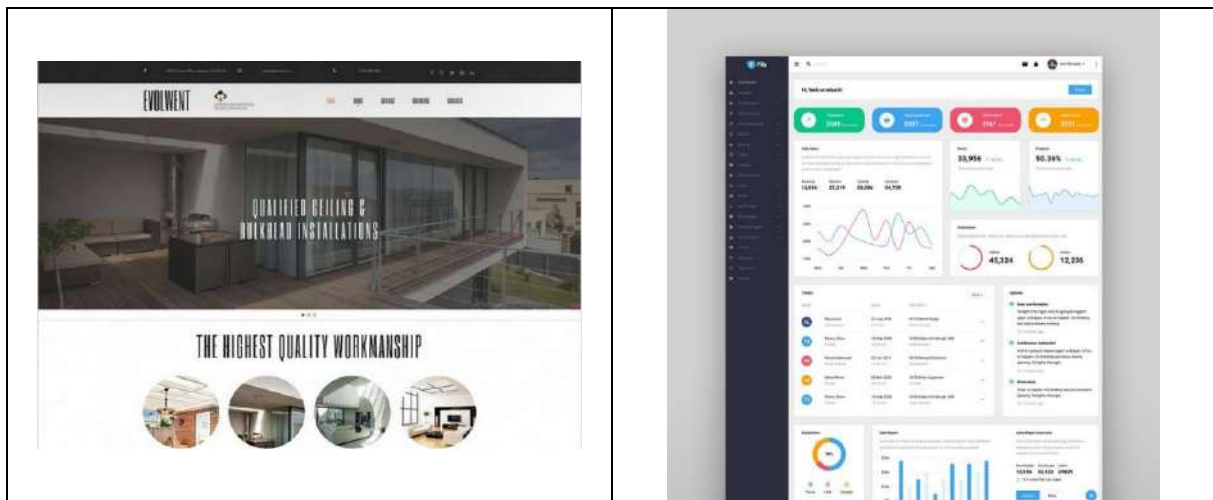
Merupakan suatu file yang sudah memiliki desain tertentu sehingga, kita tinggal memodifikasi bagian-bagian tertentu saja untuk mendapatkan hasil yang sudah bagus. Dengan menggunakan template yang sesuai dengan kebutuhan, kita bisa mengurangi waktu pembuatan suatu website secara drastis.

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 bootstrap yang merupakan framework CSS yang paling banyak digunakan di kalangan web desainer

Template AdminLTE dapat di download di <https://adminlte.io>

### Contoh Web template

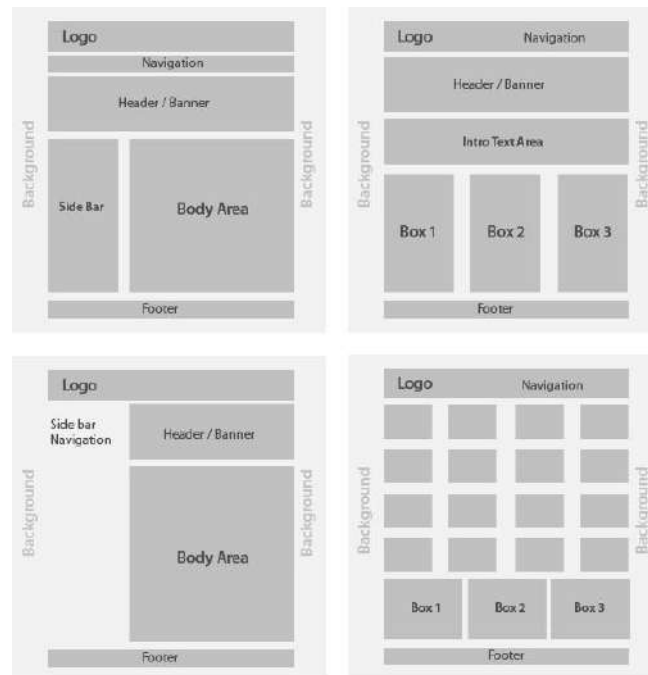
Frontend Template	Backend Template
	



## 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.

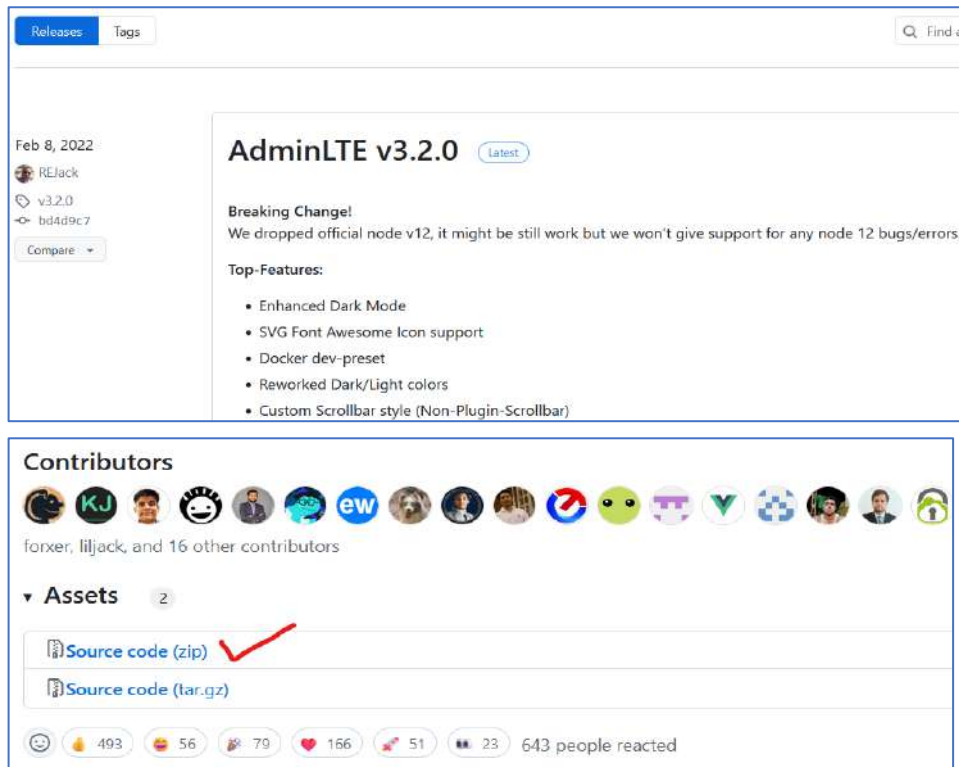
Pada layouting kali ini, kita membutuhkan **Blade Template Engine** dari Laravel untuk memecah web menjadi beberapa elemen. Komponen blade yang kita butuhkan untuk proses layouting pada pertemuan kali ini adalah

- `@include()`
- `@extend()`
- `@section()`
- `@push()`
- `@yield()`
- `@stack()`

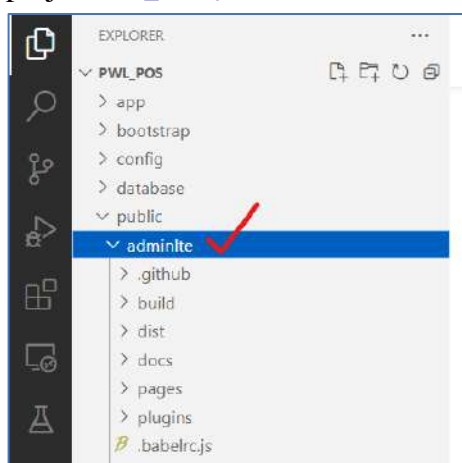


## Praktikum 1 - Layouting AdminLTE:

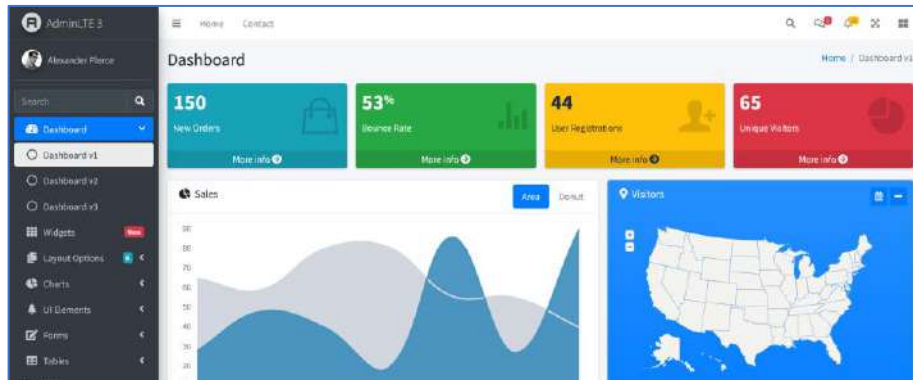
1. Kita download **AdminLTE v3.2.0** yang rilis pada 8 Feb 2022



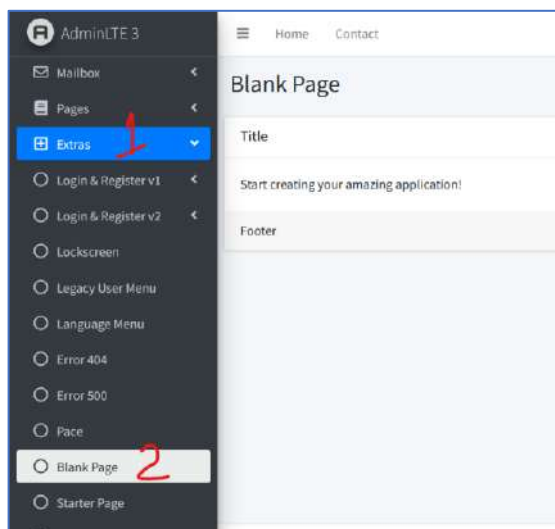
2. Setelah kita berhasil download, kita ekstrak file yang sudah di download ke folder project **PWL\_POS/public**, kemudian kita **rename** folder cukup menjadi **adminlte**



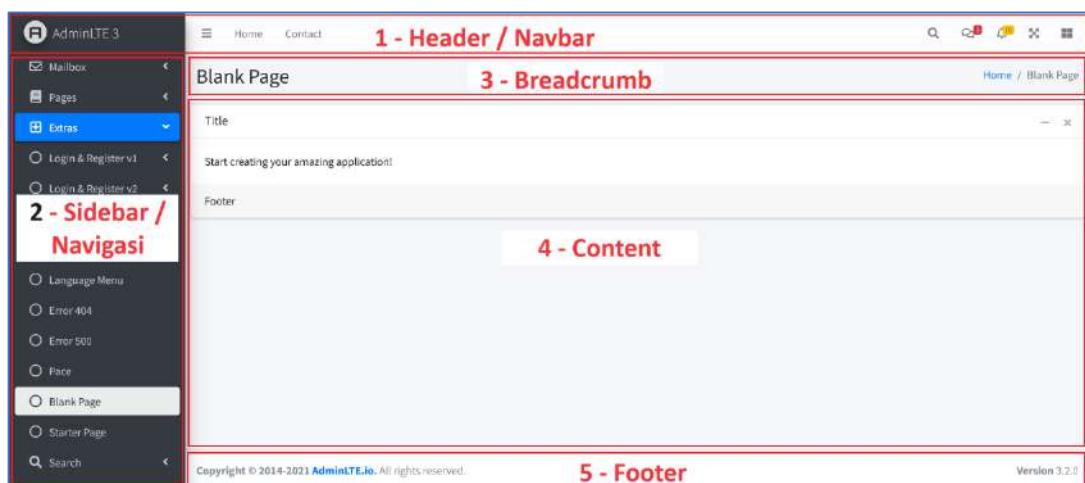
3. Selanjutnya kita buka di browser dengan alamat [http://localhost/PWL\\_POS/public/adminlte](http://localhost/PWL_POS/public/adminlte) maka akan muncul tampilan seperti berikut



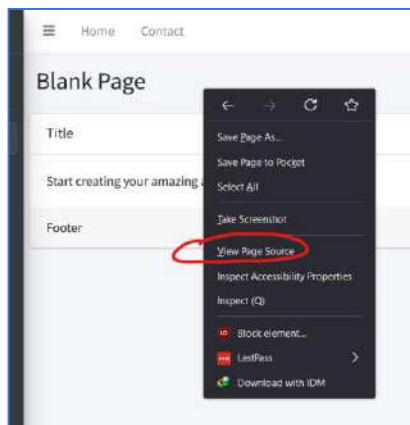
4. Kita klik menu **Extras > Blank Page**, page inilah yang akan menjadi dasar web template



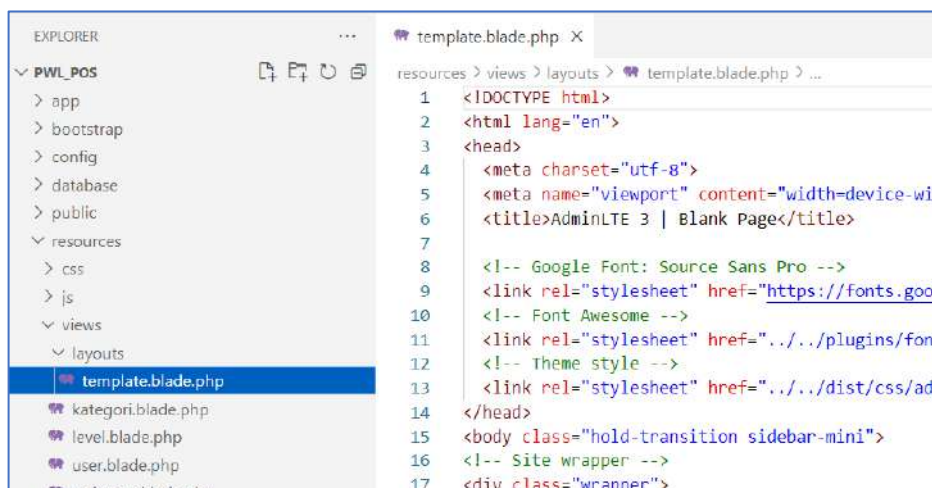
5. Dari sini kita bisa melakukan layouting halaman Blank Page ini menjadi 4 element seperti pada gambar berikut



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



7. Selanjutnya kita copy page source dari halaman **Blank Page**, kemudian 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



```
template.blade.php X
resources > views > layouts > template.blade.php > html > body.hold-transition.sidebar-mini > div.wrapper > nav.main-header.navbar.navbar-expand.navb
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <title>{{ config('app.name', 'PWL Laravel Starter Code') }}</title>
7
8 <!-- Google Font: Source Sans Pro -->
9 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
10 <!-- Font Awesome -->
11 <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
12 <!-- Theme style -->
13 <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
14 </head>
```

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

```
template.blade.php X header.blade.php
template.blade.php > html > body.hold-transition.sidebar-mini > div.wrapper > aside.mai
14 </head>
15 <body class="hold-transition sidebar-mini">
16 <!-- Site wrapper -->
17 <div class="wrapper">
18 <!-- Navbar -->
19 @include('layouts.header')
20 <!-- /.navbar -->
21
22 <!-- Main Sidebar Container -->
23 <aside class="main-sidebar sidebar-dark-primary elevation-4">
24 <!-- Brand Logo -->
25 <a href="../../../index3.html" class="brand-link">
26 AdminLTE 3</span>
28 </a>
```

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**

```
17 <div class="wrapper">
18 <!-- Navbar -->
19 @include('layouts.header')
20 <!-- /.navbar -->
21
22 <!-- Main Sidebar Container -->
23 <aside class="main-sidebar sidebar-dark-primary elevation-4">
24 <!-- Brand Logo -->
25 <a href="../../../index3.html" class="brand-link">
26 
27 <span class="brand-text font-weight-light">AdminLTE 3</span>
28 </a>
29
30 <!-- Sidebar -->
31 <div class="sidebar">
```

Menjadi



```
17 <div class="wrapper">
18 <!-- Navbar -->
19 @include('layouts.header')
20 <!-- /.navbar -->
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 PWL - Starter Code</span>
28 </a>
29
30 <!-- Sidebar -->
31 <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

```
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 PWL - Starter Code</span>
28 </a>
29
30 <!-- Sidebar -->
31 @include('layouts.sidebar')
32 <!-- /.sidebar -->
33 </aside>
```

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

```
81
82 </section>
83 <!-- /.content -->
84 </div>
85 <!-- /.content-wrapper -->
86
87 @include('layouts.footer')
88 </div>
89 <!-- ./wrapper -->
90
91 <!-- jQuery -->
92 <script src="../../plugins/jquery/jquery.min.js"></script>
```

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



```
91 <!-- jQuery -->
92 <script src="../../plugins/jquery/jquery.min.js"></script>
93 <!-- Bootstrap 4 -->
94 <script src="../../plugins/bootstrap/js/bootstrap.bundle.min.js"></script>
95 <!-- AdminLTE App -->
96 <script src="../../dist/js/adminlte.min.js"></script>
97 <!-- AdminLTE for demo purposes -->
98 <script src="../../dist/js/demo.js"></script>
99 </body>
100 </html>
```

Menjadi

```
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') }}"></script>
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 **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

```
30 <!-- Sidebar -->
31 @include('layouts.sidebar')
32 <!-- /.sidebar -->
33 </aside>
34
35 <!-- Content Wrapper. Contains page content -->
36 <div class="content-wrapper">
37 <!-- Content Header (Page header) -->
38 @include('layouts.breadcrumb')
39
40 <!-- Main content -->
41 <section class="content">
42 <!-- Default box -->
43 <div class="card">
44 <div class="card-header">
45 <h3 class="card-title">Title</h3>
46
```

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')`
19. Hasil akhir pada file utama `layouts/template.blade.php` adalah seperti berikut



```
15 <body class="hold-transition sidebar-mini">
16 <!-- Site wrapper -->
17 <div class="wrapper">
18 <!-- Navbar -->
19 @include('layouts.header') 1
20 <!-- /.navbar -->
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 PWL - Starter Code</span>
28 </a>
29
30 <!-- Sidebar -->
31 @include('layouts.sidebar') 2
32 <!-- /.sidebar -->
33 </aside>
34
35 <!-- Content Wrapper. Contains page content -->
36 <div class="content-wrapper">
37 <!-- Content Header (Page header) -->
38 @include('layouts.breadcrumb') 3
39
40 <!-- Main content -->
41 <section class="content">
42 @yield('content') 4
43 </section>
44 <!-- /.content -->
45 </div>
46 <!-- /.content-wrapper -->
47
48 @include('layouts.footer') 5
49 </div>
```

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`

```
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' => $activeMenu]);
16    }
17 }
```

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



```
welcome.blade.php X
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 aplikasi ini.
12         </div>
13     </div>
14 @endsection
```

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

```
welcome.blade.php X breadcrumb.blade.php X
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`

```
<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>
```

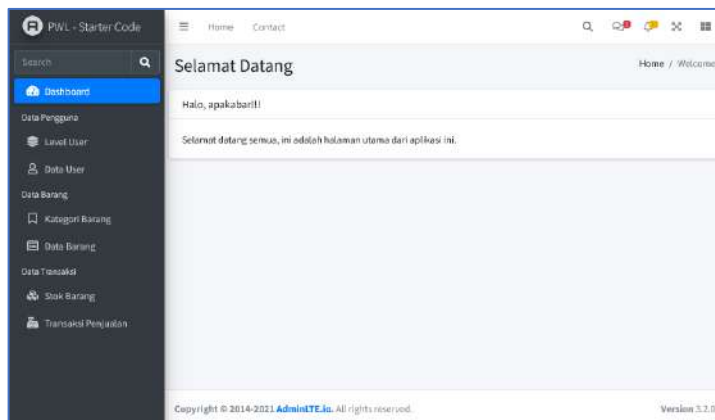


```
<li class="nav-header">Data Pengguna</li>
<li class="nav-item">
  <a href="{{ url('/level') }}" class="nav-link {{ ($activeMenu == 'level')?
'active' : '' }}">
    <i class="nav-icon fas fa-layer-group"></i>
    <p>Level User</p>
  </a>
</li>
<li class="nav-item">
  <a href="{{ url('/user') }}" class="nav-link {{ ($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>
```

5. Kita tambahkan kode berikut router web.php

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

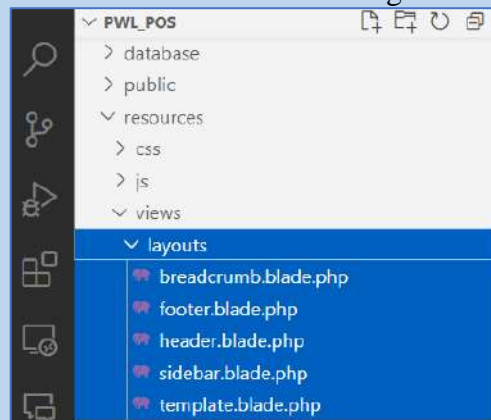
6. Sekarang kita coba jalankan di browser dengan mengetikkan url  
[http://localhost/PWL\\_POS/public](http://localhost/PWL_POS/public)



7. 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 menggunakan 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`
3. Kita modifikasi route `web.php` untuk proses CRUD user

```
6 use App\Http\Controllers\WelcomeController;  
7 use Illuminate\Support\Facades\Route;  
8 use App\Http\Controllers\UserController;  
9  
10  
11 Route::get('/', [WelcomeController::class, 'index']);  
12  
13 Route::group(['prefix' => 'user'], function () {  
14     Route::get('/', [UserController::class, 'index']); // menampilkan halaman awal user  
15     Route::post('/list', [UserController::class, 'list']); // menampilkan data user dalam bentuk json untuk datatables  
16     Route::get('/create', [UserController::class, 'create']); // menampilkan halaman form tambah user  
17     Route::post('/', [UserController::class, 'store']); // menyimpan data user baru  
18     Route::get("/{id}", [UserController::class, 'show']); // menampilkan detail user  
19     Route::get("/{id}/edit", [UserController::class, 'edit']); // menampilkan halaman form edit user  
20     Route::put("/{id}", [UserController::class, 'update']); // menyimpan perubahan data user  
21     Route::delete("/{id}", [UserController::class, 'destroy']); // menghapus data user  
22 });  
23
```

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



```
1 <?php
2 namespace App\Http\Controllers;
3
4 use App\Models\LevelModel;
5 use App\Models\UserModel;
6 use Illuminate\Http\Request;
7 use Yajra\DataTables\Facades\DataTables;
8
9 class UserController extends Controller
10 {
11     // Menampilkan halaman awal user
12     public function index()
13     {
14         $breadcrumb = (object) [
15             'title' => 'Daftar User',
16             'list' => ['Home', 'User']
17         ];
18
19         $page = (object) [
20             'title' => 'Daftar user yang terdaftar dalam sistem'
21         ];
22
23         $activeMenu = 'user'; // set menu yang sedang aktif
24
25         return view('user.index', ['breadcrumb' => $breadcrumb, 'page' => $page, 'activeMenu' => $activeMenu]);
26     }
27 }
```

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</th></tr>
                </thead>
                <tbody>
                    <tr><td>Pegawai</td><td>Aksi</td></tr>
                </tbody>
            </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
```

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

```
template.blade.php X
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="utf-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <title>{{ config('app.name', 'PWL Laravel Starter Code') }}</title>
7
8 <meta name="csrf-token" content="{{ csrf_token() }}"> <!-- Untuk mengirimkan token Laravel CSRF pada setiap request ajax -->
9
10 <!-- Google Font: Source Sans Pro -->
11 <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,700&display=fallback">
12 <!-- Font Awesome -->
13 <link rel="stylesheet" href="{{ asset('adminlte/plugins/fontawesome-free/css/all.min.css') }}">
14 <!-- DataTables -->
15 <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-bs4/css/dataTables.bootstrap4.min.css') }}">
16 <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-responsive/css/responsive.bootstrap4.min.css') }}">
17 <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-buttons/css/buttons.bootstrap4.min.css') }}">
18 <!-- Theme style -->
19 <link rel="stylesheet" href="{{ asset('adminlte/dist/css/adminlte.min.css') }}">
20
21 @stack('css') <!-- Digunakan untuk memanggil custom css dari perintah push('css') pada masing-masing view -->
22 </head>
```



```
60 <!-- jQuery -->
61 <script src="{{ asset('adminlte/plugins/jquery/jquery.min.js') }}"></script>
62 <!-- Bootstrap 4 -->
63 <script src="{{ asset('adminlte/plugins/bootstrap/js/bootstrap.bundle.min.js') }}"></script>
64 <!-- DataTables & Plugins -->
65 <script src="{{ asset('adminlte/plugins/datatables/jquery.dataTables.min.js') }}"></script>
66 <script src="{{ asset('adminlte/plugins/datatables-bs4/js/dataTables.bootstrap4.min.js') }}"></script>
67 <script src="{{ asset('adminlte/plugins/datatables-responsive/js/dataTables.responsive.min.js') }}"></script>
68 <script src="{{ asset('adminlte/plugins/datatables-responsive/js/responsive.bootstrap4.min.js') }}"></script>
69 <script src="{{ asset('adminlte/plugins/datatables-buttons/js/dataTables.buttons.min.js') }}"></script>
70 <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.bootstrap4.min.js') }}"></script>
71 <script src="{{ asset('adminlte/plugins/jszip/jszip.min.js') }}"></script>
72 <script src="{{ asset('adminlte/plugins/pdfmake/pdfmake.min.js') }}"></script>
73 <script src="{{ asset('adminlte/plugins/pdfmake/vfs_fonts.js') }}"></script>
74 <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.html5.min.js') }}"></script>
75 <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.print.min.js') }}"></script>
76 <script src="{{ asset('adminlte/plugins/datatables-buttons/js/buttons.colVis.min.js') }}"></script>
77 <!-- AdminLTE App -->
78 <script src="{{ asset('adminlte/dist/js/adminlte.min.js') }}"></script>
79 <script>
80 // Untuk mengirimkan token Laravel CSRF pada setiap request ajax
81 $.ajaxSetup({headers: {'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')}});
82 </script>
83 @stack('js') <!-- Digunakan untuk memanggil custom js dari perintah push('js') pada masing-masing view -->
84 </body>
85 </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).'">';
                csrf_field() . method_field('DELETE') .
                '<button type="submit" class="btn btn-danger btn-sm" onclick="return confirm(\'Apakah Anda yakin menghapus data ini?\');">Hapus</button></form>';
            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>
            <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>
        </form>
    </div>
</div>
```



```
<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
```

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 Detail) 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
});
```



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**

```
@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
```

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::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
});
```

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`

```
@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
```

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 DELETE 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 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 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

```
11 <div class="card-body">
12     @if (session('success'))
13         <div class="alert alert-success">{{ session('success') }}</div>
14     @endif
15     <table class="table table-bordered table-striped table-hover table-sm" id="table_user">
16         <thead>
17             <tr><th>ID</th><th>Username</th><th>Nama</th><th>Level Pengguna</th><th>Aksi</th></tr>
18         </thead>
19     </table>
20 </div>
```

Menjadi



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

## 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.

ID	Username	Nama	Level Pengguna	Aksi
1	admin	Administrator	Administrator	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
2	manager	Manager	Manager	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
3	supervisor	Supervisor	Supervisor	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
4	kasir1	Kasir 1	Kasir/Staff	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
5	kasir2	Kasir 2	Kasir/Staff	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>
6	administrator	Administrator	Administrator	<a href="#">Detail</a> <a href="#">Edit</a> <a href="#">Hapus</a>

#### 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*



```
// 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

    $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`

```
@if (session('error'))
    <div class="alert alert-danger">{{ session('error') }}</div>
@endif
<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>
</div>
<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
```

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



```
46 @push('js')
47 <script>
48 $(document).ready(function() {
49     var dataUser = $('#table_user').DataTable({
50         serverSide: true, // serverSide: true, jika ingin menggunakan server side processing
51         ajax: {
52             "url": "{{ url('user/list') }}",
53             "dataType": "json",
54             "type": "POST",
55             "data": function (d) {
56                 d.level_id = $('#level_id').val();
57             }
58         },
59         columns: [
60             {
61                 data: "DT RowIndex", // nomor urut dari laravel datatable addIndexColumn()
```

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

```
81         data: "aksi",
82         className: "",
83         orderable: false, // orderable: true, jika ingin kolom ini bisa diurutkan
84         searchable: false // searchable: true, jika ingin kolom ini bisa dicari
85     }
86 ]
87 });
88
89 $('#level_id').on('change', function() {
90     dataUser.ajax.reload();
91 });
92
93 });
94 </script>
95 @endpush
```

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

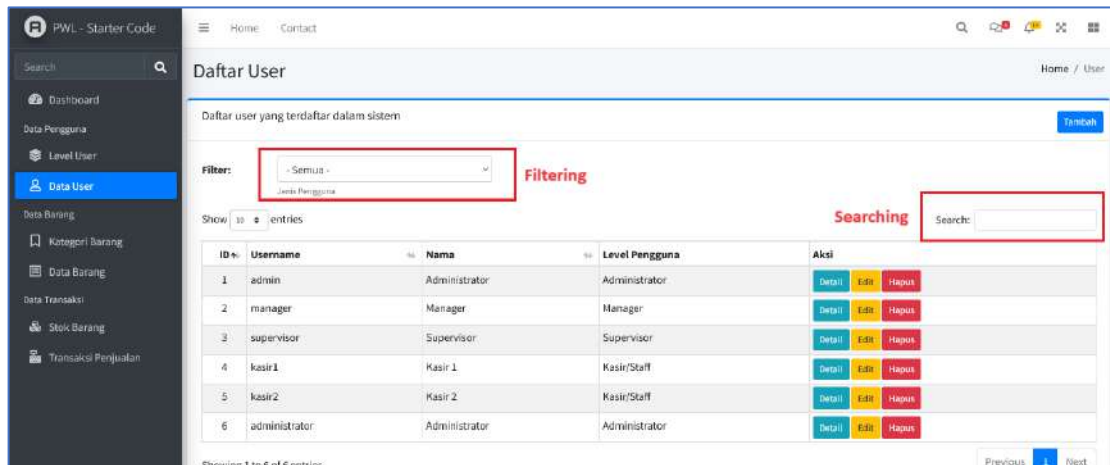
```
// Ambil data user dalam bentuk json untuk datatables
public function list(Request $request)
{
    $users = UserModel::select('user_id', 'username', 'nama', 'level_id')
        ->with('level');

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

    return DataTables::of($users)
        ->addIndexColumn() // menambahkan kolom index / no urut (default nama kolom: DT_RowIndex)
        ->addColumn('aksi', function ($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 yakin menghapus data ini?\')>Hapus
            return $btn;
        })
        ->rawColumns(['aksi']) // memberitahu bahwa kolom aksi adalah html
        ->make(true);
}
```



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

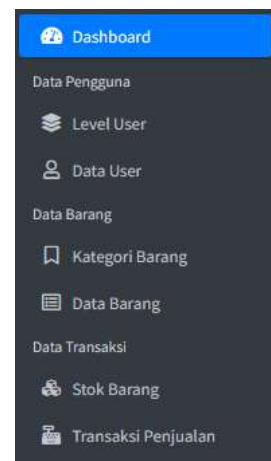


7. Selamat, sekarang Laravel Starter Code sudah ada filtering dan searching data. Starter Code sudah bisa digunakan dalam membangun sebuah sistem berbasis website.
8. Jangan lupa commit dan push ke github PWL\_POS kalian

## G. Tugas

Implementasikan web layout dan datatables, pada menu berikut ini

- ✓ Tabel m\_level
- ✓ Tabel m\_kategori
- ✓ Tabel m\_supplier
- ✓ Tabel m\_barang



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