



Courses : Advanced Web Programming (PWL)
Studina Program : D4 – Informatics Engineering / D4 – Business Information Systems
Semester : 4 (four) / 6 (six)
Meeting to- : 7 (seven)

JOBSHEET 07

PWL – LARAVEL STARTER CODE

Earlier we discussed *templating*, implementing the *laravel-adminLTE composer library*, and *laravel-yajra-datatable composer library*. Of course, these libraries are very sophisticated and can facilitate and accelerate the development of the website we are working on.

However, in this meeting, we will learn the basic implementation of adminLTE (without libraries) and the simple implementation of yajra-datatable. This is intended so that you go deeper into the basic application of laravel by using libraries.

Before we enter the material, we first create a new project that we will use to build a simple application with the topic of *Point of Sales (PoS)*, according to **the Case Study PWL.pdf**. So we created a Laravel 10 project with the name **PWL_POS**.

We will use PWL_POS project until the 12th meeting later, as a project that we will study

A. WEB TEMPLATE

Is a file that already has a certain design so, we just need to modify certain parts to get good results. By using a template that suits our needs, we can reduce the time to create a website drastically.

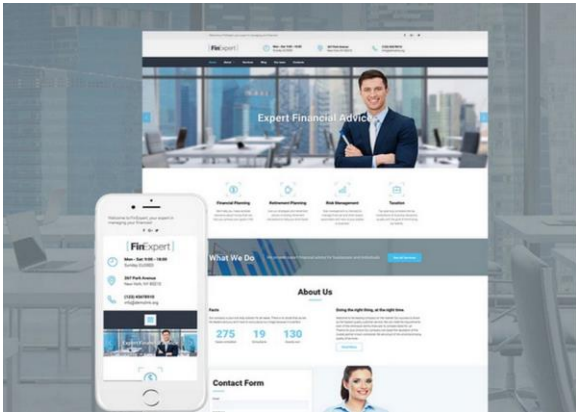
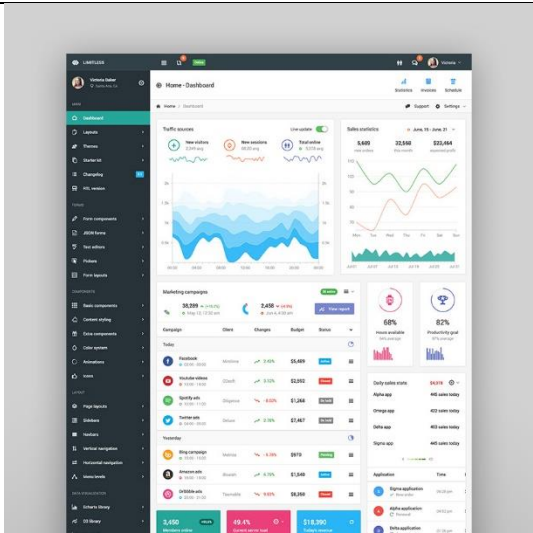

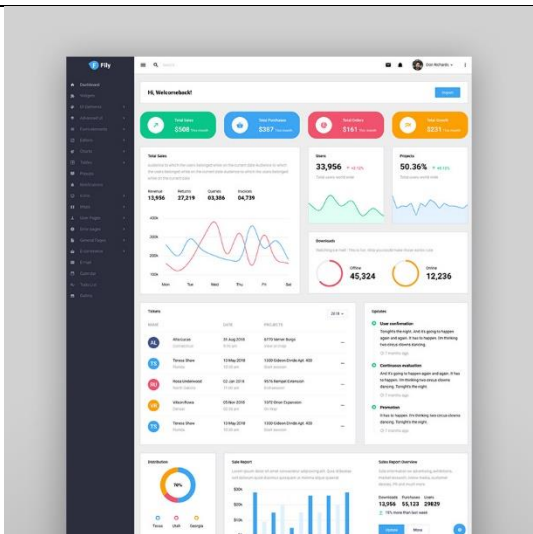
In general, we can use website templates that already exist on the internet to make it easier for us to create a website. There are many templates on the internet and we take one example template for the administrator page, such as **the AdminLTE** template.

AdminLTE is one of the templates that is often used by web developers as a backend / admin template on projects that are often worked on. This template was created using the bootstrap framework which is the most widely used CSS framework among web designers.

The AdminLTE template can be downloaded at <https://adminlte.io>



Sample Web Template

Frontend Template	Backend Template
	
	

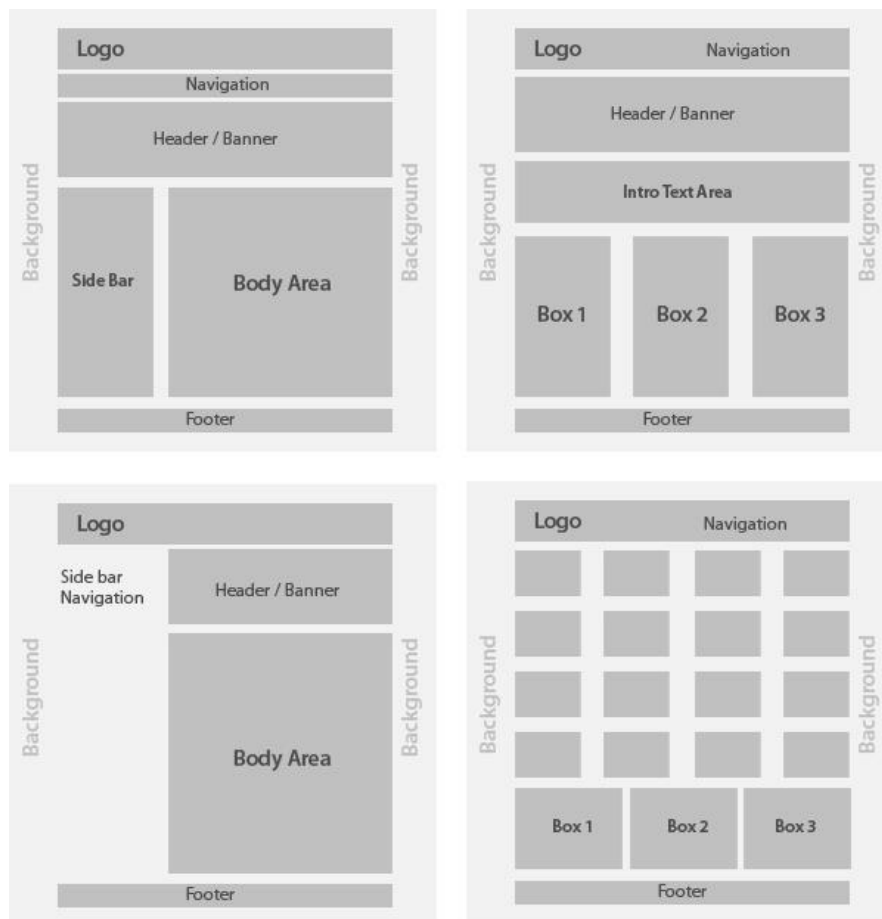
B. WEB LAYOUT

Is a set of design elements or variables related to certain media to support the concept of website creation. Basically, the purpose of using website layouts is to combine elements used to build websites. In addition to the elements in it, the website layout must also contain several elements, ranging from headers, navigation, sidebars, to footers. To understand the different elements of a website layout, here's a brief explanation

- **Header** → The layout designer can fill this section with website logos, site navigation, social media icons, and search menus.



- **Navigation** → Navigation can be understood as a guide. On websites, navigation can range from menus that appear at the top of web pages to other support menus commonly found at the bottom of the site.
- **Body/Content** → In addition, there are body/content elements that are usually filled with product information, product features, and descriptions of products sold.
- **Sidebar** → The element on the right / left side of a website. This element is sometimes deprecated, especially on the Web Frontend. However, sidebars are still used on many article pages, which will then be filled with product information, most popular products, and additional navigation.
- **Footer** → The last element is the footer. Footer is an element that is under itself as a supporting menu or as the identity of a website.





C. LAYOUTING ADMINLTE

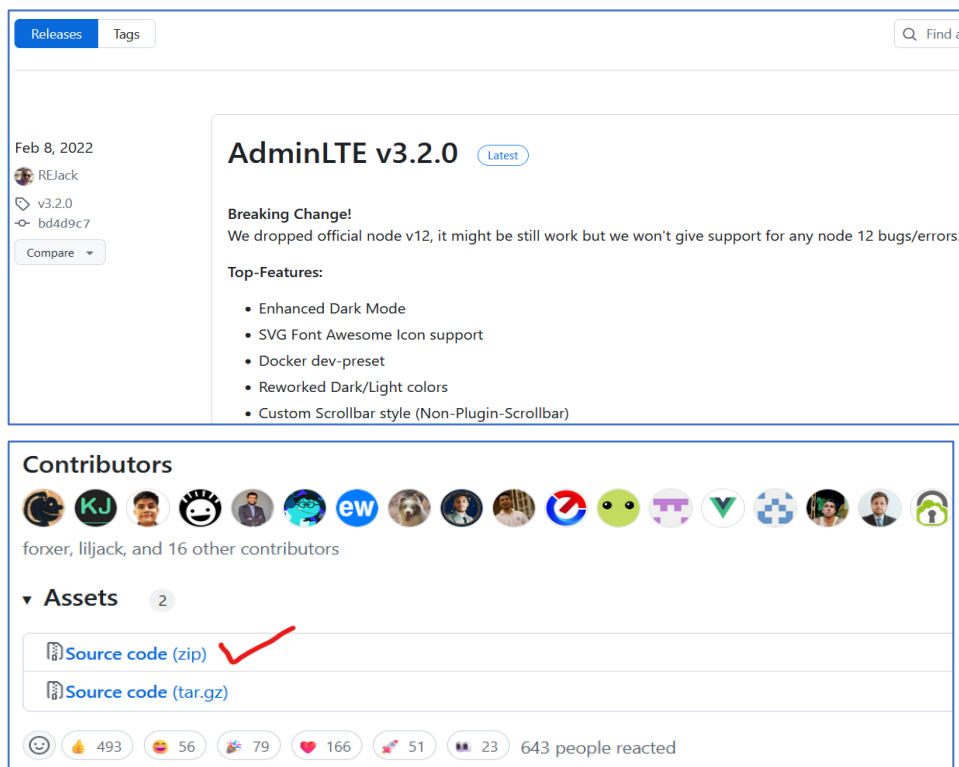
This time we will *do the* AdminLTE template layout. The purpose of *this layouting* is to break down parts of the adminLTE template into various elements. This can make it easier for us to create a website because we can use repeatedly the elements that we have created later.

In this layouting, we need Laravel's **Blade Template Engine** to break the web into elements. The blade components that we need for the layouting process at this meeting are

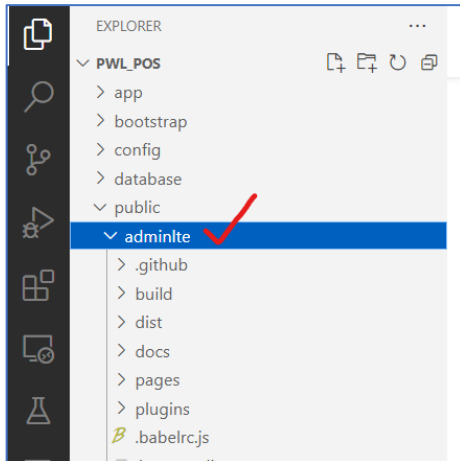
- `@include()`
- `@extend()`
- `@section()`
- `@push()`
- `@yield()`
- `@stack()`

Practicum 1 – Layouting AdminLTE:

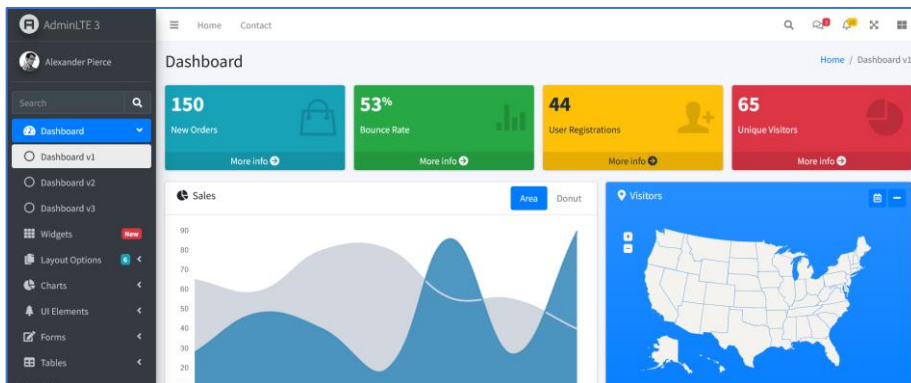
1. We download **AdminLTE v3.2.0** which was released on 8 Feb 2022



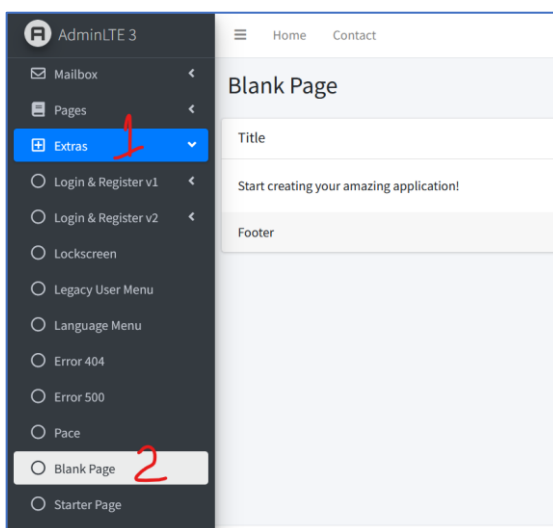
2. After we successfully download, we extract the downloaded file to the project folder `PWL_POS / public`, then we **rename** the folder simply to `adminlte`



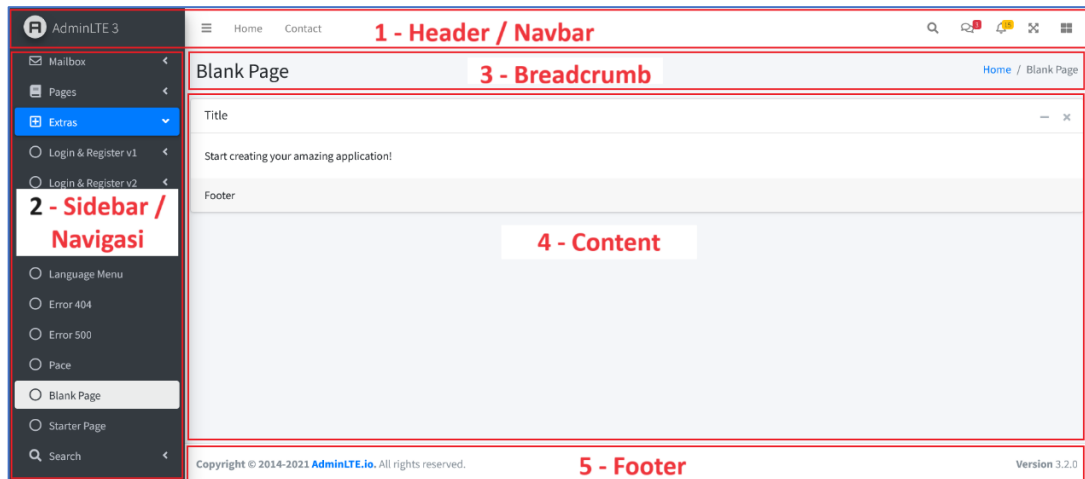
3. Next, we open it in the browser with the http://localhost/PWL_POS/public/adminlte address, a display will appear as follows:



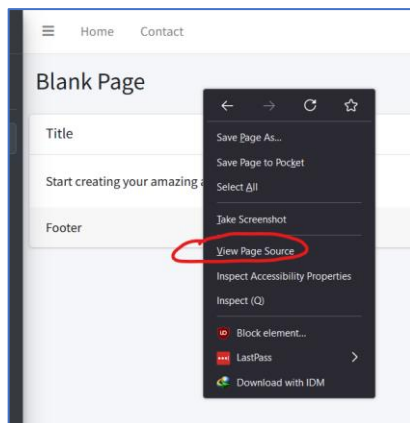
4. We click the **Extras > Blank Page** menu, this page will be the basis of the web template



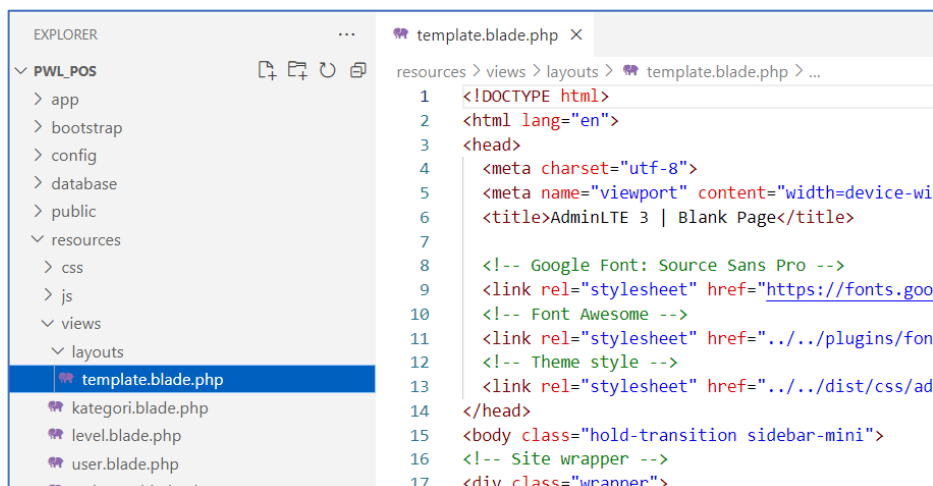
5. From here we can layout this Blank Page page into 4 elements as shown below



6. Next we right-click the **Blank Page** page and click *view page source*



7. Next we copy the page source from the **Blank Page**, then we *paste* it in `PWL_POS/resource/view/layouts/template.blade.php` (first create the **layouts** folder and `template.blade.php` file)



8. The `layouts/template.blade.php` file is the main file for website templating
9. On lines **1-14** of the `template.blade.php` file, we modify



```
template.blade.php X
resources > views > layouts > template.blade.php > html > body.hold-transition.sidebar-mini > div.wrapper > nav.main-header.navbar.navbar-expand.navbar
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>AdminLTE 3 | Blank Page</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="../../plugins/fontawesome-free/css/all.min.css">
12  <!-- Theme style -->
13  <link rel="stylesheet" href="../../dist/css/adminlte.min.css">
14 </head>
```

Become

```
template.blade.php X
resources > views > layouts > template.blade.php > html > body.hold-transition.sidebar-mini > div.wrapper > nav.main-header.navbar.navbar-expand.navbar
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. Then we block lines **19-153** (the line for **element 1-header**), then we **cut**, and **paste** it in the file **PWL_POS/resource/view/layouts/header.blade.php** (create the file **header.blade.php** first if it doesn't already exist). So that the display of the **template.blade.php** file becomes as follows

```
template.blade.php X header.blade.php
mplate.blade.php > html > body.hold-transition.sidebar-mini > div.wrapper > aside.mair
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       
27       <span class="brand-text font-weight-light">AdminLTE 3</span>
28     </a>
```

Line **19** is the Blade component to call the **layouts/header.blade.php** element to become one with the **template.blade.php** when rendered later.

11. We modify lines **25** and **26** in **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">
```

Become

```
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. Next we block lines **31-693** (the line for **element 2-sidebar**), then we **cut**, and **paste** it in the file **PWL_POS/resource/view/layouts/sidebar.blade.php** (create the file sidebar.blade.php first if it doesn't already exist). So that the display of the template.blade.php file becomes as follows

```
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. Next look at lines **87-98** (lines for **the 5-footer** element), then we **cut**, and **paste** them in the file **PWL_POS/resource/view/layouts/footer.blade.php** (create a footer.blade.php file if it doesn't already exist). So that the display of the template.blade.php file becomes as follows



```
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. Then we modify the file `template.blade.php` lines **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>
```

Become

```
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. Now go to the content section. Our content is divided into 2, namely elements for **breadcrumbs** and elements for **content**.

16. Notice the `template.blade.php` file on lines **38-52** we make it a **4-breadcrumb** element.

We block lines **38-52** and then we **cut**, and **paste** them in the `PWL_POS/resource/view/layouts/breadcrumb.blade.php` file (create a `breadcrumb.blade.php` file if it doesn't already exist). So that the display of the `template.blade.php` file becomes as follows

```
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. The last layout is in the content section. We can make layouts for content dynamic, according to what we want to present on the web we build.
18. For **content**, we'll delete lines 42-66 in template.blade.php file, and replace it with code like this `@yield('content')`
19. The final result in the main `layouts/template.blade.php` file is as follows:

```
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. Congratulations you have finished layouting the website in Laravel.

Practicum 2 – Application of Layouting:

Now we will try to apply the layouting that we have done.

1. We create a controller file named `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. We create a file at `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. We modify the file `PWL_POS/resources/views/layouts/breadcrumb.blade.php`

```
welcome.blade.php 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. We modify the 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 -->
    <div 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 { { ($activeMenu == 'user')?
'active' : '' } } ">
        <i class="nav-icon far fa-user"></i>
        <p>Data User</p>
    </a>
</li>
<li class="nav-header">Data Barang</li>
<li class="nav-item">
    <a href="{ { url('/category') } }" class="nav-link { { ($activeMenu ==
'category')? 'active' : '' } } ">
        <i class="nav-icon far fa-bookmark"></i>
        <p>Category Barang</p>
    </a>
</li>
<li class="nav-item">
    <a href="{ { url('/item') } }" class="nav-link { { ( ($activeMenu == 'item')?
'active' : '' } } ">
        <i class="nav-icon far fa-list-alt"></i>
        <p>Goods Data</p>
    </a>
</li>
<li class="nav-header">Data Transaksi</li>
<li class="nav-item">
    <a href="{ { url('/stok') } }" class="nav-link { { ($activeMenu == 'stok')?
'active' : '' } } ">
        <i class="nav-icon fas fa-cubes"></i>
        <p>Stock of Goods</p>
    </a>
</li>
<li class="nav-item">
```



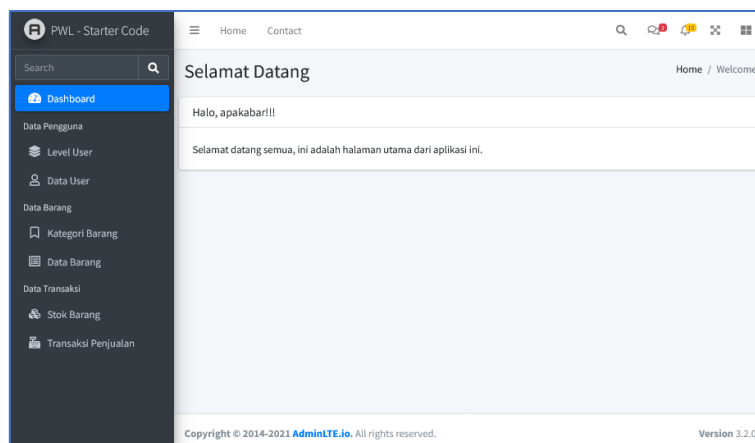
```
<a href="{{ url('/item') }}" class="nav-link {{ ( $activeMenu ==  
'sales')? 'active' : '' }}">  
    <i class="nav-icon fas fa-cash-register"></i>  
    <p>Sales Transaction</p>  
</a>  
</li>  
</ul>  
</nav>  
</div>
```

5. We add the following code router web.php

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

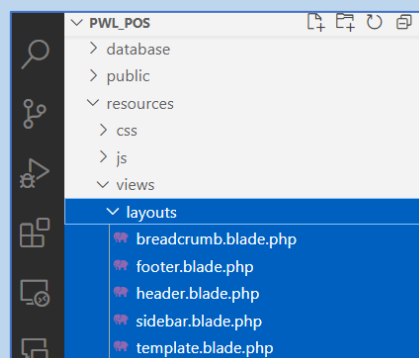
6. Now we try to run it in the browser by typing url

http://localhost/pal_pos/public



INFO

There are 5 main files in the *layouts* folder used in building a website, namely **templates**, **headers**, **sidebars**, **breadcrumbs**, **footers**. You can modify the 5 files in building a website.





D. LARAVEL DATATABLE

To display a lot of data, we can display the data in table format. DataTable is a **jQuery** plugin created to manage information data in the form of grids / tables.

INFO

AdminLTE has also implemented the Datatable library in its templates. You can check the Datatable on AdminLTE at

http://localhost/pal_pos/public/admin/pages/tables/data.html

In this practicum we **do not use Vite/NodeJS** in managing datatables, we simply use jQuery datatables built in AdminLTE and Yajra *laravel-datatables library*.

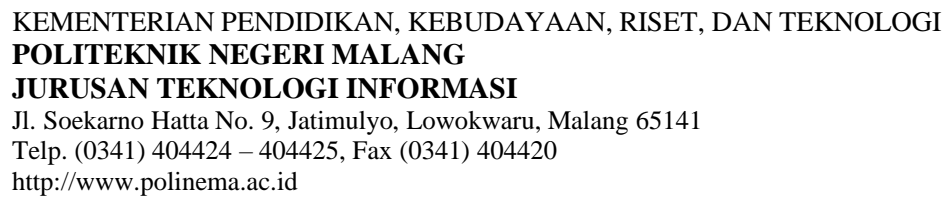
We will implement the use of jQuery datatable from AdminLTE with Laravel. In this implementation, we use the Yajra library *laravel-datatables*.

Practicum 3 – Implementation of jQuery Datatable in AdminLTE :

1. We modify the CRUD process in table `m_user` in this lab
2. We use the Yajra-datatable library by typing the command in CMD
`composer require yajra/laravel-datatables:^10.0` or
`composer require yajra/laravel-datatables-oracle`
3. We modify the web.php route for the CRUD user process

```
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. We make or fully modify it to `UserController.php`. We create an `index()` function to display the user's start page



- Then we create a view at [PWL_POS/resources/views/user/index.blade.php](#)

Jobsheet 07 – PWL 2023/2024



```
<script>
$(document).ready(function() {
    var dataUser = $('#table_user'). DataTable({
serverSide: true,  serverSide: true, if you want to use server side processing
ajax: {
        "url": "{{ url('user/list') }}",
        "dataType": "json",
        "type": "POST"
    },
    columns: [
        {
            data: "DT_RowIndex",  sequence number of laravel datatable addIndexColumn()
ClassName: "text-center",
            orderable: false,
            searchable: false
        },{
            data: "username",
ClassName: "",
            orderable: true,  orderable: true, if you want this column to be sortable
            searchable: true  searchable: true, if you want this column to be searchable
        },{
            data: "name",
ClassName: "",
            orderable: true,  orderable: true, if you want this column to be sortable
            searchable: true  searchable: true, if you want this column to be searchable
        },{
            data: "level.level_nama",
ClassName: "",
            orderable: false,  orderable: true, if you want this column to be sortable
            searchable: false  searchable: true, if you want this column to be searchable
        },{
            data: "action",
ClassName: "",
            orderable: false,  orderable: true, if you want this column to be sortable
            searchable: false  searchable: true, if you want this column to be searchable
        }
    ]
    });
});
</script>
@endpush
```

6. Then we modify the `template.blade.php` file to add the jquery datatables library from the AdminLTE template that we downloaded and are in the `public` folder



```
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/dataTables.responsive.min.css') }}">
17 <link rel="stylesheet" href="{{ asset('adminlte/plugins/datatables-buttons/css/dataTables.buttons.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/dataTables.responsive.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/dataTables.buttons.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. To be able to capture request data for the datatable, we create a **list()** function in the **UserController.php** as follows

```
Retrieve user data in json form for datatables
public function list(Request $request)
{
    $users = UserModel::select('user_id', 'username', 'nama', 'level_id')
        ->with('level');

    return DataTables::of($users)
        ->addIndexColumn() adds column index/no sort (default column name: DT_RowIndex)
        ->addColumn('action', function ($user) { add action column
```



```
$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(\'Are you sure to delete this data?\');"  
>Delete</button></form>';  
        return $btn;  
    })  
->rawColumns(['action'])  tells that the action column is html  
->make(true);  
}
```

8. Now try to run the browser, and click the **Data User...** menuWatch and observe what happens.

9. Next, we modify the `UserController.php` for the form to add user data

```
// 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. Now that we create a form to add data, we create a file `PWL_POS/resources/views/user/create.blade.php`

```
@extends('layouts.template')  
  
@section('content')  
    <div class="card card-outline card-primary">  
        <div class="card-header">  
            <h3 class="card-title">{{ $page->title }}</h3>  
            <div class="card-tools"></div>  
        </div>  
        <div class="card-body">  
            <form method="POST" action="{{ url('user') }}" class="form-horizontal">  
                @csrf  
                <div class="form-group row">
```



```
<label class="col-1 control-label col-form-label">Level</label>
<div class="col-11">
  <select class="form-control" id="level_id" name="level_id" required>
    <option value="">- Pilih Level -</option>
    @foreach($level as $item)
      <option value="{{ $item->level_id }}">{{ $item->level_nama
    }}</option>
  @endforeach
</select>
  @error('level_id')
    <small class="form-text text-danger">{{ $message }}</small>
  @enderror
</div>
</div>
<div class="form-group row">
  <label class="col-1 control-label col-form-label">Username</label>
  <div class="col-11">
    <input type="text" class="form-control" id="username" name="username"
value="{{ old('username') }}" required>
    @error('username')
      <small class="form-text text-danger">{{ $message }}</small>
    @enderror
  </div>
</div>
<div class="form-group row">
  <label class="col-1 control-label col-form-label">Nama</label>
  <div class="col-11">
    <input type="text" class="form-control" id="nama" name="nama"
value="{{ old('nama') }}" required>
    @error('nama')
      <small class="form-text text-danger">{{ $message }}</small>
    @enderror
  </div>
</div>
<div class="form-group row">
  <label class="col-1 control-label col-form-label">Password</label>
  <div class="col-11">
    <input type="password" class="form-control" id="password"
name="password" required>
    @error('password')
      <small class="form-text text-danger">{{ $message }}</small>
    @enderror
  </div>
</div>
<div class="form-group row">
```



```
<label class="col-1 control-label col-form-label"></label>
<div class="col-11">
  <button type="submit" class="btn btn-primary btn-sm">Simpan</button>
  <a class="btn btn-sm btn-default ml-1" href="{ url('user')
}}">Kembali</a>
</div>
</div>
</form>
</div>
</div>
@endsection
@push('css')
@endpush
@push('js')
@endpush
```

11. Then to be able to *handle* the data to be saved to the database, we create the **store()** **function** in the **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. Now try to open the add user data form by clicking the add button. Observe and learn..!!

13. Next, we enter the section displaying user data details (click the **Detail** button) on the user page. The route in charge of capturing the request details is

```
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. So we create/modify the **show()** **function** on the **UserController.php** as follows



```
// 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'
    ];

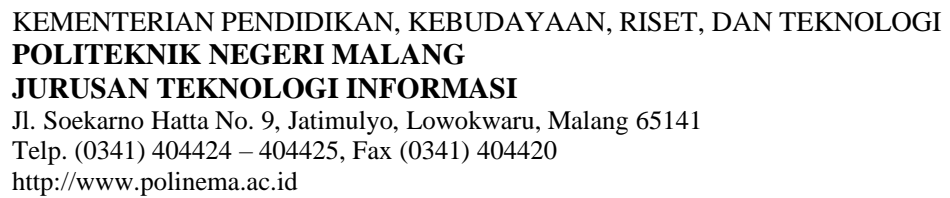
    $activeMenu = 'user'; // set menu yang sedang aktif

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

15. Then we create a *view* in `PWL_POS/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>
                    The data you are looking for was not found.
                </div>
            @else
                <table class="table table-bordered table-striped table-hover table-sm">
                    <tr>
                        <td><i>id</i></td>
                        <td>{{ $user->user_id }}</td>
                    </tr>
                    <tr>
                        <td><i>level</i></td>
                        <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>
                </table>
            @endempty
        </div>
    </div>
</section>
```



16. Now you try to see the details of the user data in the browser, and try to type in the wrong id example `http://localhost/PWL_POS/public/user/100` observe what happened, and report it!!
17. Next, we enter the section to modify user data. The route in charge of capturing edit requests is

18. So we create **edit()** and **update()** functions on 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. Next, we create a view to edit user data in
PWL_POS/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>
                    The data you are looking for was not found.
                </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')!!} <!-- add this line for edits that need the PUT -->
method

<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">Ignore (do not fill in) if you
do not want to change the 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. Now you try to edit user data in the browser, observe, understand, and report!

21. Next we'll create a handle for the delete button. The `web.php` router that functions to capture delete requests with the DELETE method is

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

22. So we create a **destroy() function** on `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. Next we modify the `PWL_POS/resources/views/user/index.blade.php` file to add a view if there is an error message



```
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>
```

Become

```
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. Then run the browser to delete one of the user data. Observe and report!

25. Congratulations, you have created Laravel Starter Code for the CRUD process using the AdminLTE template and the jQuery Datatables plugin.

E. DATA SEARCHING AND FILTERING

In displaying a data, sometimes we need to do a search based on certain keywords or filtering data based on a category (*filtering*).

1. **Searching** (Pencarian):

- Searching is the process of finding certain information or data that matches the given criteria.
- Usually, searches are performed using specific keywords or phrases to match with existing data.
- The purpose of a search is to find entities that match the search criteria, be it in a database, in a text document, or in any other context.
- Searches often involve matching patterns or keywords in existing text or data, and results may include entities that partially match or are similar to the search criteria.

2. **Filtering** :

- Filtering is the process of selecting or limiting a set of data or entities based on certain criteria.



- Usually, filtering is done to filter existing data based on certain attributes or characteristics, such as dates, categories, or other attributes.
- The purpose of filtering is to present data that is relevant or relevant to a particular user's needs or criteria.
- Filtering can be done by selecting data based on values that match certain criteria or by excluding data that does not meet those criteria.

The main difference between *searching* and *filtering* is that *searching* deals with searching for data that fits certain criteria, while *filtering* deals with filtering existing data based on certain criteria. Although the two are often used together in applications or information systems, they serve different purposes in the process of managing and analyzing data.

By default, jQuery datatables already has a searching feature that is integrated with laravel yajra-datatables. However, there is no data *filtering* process . For that we will create filtering data in our Laravel Starter Code.

The screenshot shows a web application interface for 'PWL - Starter Code'. The main content area is titled 'Daftar User' and displays a table of registered users. A search bar is highlighted with a red box, indicating the 'Searching' feature. The table has columns for ID, Username, Nama, Level Pengguna, and Aksi. The data is as follows:

ID	Username	Nama	Level Pengguna	Aksi
1	admin	Administrator	Administrator	Detail Edit Hapus
2	manager	Manager	Manager	Detail Edit Hapus
3	supervisor	Supervisor	Supervisor	Detail Edit Hapus
4	kasir1	Kasir 1	Kasir/Staff	Detail Edit Hapus
5	kasir2	Kasir 2	Kasir/Staff	Detail Edit Hapus
6	administrator	Administrator	Administrator	Detail Edit Hapus

Practicum 4 – Implementation of *Filtering* Datatables:

We will apply filtering to the datatable that we have created. This will make it easier for us to group a data according to certain categories. The steps we work through are as follows

1. We modify the `index()` function in UserController.php to add the data we want to be categorized for 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. Then we modify the view to display filtering data on `PWL_POS/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. Next, staying on the `index.blade.php` view, we add the following code to the ajax declaration in the datatable. This code is used to transmit data for 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. Then we edit at the end of the script @push('js') to add a listener if filtering data is selected

```
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
90 $('#level_id').on('change', function() {
91     dataUser.ajax.reload();
92 });
93
94 });
95 </script>
96 @endpush
```

5. The final step is to modify the **list()** function on the **UserController.php** used to display data in the datatable

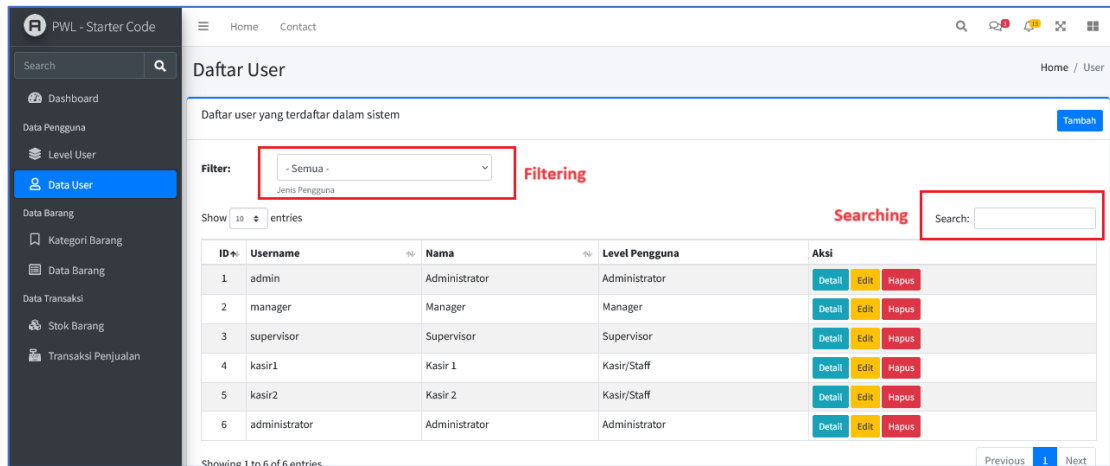
```
// Ambil data user dalam bentuk json untuk datatables
public function list(Request $request)
{
    $users = UserModel::select('user_id', 'username', 'nama', 'level_id')
        ->with('level');

    // 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?\')</form>';
            return $btn;
        })
        ->rawColumns(['aksi']) // memberitahu bahwa kolom aksi adalah html
        ->make(true);
}
```




6. The final part is we try to run it in the browser with user menu access, it will appear as follows



7. Congratulations, now Laravel Starter Code has filtering and searching data. Starter Code can already be used in building a website-based system.

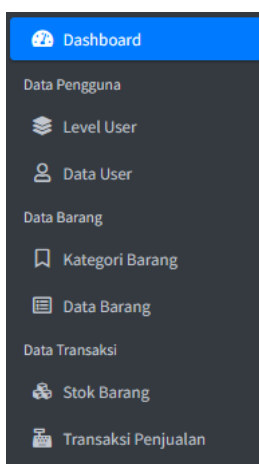
F. QUESTION

Answer the following questions according to the understanding of the material above

1. How is a *template frontend* different from a *template backend*?
2. Is *layouting* important in building a website?
3. Explain the functions of the following laravel blade components: `@include()`, `@extend()`, `@section()`, `@push()`, `@yield()`, and `@stack()`
4. What is the function and purpose of `$activeMenu` variable ?

G. ASSIGNMENT

Implement this missing menu in laravel starter code according to Simple Point of Sales Case Study. Please apply the code in the laravel starter code for the menus corresponding to the menu next to this.



*Thank you, and good luck ****