

LAPORAN PRAKTIKUM
JOBSHEET 05
BLADE VIEW, WEB TEMPLATING (AdminLTE), DATATABLES

Disusun untuk memenuhi nilai tugas
Mata Kuliah : Pemrograman Web Lanjut



Oleh :
Aqueena Regita Hapsari
2341760096
SIB 2B
03

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS
POLITEKNIK NEGERI MALANG
TAHUN AJARAN 2024/2025

Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 5 (satu)

JOBSHEET 05

Blade View, Web Templating(AdminLTE), Datatables

View merupakan tempat bagi kita untuk meletakkan kode-kode HTML. Sehingga dalam laravel tidak menggunakan lagi file berformat .html. Kita masih menggunakan tagging HTML, tapi kita perlu handle tampilan dengan lebih canggih. Menampilkan data yang diberikan oleh controller. Untuk itu kita akan menggunakan templating engine, yaitu Blade.

Blade merupakan templating engine bawaan Laravel. Berguna untuk mempermudah dalam menulis kode tampilan. Dan juga memberikan fitur tambahan untuk memanipulasi data di view yang dilempar dari controller. Blade juga memungkinkan penggunaan plain PHP pada kode View.

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. View

Karena Laravel menggunakan templating engine bawaan Blade, maka setiap file View diakhiri dengan .blade.php. Misal: index.blade.php, home.blade.php, product.blade.php. Contoh sederhana dari view dengan nama *file* hello.blade.php adalah sebagai berikut.

```
<!-- View pada resources/views/hello.blade.php -->
<html>
    <body>
        <h1>Hello, {{ $name }}</h1>
    </body>
</html>
```

View tersebut dapat dijalankan melalui Routing, dimana route akan memanggil View sesuai dengan nama file tanpa 'blade.php'

```
Route::get('/hello', function () {
    return view('hello', ['name' => 'Andi']);
});
```

Setiap kita membuat view, kita mungkin akan menampilkannya melalui router atau controller menggunakan global view helper. Selain menggunakan global view helper, kita juga dapat menggunakan view facade untuk menampilkan view.

```
Use Illuminate\Support\Facades\View;
return View::make('hello', ['name' => 'Andi']);
```

Seperti yang kita lihat, argumen pertama yang diteruskan ke view helper sesuai dengan nama file view di direktori resources / views. Argumen kedua adalah larik data yang harus tersedia untuk tampilan. Dalam hal ini, laravel meneruskan variabel nama yang ditampilkan dalam tampilan menggunakan sintaks Blade.

1. View di dalam direktori

Jika di dalam direktori `resources/views` terdapat direktori lagi untuk menyimpan *file* view, sebagai contoh `hello.blade.php` ada di dalam direktori `blog`, maka kita bisa menggunakan "dot" notation untuk mereferensikan direktori, sehingga syntax dalam route akan seperti berikut

```
Route::get('/hello', function () {
    return view('blog.hello', ['name' => 'Andi']);
});
```

2. Menampilkan view dari controller

View dapat dipanggil melalui Controller. Sehingga Routing akan memanggil Controller terlebih dahulu, dan Controller akan me-return view yang dimaksud

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

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class WelcomeController extends Controller
{
    public function hello() {
        return view('blog.hello', ['name' => 'Andi']);
    }
}
```

3. Meneruskan data ke view

Seperti yang anda lihat di contoh sebelumnya, anda dapat meneruskan data array ke view agar data tersebut tersedia untuk view

```
return view('hello', ['name' => 'Andi']);
```

Saat meneruskan informasi dengan cara ini, data harus berupa array dengan pasangan kunci / nilai. Setelah memberikan data ke view, anda kemudian dapat mengakses setiap nilai dalam view menggunakan kunci data seperti: `<?php echo $name; ?>` atau `{{ $name }}`.

Sebagai alternatif untuk meneruskan array data lengkap ke fungsi view helper, anda dapat menggunakan metode `with` untuk menambahkan bagian data individual ke view. Metode `with` mengembalikan instance view objek sehingga anda dapat melanjutkan rangkaian metode sebelum mengembalikan tampilan

```
return view('hello')
    ->with('name', 'Andi')
    ->with('occupation', 'Astronaut');
```

4. Berbagi data dengan semua view

Terkadang, anda mungkin perlu berbagi data dengan semua tampilan yang dirender oleh aplikasi anda. anda dapat melakukannya dengan menggunakan metode berbagi tampilan fasad. Biasanya, anda harus melakukan panggilan ke share methodi dalam metode boot penyedia layanan. anda bebas menambahkannya ke kelas `App\Providers\AppServiceProvider` atau membuat penyedia layanan terpisah untuk menampungnya

```
<?php
namespace App\Providers;

use Illuminate\Support\Facades\View;

class AppServiceProvider extends ServiceProvider
{
    /**
     * Register any application services.
     *
     * @return void
     */
    public function register()
    {
        //
    }
}
```

```

    }

    /**
     * Bootstrap any application services.
     *
     * @return void
     */
    public function boot()
    {
        View::share('key', 'value');
    }
}

```

B. Blade Layout, Section, dan Component

Dalam membuat suatu tampilan, seringkali ada beberapa bagian yang sama dan selalu ditampilkan di setiap halaman view. Bagian-bagian ini dapat dibuat template sehingga tidak perlu dibuat berulang kali di setiap halaman view.

1. Layout dan Section

Pada laravel, layout digunakan untuk membuat master view yang akan selalu ditampilkan oleh view-view child yang menggunakannya. Dalam sebuah layout kita bisa memberikan tempat-tempat yang bisa digunakan oleh child view. Tempat-tempat tersebut adalah section. Misalnya, dalam layout utama, kita definisikan section sidebar, main_content, dan footer. Selanjutnya, setiap child view yang menggunakan layout utama dapat menempatkan kode view di masing-masing section yang tersedia di layout utama. Pada layout, setiap kode html akan digunakan oleh child view. Sehingga child view tidak perlu mendefinisikan tag html, head, title, dll pada tiap-tiap view. Terdapat beberapa istilah yang digunakan untuk menerapkan layout.

@yield

@yield("nama_section") digunakan untuk mendefinisikan bagian dari layout yang akan digunakan dan diisi oleh child view.

@section

@section digunakan selain untuk mendefinisikan sebuah section, juga bisa untuk mengisi section yang diharapkan oleh parent view / layout melalui @yield. Diakhiri dengan @endsction.

@parent

Dalam child view kita bisa menampilkan juga konten yang ada pada parent dalam section tertentu, hal tersebut dilakukan dengan @parent.

@extends

Extends digunakan pada setiap child view yang ingin menggunakan sebuah view sebagai parent / layout.

Berikut adalah contoh dari sebuah layout dari master/parent view.

```
<!-- Disimpan di resources/views/layouts/app.blade.php -->
<html>
  <head>
    <title> Halaman @yield('title')</title>
  </head>
  <body>
    @section('sidebar')
      Ini adalah master sidebar.
    @show
    <div class="container">
      @yield('content')
    </div>
  </body>
</html>
```

Kode yang merupakan layout global, mendefinisikan title, sidebar, dan content yang dapat diisi oleh child view yang menggunakannya.

Berikut adalah contoh child view yang menggunakan layout app.blade.php.

```

<!--Disimpan di resources/views/child.blade.php -->
@extends('layouts.app')
@section('title', 'Profil')
@section('sidebar')
    @parent
    <p>Sidebar halaman Profil.</p>
@endsection
@section('content')
    <p>Ini adalah bagian konten. NIM - Nama</p>
@endsection

```

Keterangan: `@extends("layouts.app")` digunakan untuk menjadikan file view `app.blade.php` sebagai master view. Kemudian kita isi section title dengan "Profil" yang akan dirender sebagai `<title> Halaman Profil </title>`. Kemudian kita juga mengisi section sidebar menggunakan directive `@parent`. Sehingga ditampilkan konten sidebar parent/master serta menampilkan kalimat "Sidebar halaman Profil". Setelah itu kita juga isi section "content" dengan tulisan "Ini adalah bagian konten. NIM - Nama".

2. Component

Component berfungsi untuk membuat view yang dapat kita gunakan berulang kali. Berbeda dengan layout yang bertindak sebagai master, component dapat dianggap sebagai child view yang bisa kita pakai di view lain yang membutuhkannya. Misalnya dalam pengembangan sebuah aplikasi kita akan membutuhkan view untuk alert yang memberikan notifikasi kepada pengguna aplikasi terkait informasi, peringatan ataupun pesan error. Alert ini akan digunakan berulang kali di aplikasi. Oleh karena itu kita bisa membuatnya sebagai component yang bisa digunakan di view lainnya. Berikut adalah contoh component untuk membuat alert.

```

<!-- Disimpan di resources/views/components/alert.blade.php -->
<div class="alert alert-danger">
    {{ $slot }}
</div>

```

Di bawah ini adalah contoh view yang menerapkan component alert.

```
@extends('layouts.app')
// kode...
@component("alert")
<b>Tulisan ini akan mengisi variabel $slot</b>
@endcomponent
```

C. Bootstrap pada Laravel

Laravel menyediakan titik awal dasar menggunakan Bootstrap. Secara default, Laravel menggunakan NPM untuk menginstal paket frontend ini. Untuk menggunakannya, dapat dilakukan langkah-langkah berikut:

Dalam jobsheet ini instalasi bootstrap dilakukan dibagian praktikum. Berikut ini hanya pengetahuan cara instalasi umumnya.

1. Melakukan instalasi Node.js. Installer dapat diperoleh dari <https://nodejs.org/en/>. Sesuaikan dengan sistem operasi yang digunakan.
2. Untuk memastikan instalasi Node.js dan NPM berjalan lancar, Anda dapat memeriksanya dengan menjalankan dua perintah berikut melalui Command Prompt:

```
node -v
```

```
npm -v
```

Setelah Anda mengetikkan kedua perintah tersebut, Command Prompt akan menunjukkan versi Node.js dan NPM yang ter-install di komputer.

3. Kemudian melalui command prompt, ubah direktori ke dalam direktori project laravel yang telah dibuat sebelumnya.
4. Scaffolding Bootstrap yang disediakan oleh Laravel terletak di dalam paket Composer, yang dapat diinstal menggunakan Composer:laravel/ui.

```
composer require laravel/ui
```

5. Setelah paket diinstal, kita dapat menginstal scaffolding frontend menggunakan perintah Artisan:laravel/ui


```
php artisan ui bootstrap
```

```
php artisan ui bootstrap --auth
```

6. Sebelum mengompilasi CSS, instal dependensi frontend proyek Anda menggunakan Node package manager (NPM) :

```
npm install
```

7. Setelah paket diinstal, Anda dapat menggunakan perintah berikut untuk mengompilasi aset Anda .

```
npm run dev
```

Menggunakan Template Bootstrap di Laravel

Sebenarnya pada project laravel, sudah ada file css bootstrap secara default pada pertama kali kita menginstall laravel. Letaknya ada pada file app.css dalam folder css. Anda dapat langsung menggunakannya dengan menghubungkan file app.css tersebut dengan file view anda.

Untuk menghubungkan file css ke laravel anda dapat menggunakan syntax berikut:

```
<link rel="stylesheet" type="text/css" href="/css/style.css">
```

Atau

```
<link rel="stylesheet" type="text/css" href="{{ asset('/css/app.css') }}">
```

Dan file JSnya:

```
<script type="text/javascript" src="/js/app.js"></script>
```

Atau

```
<script type="text/javascript" src="{ asset('/js/app.js') }"></script>
```

Secara default linknya dimulai dari folder public. Jadi anda bisa meletakkan file css dan js didalam folder public.

Praktikum 1 – Integrasi Laravel dengan AdminLTE3

Cara paling cepat dan clean dapat dilakukan menggunakan repository dalam tautan berikut <https://github.com/jeroennoten/Laravel-AdminLTE/wiki/Installation>.

1. Dalam root folder project lakukan command berikut, untuk mendefinisikan requirement project.

```
composer require jeroennoten/laravel-adminlte
```

2. Melakukan instalasi requirement project di atas dengan command berikut:

```
php artisan adminlte:install
```

Perintah di atas akan meng-install:

- **AdminLTE distribution files** dan dependensinya (Bootstrap, jQuery, etc.) dalam folder public/vendor .
- Konfigurasi package di file config/adminlte.php
- Paket translasi di folder lang/vendor/adminlte/
- Dalam composer.json akan otomatis ditambahkan require untuk laravel-adminlte

```
{ composer.json > ...
1  {
2      "name": "laravel/laravel",
3      "type": "project",
4      "description": "The skeleton application for the Laravel framework.",
5      "keywords": ["laravel", "framework"],
6      "license": "MIT",
7      "require": {
8          "php": "^8.1",
9          "guzzlehttp/guzzle": "^7.2",
10         "jeroennoten/laravel-adminlte": "^3.9",
11         "laravel/framework": "^10.10",
12         "laravel/sanctum": "^3.3",
13         "laravel/tinker": "^2.8"
14     },
15     "require-dev": {
16         "fakerphp/faker": "^1.9.1",
17         "laravel/pint": "^1.0",
18         "laravel/sail": "^1.18",
19         "mockery/mockery": "^1.4.4",
20         "nunomaduro/collision": "^7.0",
21         "phpunit/phpunit": "^10.1",
22         "spatie/laravel-ignition": "^2.0"
23     },
24     "autoload": {
25         "psr-4": {
```

3. Buat file resources/views/layout/app.blade.php. Isi dengan kode berikut.

```
@extends('adminlte::page')

{{-- Extend and customize the browser title --}}

@section('title')
    {{ config('adminlte.title') }}
    @hasSection('subtitle') | @yield('subtitle') @endif
@stop

{{-- Extend and customize the page content header --}}

@section('content_header')
    @hasSection('content_header_title')
        <h1 class="text-muted">
            @yield('content_header_title')

            @hasSection('content_header_subtitle')
                <small class="text-dark">
                    <i class="fas fa-xs fa-angle-right text-muted"></i>
                    @yield('content_header_subtitle')
                </small>
            @endif
        </h1>
    @endif
@stop

{{-- Rename section content to content_body --}}

@section('content')
    @yield('content_body')
@stop

{{-- Create a common footer --}}

@section('footer')
    <div class="float-right">
        Version: {{ config('app.version', '1.0.0') }}
    </div>

    <strong>
```

```

        <a href="{{ config('app.company_url', '#') }}">
            {{ config('app.company_name', 'My company') }}
        </a>
    </strong>
@stop

{{-- Add common Javascript/Jquery code --}}

@push('js')
<script>

    $(document).ready(function() {
        // Add your common script logic here...
    });

</script>
@endpush

{{-- Add common CSS customizations --}}

@push('css')
<style type="text/css">

    {{-- You can add AdminLTE customizations here --}}
    /*
    .card-header {
        border-bottom: none;
    }
    .card-title {
        font-weight: 600;
    }
    */
</style>
@endpush

```

4. Edit `resources/views/welcome.blade.php`, kemudian replace seluruh kodenya dengan kode berikut

```
resources > views > welcome.blade.php > ...
1  @extends('layouts.app')
2
3  {{-- Customize layout sections --}}
4
5  @section('subtitle', 'Welcome')
6  @section('content_header_title', 'Home')
7  @section('content_header_subtitle', 'Welcome')
8
9  {{-- Content body: main page content --}}
10
11 @section('content_body')
12 |   <p>Welcome to this beautiful admin panel.</p>
13 @stop
14
15 {{-- Push extra CSS --}}
16
17 @push('css')
18 |   {{-- Add here extra stylesheets --}}
19 |   {{-- <link rel="stylesheet" href="/css/admin_custom.css"> --}}
20 @endpush
21
22 {{-- Push extra scripts --}}
23
24 @push('js')
25 |   <script> console.log("Hi, I'm using the Laravel-AdminLTE package!"); </script>
26 @endpush
```

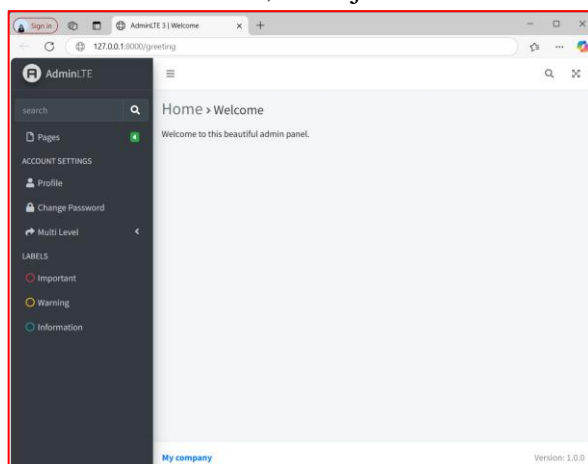
Untuk menggunakan blade template ini cukup dengan extend **AdminLTE layout** dengan cara `@extends('adminlte:page')`.

Template yields di beberapa bagian terklasifikasi dalam 2 yield:

- **main:** Biasa digunakan untuk extending the layout.
- **misc:** untuk kasus yang tidak biasa, atau hanya situasi tertentu.

Dokumentasi lebih detail terdapat di link berikut: <https://github.com/jeroennoten/Laravel-AdminLTE/wiki/Usage>

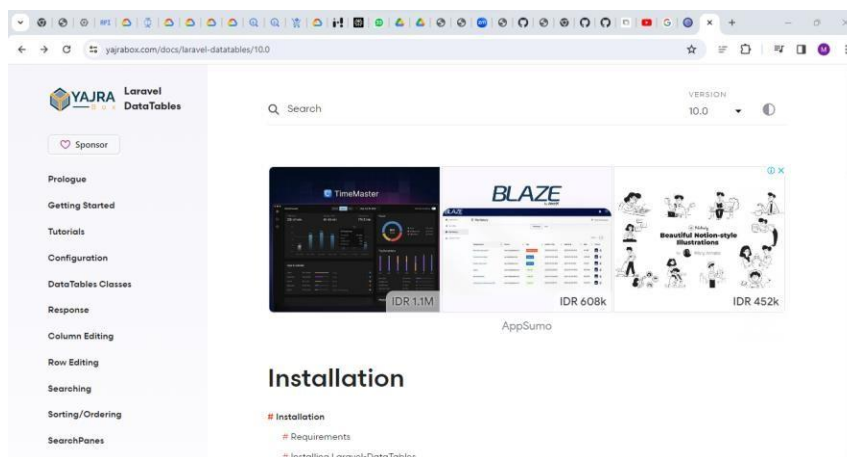
5. Kembali ke browser, menuju ke halaman awal.



Praktikum 2 – Integrasi dengan DataTables

Datatables biasa digunakan untuk menampilkan list data dengan banyak fitur seperti searching, pagination, sorting dan lain-lainnya. Dalam laravel biasa digunakan yajra datatables karena didesain agar sesuai dengan laravel.

<https://yajrabox.com/docs/laravel-datatables/10.0>



Berikut langkah-langkah praktikum

1. Install Laravel DataTables

`composer require laravel/ui --dev`

```
PS D:\LARAGON\laragon\www\PM1.2025\winggu5> composer require yajra/laravel-datatables:^10.0
The "10.0" constraint for "yajra/laravel-datatables" appears too strict and will likely not match what you want. See
https://getcomposer.org/constraints
./composer.json has been updated
Running composer update yajra/laravel-datatables
Loading composer repositories with package information
Updating dependencies
Lock file operations: 7 installs, 0 updates, 0 removals
- Locking league/fractal (0.20.2)
- Locking yajra/laravel-datatables (v10.0.0)
- Locking yajra/laravel-datatables-buttons (v10.0.9)
- Locking yajra/laravel-datatables-editor (v1.25.4)
- Locking yajra/laravel-datatables-fractal (v10.0.0)
- Locking yajra/laravel-datatables-html (v10.12.0)
- Locking yajra/laravel-datatables-oracle (v10.11.4)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 7 installs, 0 updates, 0 removals
- Downloading yajra/laravel-datatables-oracle (v10.11.4)
- Downloading yajra/laravel-datatables-html (v10.12.0)
- Downloading league/fractal (0.20.2)
- Downloading yajra/laravel-datatables-fractal (v10.0.0)
- Downloading yajra/laravel-datatables-editor (v1.25.4)
- Downloading yajra/laravel-datatables-buttons (v10.0.9)
8 package suggestions were added by new dependencies, use 'composer suggest' to see details.
Generating optimized autoload files
> illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

INFO Discovering packages.
jeroennoten/laravel-adminlte ... DONE
laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
laravel/ui ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
spatie/laravel-ignition ..... DONE
```

composer require yajra/laravel-datatables:^10.0

```
PS D:\LARAGON\laragon\www\PWL2025\Minggu5> composer require yajra/laravel-datatables:^10.0
The "10.0" constraint for "yajra/laravel-datatables" appears too strict and will likely not match what you want. See
https://getcomposer.org/constraints
./composer.json has been updated
Running composer update yajra/laravel-datatables
Loading composer repositories with package information
Updating dependencies
Lock file operations: 7 installs, 0 updates, 0 removals
- Locking league/fractal (0.20.2)
- Locking yajra/laravel-datatables (v10.0.0)
- Locking yajra/laravel-datatables-buttons (v10.0.9)
- Locking yajra/laravel-datatables-editor (v1.25.4)
- Locking yajra/laravel-datatables-fractal (v10.0.0)
- Locking yajra/laravel-datatables-html (v10.12.0)
- Locking yajra/laravel-datatables-oracle (v10.11.4)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 7 installs, 0 updates, 0 removals
- Downloading yajra/laravel-datatables-oracle (v10.11.4)
- Downloading yajra/laravel-datatables-html (v10.12.0)
- Downloading league/fractal (0.20.2)
- Downloading yajra/laravel-datatables-fractal (v10.0.0)
- Downloading yajra/laravel-datatables-editor (v1.25.4)
- Downloading yajra/laravel-datatables-buttons (v10.0.9)
8 package suggestions were added by new dependencies, use "composer suggest" to see details.
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi

 INFO  Discovering packages.

jeroennoten/laravel-adminlte ... DONE
laravel/sail ..... DONE
laravel/sanctum ..... DONE
laravel/tinker ..... DONE
laravel/ui ..... DONE
nesbot/carbon ..... DONE
nunomaduro/collision ..... DONE
nunomaduro/termwind ..... DONE
```

2. Pastikan nodejs sudah terinstall, dengan perintah.

Jika belum install nodejs package manager dari <https://nodejs.org/dist/v20.11.1/node-v20.11.1-x64.msi>

Npm digunakan untuk manajemen package javascript.

```
PS D:\LARAGON\laragon\www\PWL2025\Minggu5> node -v
v22.14.0
```

3. Install Laravel DataTables Vite dan sass

`npm i laravel-datatables-vite --save-dev`

```
C:\Users\LEGION>npm i laravel-datatables-vite --save-dev
added 11 packages in 9s

3 packages are looking for funding
  run `npm fund` for details
npm notice
npm notice New major version of npm available! 10.9.2 -> 11.2.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.2.0
npm notice To update run: npm install -g npm@11.2.0
npm notice
```

`npm install -D sass`

```
C:\Users\LEGION>npm install -D sass

added 17 packages, and audited 29 packages in 4s

8 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```


4. Edit file resources/js/app.js

```
resources > js > JS app.js
1  import './bootstrap';
2  import "../sass/app.scss";
3  import 'laravel-datatables-vite';
4
```

5. Buatlah file resources/sass/app.scss

```
resources > sass > app.scss
1  // Fonts
2  @import url('https://fonts.bunny.net/css?family=Nunito');
3
4
5
6  // Bootstrap
7  @import 'bootstrap/scss/bootstrap';
8
9  // DataTables
10 @import 'bootstrap-icons/font/bootstrap-icons.css';
11 @import 'datatables.net-bs5/css/dataTables.bootstrap5.min.css';
12 @import 'datatables.net-buttons-bs5/css/buttons.bootstrap5.min.css';
13 @import 'datatables.net-select-bs5/css/select.bootstrap5.css';
```

6. Jalankan dengan `npm run dev`

```
C:\laragon\www\pwl_pos>npm run dev
> dev
> vite

VITE v5.1.5 ready in 250 ms
  → Local:   http://localhost:5173/
  → Network: use --host to expose
  → press h + enter to show help

LARAVEL v10.46.0 plugin v1.0.2
  → APP_URL: http://localhost
```

7. Selanjutnya kita akan buat datatables untuk kategori
`php artisan datatables:make Kategori`

```
C:\laragon\www\pwl_pos>php artisan datatables:make Kategori

INFO DataTable [C:\laragon\www\pwl_pos\app\DataTables\KategoriDataTable.php] created successfully.
```

Hasil :

```
INFO DataTable [D:\LARAGON\laragon\www\pwl2025\Minggu5\app\DataTables\KategoriDataTable.php] created successfully.
```

8. Kita edit KategoriDataTable untuk mengatur kolom apasaja yang ingin ditampilkan

Sesuaikan dengan kodeprogram berikut

```
<?php
```

```
namespace App\DataTables;

use App\Models\KategoriModel;
use Illuminate\Database\Eloquent\Builder as QueryBuilder;
use Yajra\DataTables\EloquentDataTable;
use Yajra\DataTables\Html\Builder as HtmlBuilder;
use Yajra\DataTables\Html/Button;
use Yajra\DataTables\Html\Column;
use Yajra\DataTables\Html\Editor\Editor;
use Yajra\DataTables\Html\Editor\Fields;
use Yajra\DataTables\Services\DataTable;

class KategoriDataTable extends DataTable
{
    /**
     * Build the DataTable class.
     *
     * @param QueryBuilder $query Results from query() method.
     */
    public function dataTable(QueryBuilder $query): EloquentDataTable
    {
        return (new EloquentDataTable($query))
            ->addColumn('action', 'kategori.action')
            ->setRowId('id');
    }

    /**
     * Get the query source of dataTable.
     */
    public function query(KategoriModel $model): QueryBuilder
    {
        return $model->newQuery();
    }

    /**
     * Optional method if you want to use the html builder.
     */
    public function html(): HtmlBuilder
    {
        return $this->builder()
            ->setTableId('kategori-table')
```

```

        ->columns($this->getColumns())
        ->minifiedAjax()
        //->dom('Bfrtip')
        ->orderBy(1)
        ->selectStyleSingle()
        ->buttons([
            Button::make('excel'),
            Button::make('csv'),
            Button::make('pdf'),
            Button::make('print'),
            Button::make('reset'),
            Button::make('reload')
        ]);
    }

    /**
     * Get the dataTable columns definition.
     */
    public function getColumns(): array
    {
        return [
            /*          Column::computed('action')
                ->exportable(false)
                ->printable(false)
                ->width(60)
                ->addClass('text-center'), */
            Column::make('kategori_id'),
            Column::make('kategori_kode'),
            Column::make('kategori_nama'),
            Column::make('created_at'),
            Column::make('updated_at'),
        ];
    }

    /**
     * Get the filename for export.
     */
    protected function filename(): string
    {
        return 'Kategori_' . date('YmdHis');
    }
}

```

9. Ubah kategori model, sesuaikan seperti berikut

```
app > Models > KategoriModel.php > KategoriModel
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Database\Eloquent\Relations\HasMany;
7
8  class KategoriModel extends Model
9  {
10     protected $table = 'm_kategori';
11     protected $primaryKey = 'kategori_id';
12
13     protected $fillable = ['kategori_kode', 'kategori_nama'];
14
15     public function barang(): HasMany
16     {
17         return $this->hasMany(BarangModel::class, 'barang_id', 'barang_id');
18     }
19 }
```

10. Ubah Kategori Controller sesuaikan dengan skrip berikut:

```
app > Http > Controllers > KategoriController.php > KategoriController > index
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\DataTables\KategoriDataTable;
7
8  class KategoriController extends Controller
9  {
10     public function index(KategoriDataTable $dataTable)
11     {
12         return $dataTable->render('kategori.index');
13     }
14 }
```

11. Buat folder kategori di dalam resources/view, kemudian buat view blade index untuk kategori `resources/views/kategori/index.blade.php`

```
resources > views > kategori > index.blade.php > ...
1  @extends('layouts.app')
2
3  {{-- Customize layout sections --}}
4
5  @section('subtitle', 'Kategori')
6  @section('content_header_title', 'Home')
7  @section('content_header_subtitle', 'Kategori')
8
9  @section('content')
10     <div class="container">
11         <div class="card">
12             <div class="card-header">Manage Kategori</div>
13             <div class="card-body">
14                 {{ $dataTable->table() }}
15             </div>
16         </div>
17     </div>
18 @endsection
19
20 @push('scripts')
21     {{ $dataTable->scripts() }}
22 @endpush
23
24
```

12. Pastikan route untuk kategori sudah tersedia

```
24 Route::get('/kategori', [KategoriController::class, 'index']);
```

13. Sesuaikan app layout seperti kode berikut

```
@extends('adminlte::page')

{{-- Extend and customize the browser title --}}

@section('title')
    {{ config('adminlte.title') }}
    @hasSection('subtitle') | @yield('subtitle') @endif
@stop

@vite('resources/js/app.js')

{{-- Extend and customize the page content header --}}

@section('content_header')
    @hasSection('content_header_title')
        <h1 class="text-muted">
            @yield('content_header_title')

            @hasSection('content_header_subtitle')
```

```

        <small class="text-dark">
            <i class="fas fa-xs fa-angle-right text-muted"></i>
            @yield('content_header_subtitle')
        </small>
    @endif
</h1>
@endif
@stop

{{-- Rename section content to content_body --}}

@section('content')
    @yield('content_body')
@stop

{{-- Create a common footer --}}

@section('footer')
    <div class="float-right">
        Version: {{ config('app.version', '1.0.0') }}
    </div>

    <strong>
        <a href="{{ config('app.company_url', '#') }}">
            {{ config('app.company_name', 'My company') }}
        </a>
    </strong>
@stop

{{-- Add common Javascript/Jquery code --}}

@push('js')
<script src="https://cdn.datatables.net/2.0.2/js/dataTables.js"></script>

@endpush

@stack('scripts')

{{-- Add common CSS customizations --}}

@push('css')

```

```

<link                                                    rel="stylesheet"
href="https://cdn.datatables.net/2.0.2/css/dataTables.dataTables.css" />

<style type="text/css">



    {{-- You can add AdminLTE customizations here --}}
    /*
    .card-header {
        border-bottom: none;
    }
    .card-title {
        font-weight: 600;
    }
    */

</style>

@endpush

```

14. Menset ViteJs / script type defaults

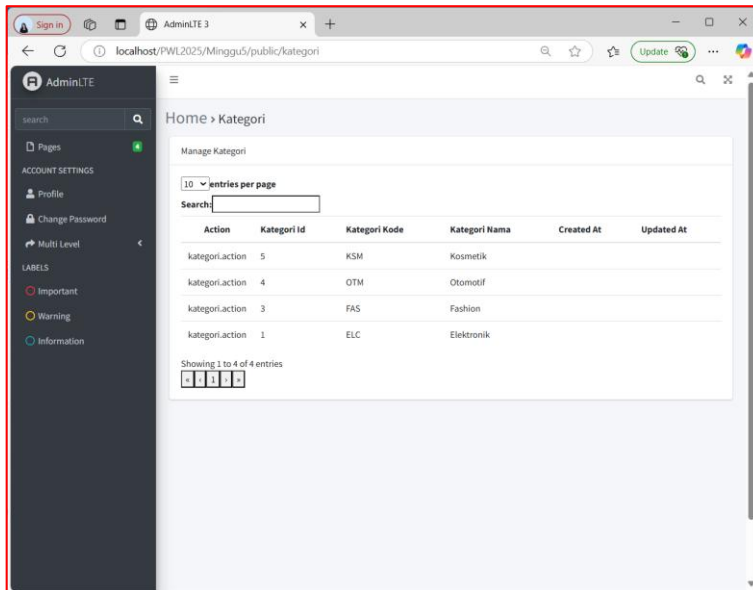
app > Providers >  AppServiceProvider.php >  AppServiceProvider >  boot

```

1  <?php
2
3  namespace App\Providers;
4
5  use Illuminate\Support\ServiceProvider;
6  use Yajra\DataTables\Html\Builder;
7
8  class AppServiceProvider extends ServiceProvider
9  {
10     /**
11      * Register any application services.
12      */
13     public function register(): void
14     {
15         //
16     }
17
18     /**
19      * Bootstrap any application services.
20      */
21     public function boot(): void
22     {
23         Builder::useVite();
24     }
25 }

```

Hasil di Browser :

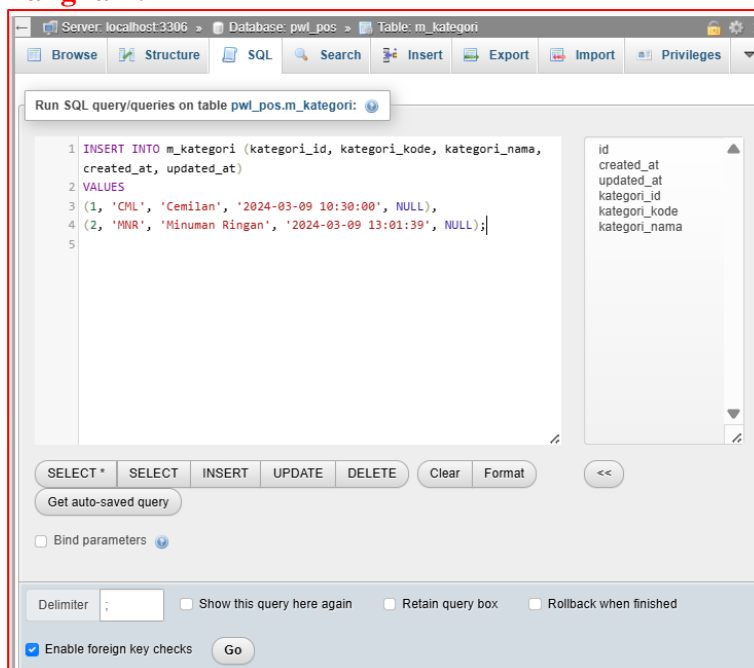


15. Isikan beberapa data ke table kategori

pwl_pos.m_kategori: 2 rows total (approximately)

kategori_id	kategori_kode	kategori_nama	created_at	updated_at
1	CML	Cemilan	2024-03-09 10:30:00	(NULL)
2	MNR	Minuman Ringan	2024-03-09 13:01:39	(NULL)

Langkah :



		id	created_at	updated_at	kategori_id	kategori_kode	kategori_nama
<input type="checkbox"/>				1	NULL	NULL	1 ELC Elektronik
<input type="checkbox"/>				3	NULL	NULL	3 FAS Fashion
<input type="checkbox"/>				4	NULL	NULL	4 OTM Otomotif
<input type="checkbox"/>				5	NULL	NULL	5 KSM Kosmetik
<input type="checkbox"/>				7	2024-03-09 10:30:00	NULL	1 CML Cemilan
<input type="checkbox"/>				8	2024-03-09 13:01:39	NULL	2 MNR Minuman Ringan

Hasil di adminLTE:

Home > Kategori

Manage Kategori

10 entries per page

Search:

Action	Kategori Id	Kategori Kode	Kategori Nama	Created At	Updated At
kategori.action	5	KSM	Kosmetik		
kategori.action	4	OTM	Otomotif		
kategori.action	3	FAS	Fashion		
kategori.action	2	MNR	Minuman Ringan	2024-03-09T13:01:39.000000Z	
kategori.action	1	ELC	Elektronik		
kategori.action	1	CML	Cemilan	2024-03-09T10:30:00.000000Z	

Showing 1 to 6 of 6 entries

16. Datatables sudah dapat di load di `/kategori`

Home > Kategori

Manage Kategori

Show 10 entries

Search:

Kategori Id	Kategori Kode	Kategori Nama	Created At	Updated At
1	CML	Cemilan	2024-03-09T10:30:00.000000Z	
2	MNR	Minuman Ringan	2024-03-09T13:01:39.000000Z	

Showing 1 to 2 of 2 entries

Previous 1 Next

Praktikum 3 – Membuat form kemudian menyimpan data dalam database

Langkah-langkahnya adalah sebagai berikut:

1. Menyesuaikan routing, tambahkan dua routing berikut

```
Route::get('/kategori/create', [KategoriController::class, 'create']);
Route::post('/kategori', [KategoriController::class, 'store']);
```

2. Tambahkan dua function berikut dalam KategoriController

```
app > Http > Controllers > KategoriController.php > KategoriController > store
1  <?php
2
3  namespace App\Http\Controllers;
4  use App\Models\KategoriModel;
5  use Illuminate\Http\Request;
6  use App\DataTables\KategoriDataTable;
7
8  class KategoriController extends Controller
9  {
10     public function index(KategoriDataTable $dataTable)
11     {
12         return $dataTable->render('kategori.index');
13     }
14
15     public function create()
16     {
17         return view('kategori.create');
18     }
19
20     public function store(Request $request)
21     {
22         KategoriModel::create([
23             'kategori_kode' => $request->kodeKategori,
24             'kategori_nama' => $request->namaKategori,
25
26         ]);
27         return redirect('/kategori');
28     }
29 }
```

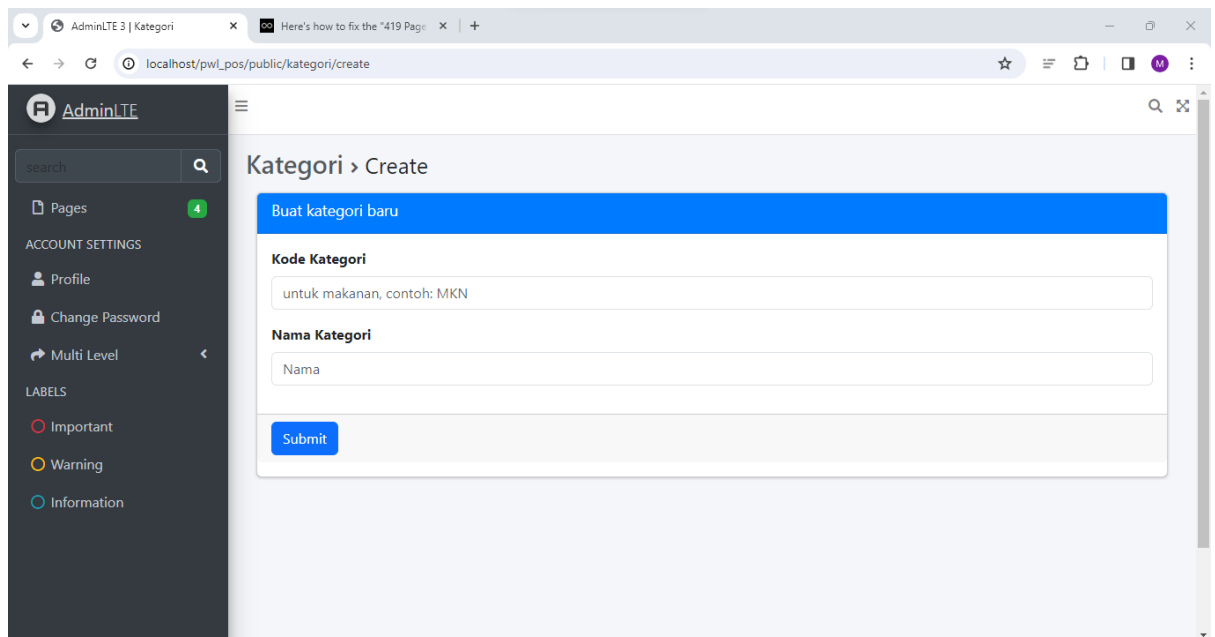
3. Dalam folder `views/kategori`, buatlah file dengan nama `create.blade.php`

```
resources > views > kategori > create.blade.php > ...
1  @extends('layouts.app')
2  {{-- Customize layout sections --}}
3  @section('subtitle', 'Kategori')
4  @section('content_header_title', 'Kategori')
5  @section('content_header_subtitle', 'Create')
6  {{-- Content body: main page content --}}
7  @section('content')
8      <div class="container">
9          <div class="card card-primary">
10             <div class="card-header">
11                 <h3 class="card-title">Buat kategori baru</h3>
12             </div>
13
14             <form method="post" action="../kategori">
15                 <div class="card-body">
16                     <div class="form-group">
17                         <label for="kodeKategori">Kode Kategori</label>
18                         <input type="text" class="form-control" id="kodeKategori" name="kodeKategori" placeholder="Kode Kategori" />
19                     </div>
20                     <div class="form-group">
21                         <label for="namaKategori">Nama Kategori</label>
22                         <input type="text" class="form-control" id="namaKategori" name="namaKategori" placeholder="Nama Kategori" />
23                     </div>
24                 </div>
25
26                 <div class="card-footer">
27                     <button type="submit" class="btn btn-primary">Submit</button>
28                 </div>
29             </form>
30         </div>
31     </div>
32 @endsection
```

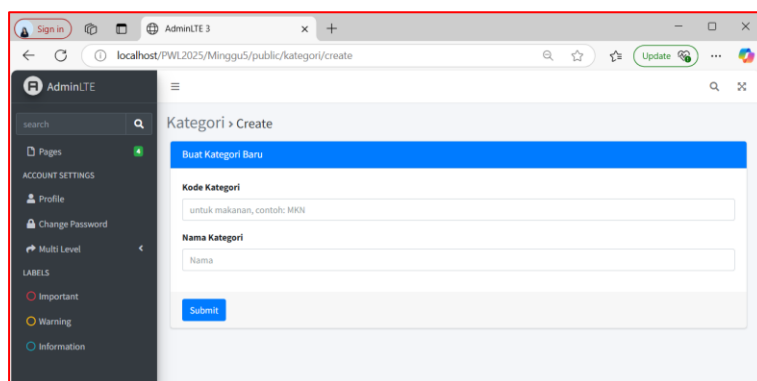
4. Kita lakukan pengecualian proteksi CsrfToken. Karena kita belum melakukan otentikasi . Kita edit dalam file berikut:

```
app > Http > Middleware > VerifyCsrfToken.php > VerifyCsrfToken > $except
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as Middleware;
6
7  class VerifyCsrfToken extends Middleware
8  {
9      /**
10       * The URIs that should be excluded from CSRF verification.
11       *
12       * @var array<int, string>
13       */
14       protected $except = [
15           '/kategori'
16       ];
17  }
18
```

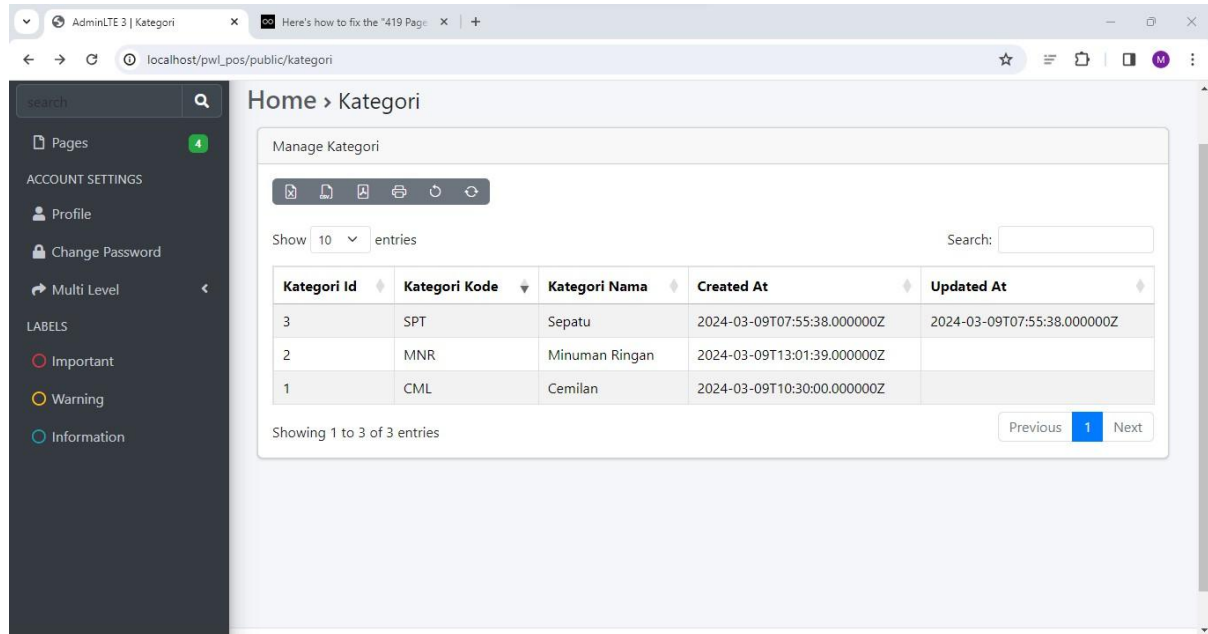
5. Akses kategori/create



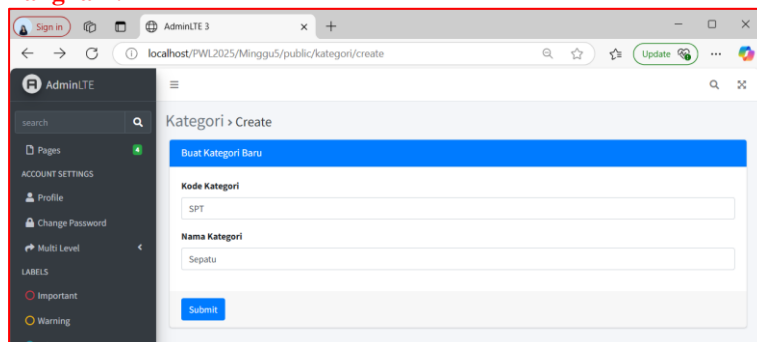
Hasil :



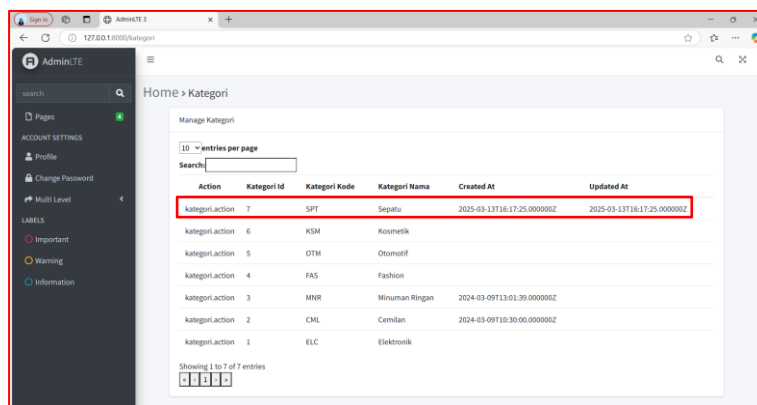
6. Halaman kategori



Langkah :



Hasil :



Tugas Praktikum

1. Tambahkan button Add di halam manage kategori, yang mengarah ke create kategori baru

Jawaban :

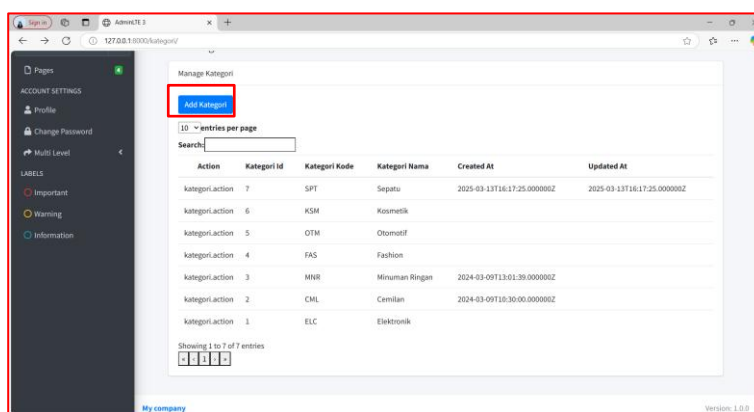
url('/kategori/create') → Mengarahkan ke route create() di KategoriController.

class="btn btn-primary mb-3" → Bootstrap styling untuk tombol.

```

1  @extends('layouts.app')
2
3  @section('subtitle', 'Kategori')
4  @section('content_header_title', 'Home')
5  @section('content_header_subtitle', 'Kategori')
6
7  @section('content')
8  <div class="container">
9      <div class="card">
10         <div class="card-header">Manage Kategori</div>
11         <div class="card-body">
12             <!-- menambahkan tombol Add di sini -->
13             <a href="{{ url('/kategori/create') }}" class="btn
14                 btn-primary mb-3">
15                 Add Kategori
16             </a>
17             <!-- menampilkan DataTables -->
18             {{ $dataTable->table() }}
19         </div>
20     </div>
21 </div>
22 @endsection
23
24 @push('scripts')
25     {{ $dataTable->scripts() }}
26 @endpush

```



2. Tambahkan menu untuk halaman manage kategori, di daftar menu navbar

Jawaban :

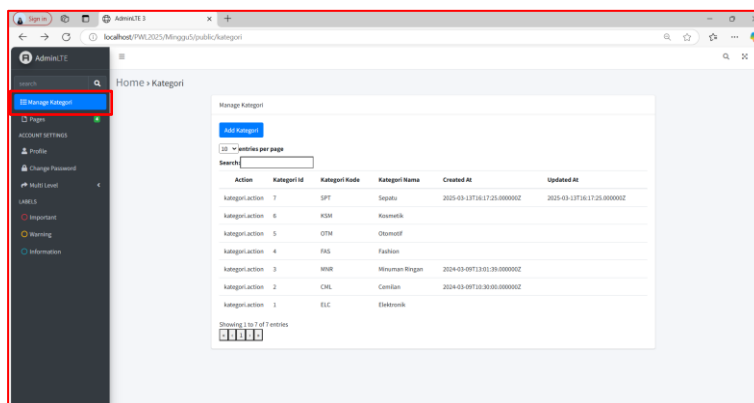
Dengan menambahkan di config/adminlte.php yang dimana mengelola menu sidebar dari file konfigurasi. Saya menambahkan menu "Manage Kategori" seperti ini:

```

config > adminlte.php
3   return [
301     'menu' => [
320         [
321             'text' => 'Manage Kategori',
322             'url' => '/kategori',
323             'icon' => 'fas fa-list',
324         ],
    ],
];

```

Kemudian tak lupa membersihkan cache dengan command (terminal vscode) : `php artisan config:clear`



3. Tambahkan action edit di datatables dan buat halaman edit serta controllernya

Jawaban :

Pertama, kita harus set kolom action di `datatables.php` dulu

```

app > DataTables > KategoriDataTable.php > KategoriDataTable > dataTable
16   class KategoriDataTable extends DataTable
23   public function dataTable($query)
25   {
26       return datatables()
27           ->eloquent($query)
28           ->addColumn('action', function ($row) {
29               return '<a href="'.route('kategori.edit', $row->id).
30                   '" class="btn btn-sm btn-warning">
31                   <i class="fas fa-edit"></i> Edit
32                   </a>';
33           });
34   }
35   ->rawColumns(['action']); // render html tombol edit

```


Tambahkan addColumn('action', ...) di Builder

```

web.php M KategoriController.php M KategoriDataTable.php U X index.blade.php U edit.blade.php U
app > DataTables > KategoriDataTable.php > KategoriDataTable > html
16 class KategoriDataTable extends DataTable
17 {
18     public function html(): HtmlBuilder
19     {
20         return $this->builder()
21             ->setTableId('kategori-table')
22             ->columns([
23                 ['data' => 'kategori_id', 'name' => 'kategori_id', 'title' => 'Kategori ID'], // Sesuai database
24                 ['data' => 'kategori_kode', 'name' => 'kategori_kode', 'title' => 'Kategori Kode'],
25                 ['data' => 'kategori_nama', 'name' => 'kategori_nama', 'title' => 'Kategori Nama'],
26                 ['data' => 'created_at', 'name' => 'created_at', 'title' => 'Created At'],
27                 ['data' => 'updated_at', 'name' => 'updated_at', 'title' => 'Updated At'],
28                 ['data' => 'action', 'name' => 'action', 'title' => 'Action', 'orderable' => false, 'searchable' => false],
29             ])
30             ->minifiedAjax()
31             ->dom('Bfritip')
32             ->orderBy(1)
33             ->selectStylesingle()
34             ->buttons([
35                 Button::make('excel'),
36                 Button::make('csv'),
37                 Button::make('pdf'),
38                 Button::make('print'),
39                 Button::make('reset'),
40                 Button::make('reload')
41             ]);
42     }
43 }

```

Tambahkan kolom action di index.blade.php

```

resources > views > kategori > index.blade.php > div.container > div.card > div.card-body > table#kategori-table
8 <div class="container">
9 <div class="card">
10 <div class="card-body">
11
12
13
14
15
16
17
18 <table class="table table-bordered" id="kategori-table">
19 <thead>
20 <tr>
21 <th>Kategori Id</th>
22 <th>Kategori Kode</th>
23 <th>Kategori Nama</th>
24 <th>Created At</th>
25 <th>Updated At</th>
26 <th>Action</th> <!-- Tambahkan kolom Action -->
27 </tr>
28 </thead>
29 </table>
30 </div>
31 </div>
32 </div>
33 @endsection

```

Agar bisa ter update maka diperlukan page edit, maka

```

app > Http > Controllers > KategoriController.php > KategoriController > update
12 class KategoriController extends Controller
13 {
14     public function edit($id)
15     {
16         $kategori = KategoriModel::findOrFail($id);
17         return view('kategori.edit', compact('kategori'));
18     }
19
20     public function update(Request $request, $id)
21     {
22         $request->validate([
23             'kategori_kode' => 'required|string|max:255',
24             'kategori_nama' => 'required|string|max:255',
25         ]);
26
27         $kategori = KategoriModel::findOrFail($id);
28         $kategori->update([
29             'kategori_kode' => $request->kategori_kode,
30             'kategori_nama' => $request->kategori_nama,
31         ]);
32
33         return redirect()->route('kategori.index')->with('success', 'Kategori berhasil diperbarui!');
34     }
35 }

```

```

Route::get('/kategori/edit/{id}', [KategoriController::class, 'edit']->name('kategori.edit'));
Route::put('/kategori/update/{id}', [KategoriController::class, 'update']->name('kategori.update'));

```

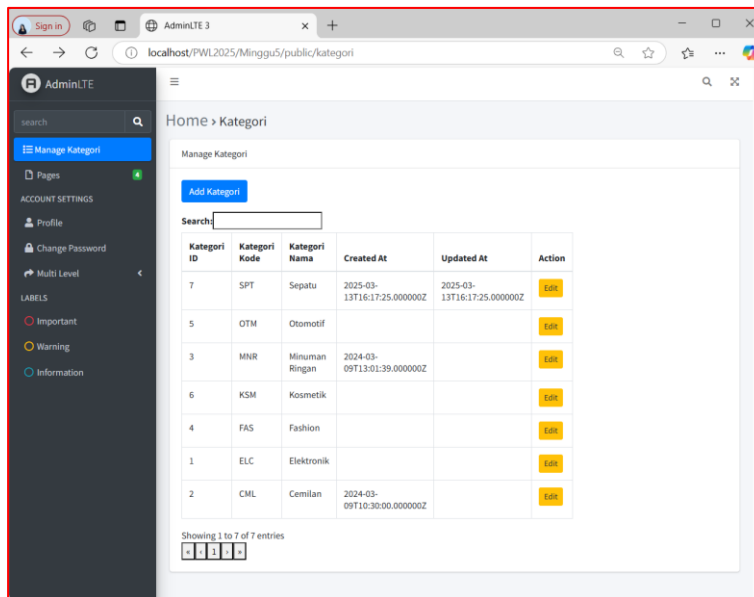


```

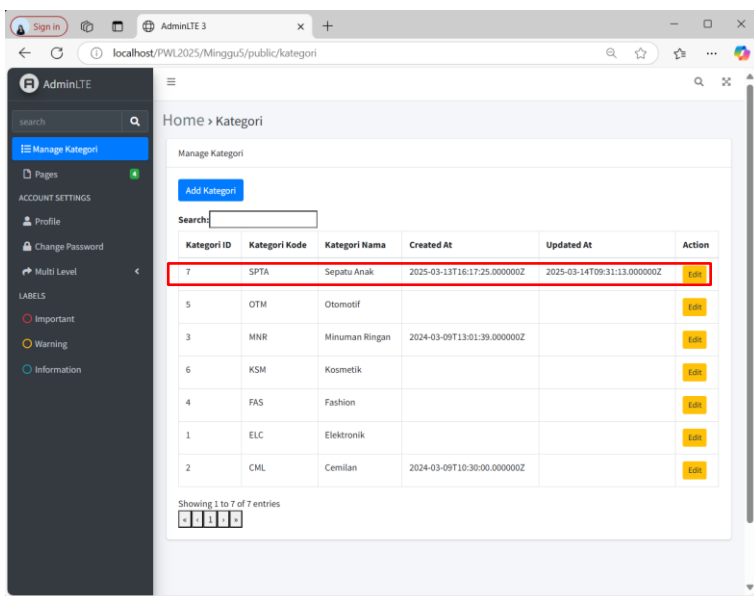
resources > views > kategori > edit.blade.php > @div.container
1 @extends('layouts.app')
2
3 @section('content')
4 <div class="container">
5 <h2>Edit Kategori</h2>
6
7 <form action="{{ route('kategori.update', $kategori->kategori_id) }}" method="POST">
8 @csrf
9 @method('PUT')
10
11 <div class="mb-3">
12 <label for="kategori_kode" class="form-label">Kategori Kode</label>
13 <input type="text" class="form-control" id="kategori_kode" name="kategori_kode" value="{{ $kategori->kategori_kode }}" required>
14 </div>
15
16 <div class="mb-3">
17 <label for="kategori_nama" class="form-label">Kategori Nama</label>
18 <input type="text" class="form-control" id="kategori_nama" name="kategori_nama" value="{{ $kategori->kategori_nama }}" required>
19 </div>
20
21 <button type="submit" class="btn btn-primary">Update</button>
22 <a href="{{ route('kategori.index') }}" class="btn btn-secondary">Cancel</a>
23 </form>
24 </div>
25 @endsection

```

Hasilnya :



Ketika ingin mengedit kategori sepatu menjadi sepatu anak maka akan ditampilkan sebagai berikut



		created_at	updated_at	kategori_id	kategori_kode	kategori_nama
<input type="checkbox"/>	Edit Copy Delete	NULL	NULL	1	ELC	Elektronik
<input type="checkbox"/>	Edit Copy Delete	2024-03-09 10:30:00	NULL	2	CML	Cemilan
<input type="checkbox"/>	Edit Copy Delete	2024-03-09 13:01:39	NULL	3	MNR	Minuman Ringan
<input type="checkbox"/>	Edit Copy Delete	NULL	NULL	4	FAS	Fashion
<input type="checkbox"/>	Edit Copy Delete	NULL	NULL	5	OTM	Otomotif
<input type="checkbox"/>	Edit Copy Delete	NULL	NULL	6	KSM	Kosmetik
<input type="checkbox"/>	Edit Copy Delete	2025-03-13 16:17:25	2025-03-14 09:31:13	7	SPTA	Sepatu Anak

4. Tambahkan action delete di datatables serta controllernya

Jawaban :

Tambahkan tombol Delete di dalam kolom action di KategoriDataTable.php

- tombol Delete dibuat dalam form agar bisa melakukan DELETE request.
- csrf_field() dan method_field("DELETE") digunakan untuk keamanan Laravel.
- Konfirmasi JavaScript akan muncul sebelum menghapus.

```

app > DataTables > KategoriDataTable.php > KategoriDataTable
16 class KategoriDataTable extends DataTable
17 {
18     public function dataTable($query)
19     {
20         return datatables()
21             ->eloquent($query)
22             ->addColumn('action', function ($row) {
23                 $editUrl = route('kategori.edit', ['id' =>
24                     $row->kategori_id]);
25                 $deleteUrl = route('kategori.destroy', ['id' =>
26                     $row->kategori_id]);
27
28                 return '
29                     <a href="'.$editUrl.'" class="btn btn-warning
30                         btn-sm">Edit</a>
31                     <form action="'.$deleteUrl.'" method="POST"
32                         style="display:inline;">
33                         '.csrf_field().'
34                         '.method_field("DELETE").'
35                         <button type="submit" class="btn btn-danger
36                             btn-sm" onclick="return confirm(\'Yakin ingin
37                                 menghapus kategori ini?\')">Delete</button>
38                     </form>
39                 ';
40             });
41     }

```

Di dalam routes/web.php, tambahkan route kategori.destroy sama seperti di datatables

```

Route::delete('/kategori/delete/{id}', [KategoriController::class,
'destroy'])->name('kategori.destroy');

```

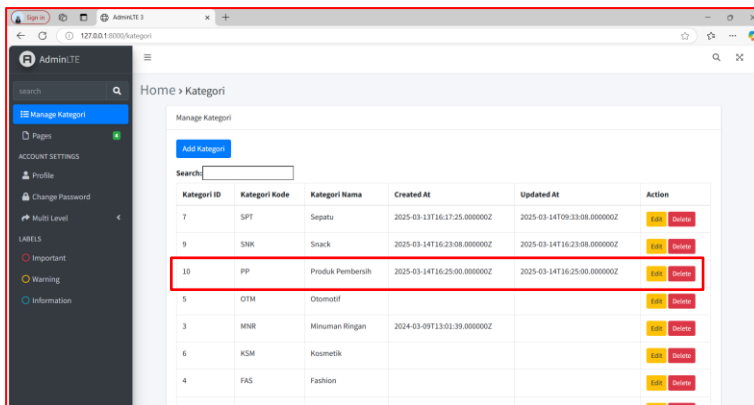
Di dalam KategoriController.php, tambahkan method destroy untuk menangani proses penghapusan data. findOrFail(\$id) akan mencari data berdasarkan ID, jika tidak ditemukan akan menampilkan error. \$kategori->delete(); menghapus data dari database.

Setelah berhasil, akan kembali ke halaman kategori dengan pesan sukses.

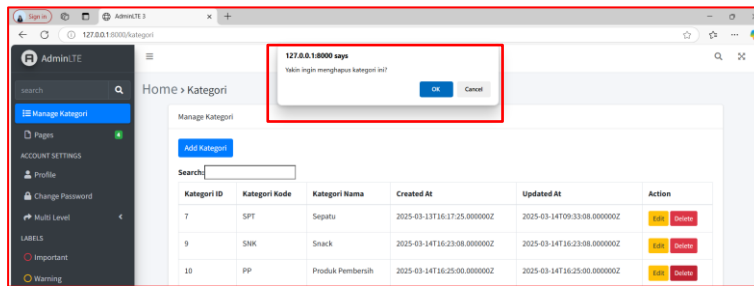
Kemudian, sebenarnya tanpa delete.blade.php juga bisa tapi saya disini bikin delete.blade.php untuk konfirmasi penghapusan data.

```
resources > views > kategori > delete.blade.php > ...
1  @extends('adminlte::page')
2
3  @section('title', 'Hapus Kategori')
4
5  @section('content_header')
6      <h1>Konfirmasi Hapus Kategori</h1>
7  @endsection
8
9  @section('content')
10     <div class="card">
11         <div class="card-body">
12             <p>Apakah kamu yakin ingin menghapus kategori <b>{{
13                 $kategori->kategori_nama }}</b>?</p>
14             <form action="{{ route('kategori.destroy',
15                 $kategori->id }}" method="POST">
16                 @csrf
17                 @method('DELETE')
18                 <button type="submit" class="btn btn-danger">Ya,
19                     Hapus</button>
20                 <a href="{{ route('kategori.index') }}" class="btn
21                     btn-secondary">Batal</a>
22             </form>
23         </div>
24     </div>
25 @endsection
```

Hasil :



	created_at	updated_at	kategori_id	kategori_kode	kategori_nama
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	NULL	NULL	1	ELC	Elektronik
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2024-03-09 10:30:00	NULL	2	CML	Cemilan
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2024-03-09 13:01:39	NULL	3	MNR	Minuman Ringan
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	NULL	NULL	4	FAS	Fashion
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	NULL	NULL	5	OTM	Otomotif
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	NULL	NULL	6	KSM	Kosmetik
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2025-03-13 16:17:25	2025-03-14 09:33:08	7	SPT	Sepatu
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2025-03-14 16:23:08	2025-03-14 16:23:08	9	SNK	Snack
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2025-03-14 17:05:30	2025-03-14 17:05:30	11	PP	Produk Pembersih



Kategori ID	Kategori Kode	Kategori Nama	Created At	Updated At	Action
7	SPT	Sepatu	2025-03-13T16:17:25.000000Z	2025-03-14T09:33:08.000000Z	Edit Delete
9	SNK	Snack	2025-03-14T16:23:08.000000Z	2025-03-14T16:23:08.000000Z	Edit Delete
10	PP	Produk Pembersih	2025-03-14T16:25:00.000000Z	2025-03-14T16:25:00.000000Z	Edit Delete

			created_at	updated_at	kategori_id	kategori_kode	kategori_nama
<input type="checkbox"/>	Edit	Copy	Delete	NULL	1	ELC	Elektronik
<input type="checkbox"/>	Edit	Copy	Delete	2024-03-09 10:30:00	2	CML	Cemilan
<input type="checkbox"/>	Edit	Copy	Delete	2024-03-09 13:01:39	3	MNR	Minuman Ringan
<input type="checkbox"/>	Edit	Copy	Delete	NULL	4	FAS	Fashion
<input type="checkbox"/>	Edit	Copy	Delete	NULL	5	OTM	Otomotif
<input type="checkbox"/>	Edit	Copy	Delete	NULL	6	KSM	Kosmetik
<input type="checkbox"/>	Edit	Copy	Delete	2025-03-13 16:17:25	7	SPT	Sepatu
<input type="checkbox"/>	Edit	Copy	Delete	2025-03-14 16:23:08	9	SNK	Snack

- Klik tombol Delete di DataTables.
- Muncul pop-up konfirmasi.
- Jika setuju, data akan terhapus dari database.

References

1. <https://github.com/jeroennoten/Laravel-AdminLTE/wiki/Usage>
2. <https://yajrabox.com/docs/laravel-datatables/10.0/quick-starter>