



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 method` 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`

```
PS C:\laragon\www\PWL2025\PWL_Week5\Js5\PWL_POS> composer require jeroennoten/laravel-adminlte
./composer.json has been updated
Running composer update jeroennoten/laravel-adminlte
Loading composer repositories with package information
Updating dependencies
```

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

`php artisan adminlte:install`

```
PS C:\laragon\www\PWL2025\PWL_Week5\Js5\PWL_POS> php artisan adminlte:install
AdminLTE assets files published successfully
Configuration file published successfully
Translation files published successfully
The installation is complete.
```

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
```

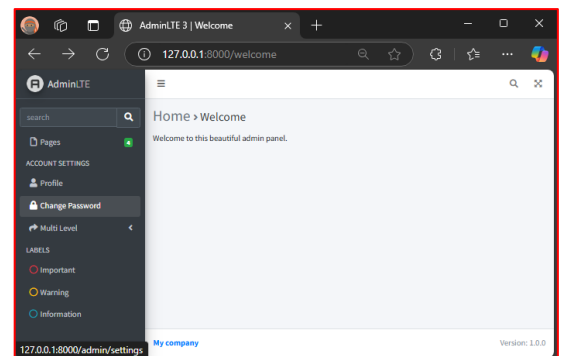
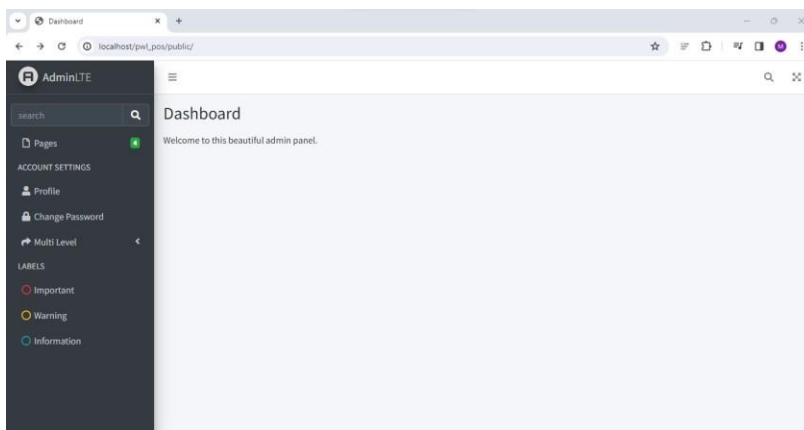
Untuk menggunakan blade template ini cukup dengan extend **AdminLTE layout** dengan cara `@extends('adminlte:page')`.

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.

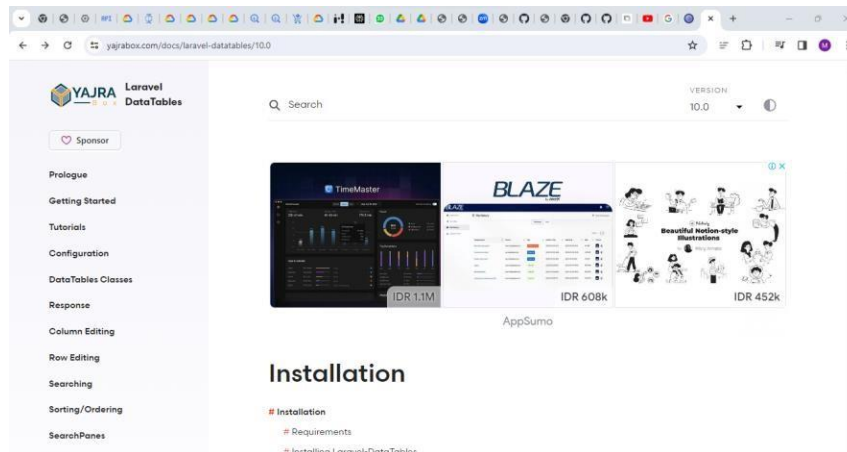




## Praktikum 2 – Integrasi dengan DataTables

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 DataTables1

`composer require laravel/ui --dev`

```
PS C:\laragon\www\PWL2025\PWL_Week5\Js5\PWL_POS> composer require laravel/ui --dev
./composer.json has been updated
Running composer update laravel/ui
```

`composer require yajra/laravel-datatables:^10.0`

```
PS C:\laragon\www\PWL2025\PWL_Week5\Js5\PWL_POS> 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
```

### 2. Pastikan nodejs sudah terinstall, dengan perintah `npm -v`.

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.

### 3. Install Laravel DataTables Vite dan sass

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

```
PS C:\laragon\www\PWL2025\PWL_Week5\Js5\PWL_POS> npm i laravel-datatables-vite --save-dev
added 47 packages, and audited 48 packages in 32s
```

`dev npm install -D sass`

```
PS C:\laragon\www\PWL2025\PWL_Week5\Js5\PWL_POS> npm install -D sass
[.....] | reify:@esbuild/aix-ppc64: timing reifyNode:node_modules/@esbuild/android-arm Complete
```



#### 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';
```

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

#### 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';
```

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

```
PS C:\laragon\www\PWL2025\PWL_Week5\Js5\PWL_POS> npm run dev
> dev
> vite

VITE v5.4.14 ready in 459 ms
→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help

LARAVEL v10.48.28 plugin v1.2.0
→ 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.
```

#### 8. Kita edit KategoriDatable untuk mengatur kolom apa saja 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\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 }
```



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)

16. Datatables sudah dapat di load di /kategori

Home > Kagegori

Manage Kategori

Show 10 entries

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

Home > Kategori

Manage Kategori

10 entries per page

Kategori Id	Kategori Kode	Kategori Nama	Created At	Updated At
5	KTG005	Pakaiian		
4	KTG004	Peralatan Rumah		
3	KTG003	Elektronik		
2	KTG002	Minuman		
1	KTG001	Makanan		

Showing 1 to 5 of 5 entries



### 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         return redirect('/kategori');  
27     }  
28 }  
29
```





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

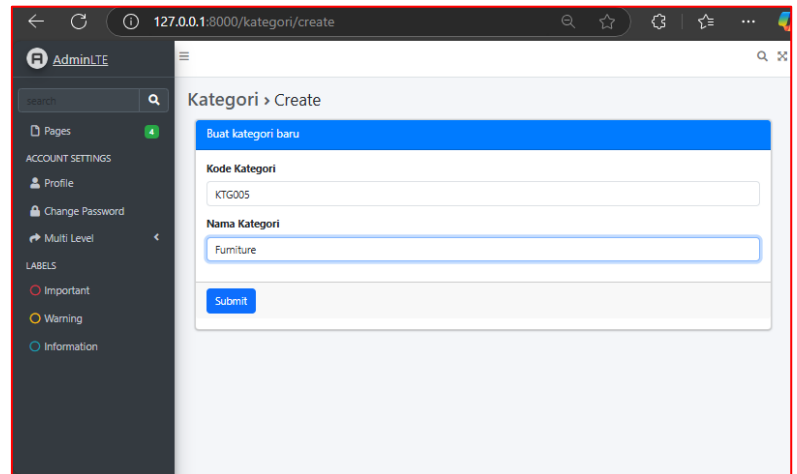
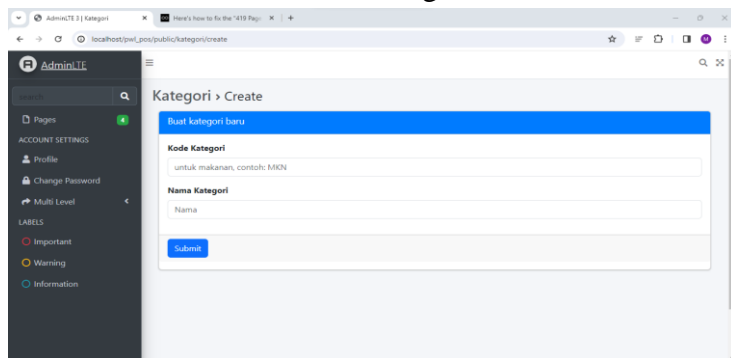
```
resources > views > kategori > create.blade.php > ...
1  @extends('layouts.app')
2  {{-- Customize layout sections --}}
3  @section('subtitle', 'Kategori')
4  @section('content_header_title', 'Kategori')
5  @section('content_header_subtitle', 'Create')
6  {{-- Content body: main page content --}}
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="">
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="">
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:

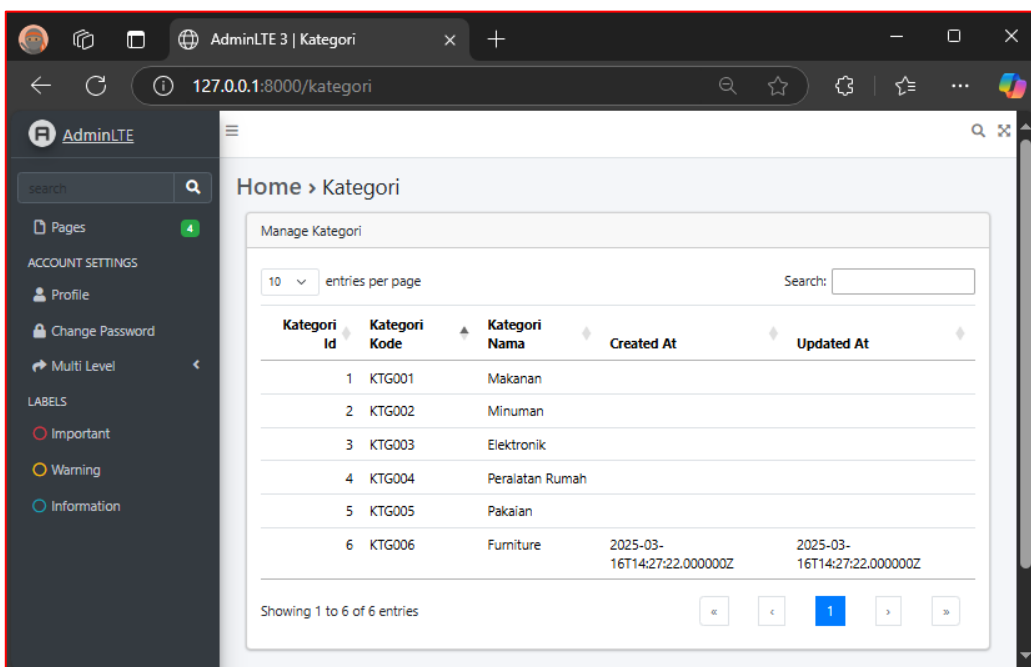
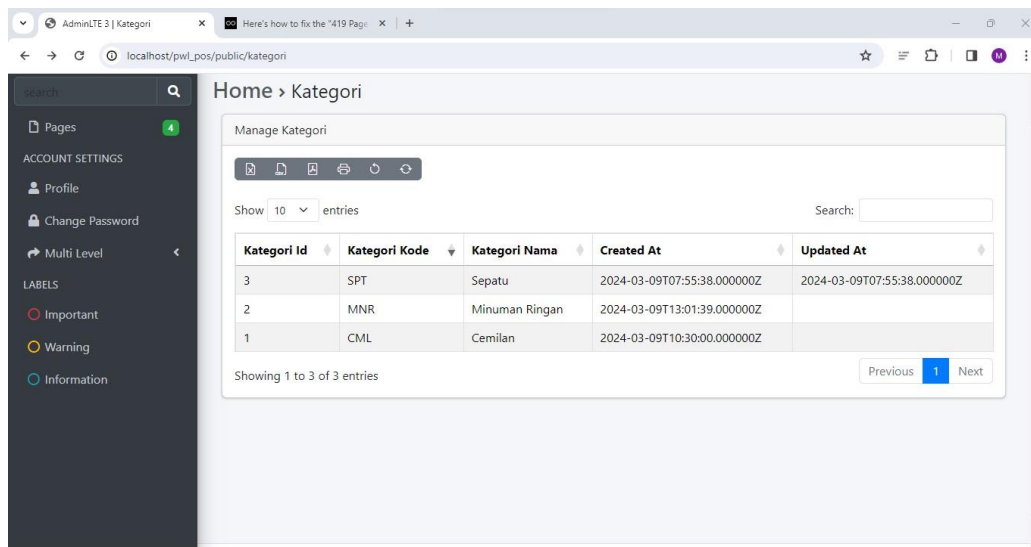
```
app > Http > Middleware > VerifyCsrfToken.php > VerifyCsrfToken > $except
1  <?php
2
3  namespace App\Http\Middleware;
4
5  use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as Middleware;
6
7  class VerifyCsrfToken extends Middleware
8  {
9      /**
10       * The URIs that should be excluded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



## 5. Akses kategori/create



## 6. Halaman kategori







## Tugas Praktikum

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

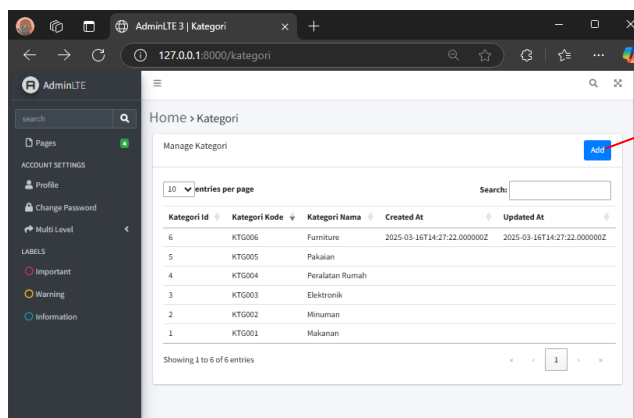
- Menambahkan kode program berikut

```
<div class="card-header">
  <h3 class="card-title">Manage Kategori</h3>
  <a href="{{ url(path: '/kategori/create') }}" class="btn btn-primary float-right">Add</a>
</div>
```

- Hingga menjadi berikut :

```
1 @extends(view: 'layouts.app')
2
3 {{-- Customize layout sections --}}
4 @section(section: 'subtitle', content: 'Kategori')
5 @section(section: 'content_header_title', content: 'Home')
6 @section(section: 'content_header_subtitle', content: 'Kategori')
7
8 @section(section: 'content')
9   <div class="container">
10     <div class="card">
11       <div class="card-header">
12         <h3 class="card-title">Manage Kategori</h3>
13         <a href="{{ url(path: '/kategori/create') }}" class="btn btn-primary float-right">Add</a>
14       </div>
15       <div class="card-body">
16         {{ $dataTable->table() }}
17       </div>
18     </div>
19   </div>
20 @endsection
21
22 @push('scripts')
23 {{ $dataTable->scripts() }}
24 @endpush
```

- Tampilan Web Index Kategori :



Kategori > Create

Buat kategori baru

Kode Kategori

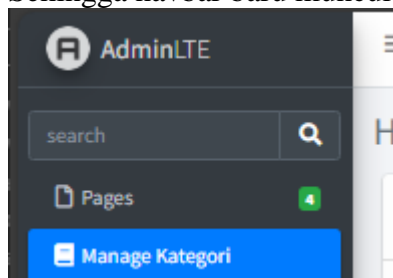
Nama Kategori

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

- Edit file **config/adminlte.php**, dengan menambahkan kode program berikut :

```
[
  'text' => 'Manage Kategori',
  'url' => 'kategori',
  'icon' => 'fas fa-fw fa-book',
],
```

- Sehingga navbar baru muncul :





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

➤ Update KategoriController

Edit **KategoriController.php**, tambahkan method untuk edit dan update data:

```
54 public function edit($id): Factory|View {  
55     $kategori = KategoriModel::findOrFail($id);  
56     return view('kategori.edit', data: compact('kategori'));  
57 }  
58  
59 0 references | 0 overrides | Codeium: Refactor | Explain | Generate Function Comment | X  
60 public function update(Request $request, $id): Redirector|RedirectResponse {  
61     $kategori = KategoriModel::findOrFail($id);  
62     $kategori->update(attributes: [  
63         'kategori_kode' => $request->kodeKategori,  
64         'kategori_nama' => $request->namaKategori,  
65     ]);  
66     return redirect(to: '/kategori');
```

➤ Tambahkan Route untuk Edit dan Update

Tambahkan route berikut di **web.php**:

```
33 Route::get(uri: '/kategori/{id}/edit', action: [KategoriController::class, 'edit']);  
34 Route::post(uri: '/kategori/{id}', action: [KategoriController::class, 'update']);
```

➤ Buat Halaman Edit

Buat file baru **resources/views/kategori/edit.blade.php**:

```
resources > views > kategori > editblade.php > ...  
1 @extends('layouts.app')  
2  
3 @section('subtitle', content: 'Edit Kategori')  
4 @section('content_header_title', content: 'Edit Kategori')  
5  
6 @section('content')  
7 <div class="container">
```

➤ Tambahkan Tombol Edit di DataTables

Edit **KategoriDataTable.php**, lalu tambahkan tombol edit di kolom aksi:

```
20 public function dataTable(QueryBuilder $query): EloquentDataTable  
21 {  
22     return (new EloquentDataTable(model: $query))  
23         ->addColumn(name: 'action', content: function ($row): string {  
24             return '<a href="'.url(path: "/kategori/".$row->kategori_id."/edit").'" class="btn btn-sm btn-warning">Edit</a>';  
25         })  
26         ->setRowId(content: 'id');  
27 }
```

➤ Menambahkan Tombol "Edit" di Halaman indexkategori.blade.php

Selain menambahkan tombol **Edit** di DataTables, kita juga harus memastikan tombol tersebut tampil di halaman **indexkategori**.

```
<td>{{ $k->created_at }}</td>  
<td>{{ $k->updated_at }}</td>  
<td>  
    <a href="{{ url(path: '/kategori/'.$k->kategori_id.'/edit') }}"  
        class="btn btn-sm btn-warning">Edit</a>  
</td>  
</tr>
```

```
public function getColumns(): array  
{  
    return [  
        Column::make(data: 'kategori_id'),  
        Column::make(data: 'kategori_kode'),  
        Column::make(data: 'kategori_nama'),  
        Column::make(data: 'created_at'),  
        Column::make(data: 'updated_at'),  
        Column::computed(data: 'action')  
            ->exportable(flag: false)  
            ->printable(flag: false)  
            ->width(value: 100)  
            ->addClass(class: 'text-center'),  
    ];  
}
```



## ➤ Hasil

Home > Kategori

Manage Kategori

10 entries per page

Search:

Kategori Id	Kategori Kode	Kategori Nama	Created At	Updated At	Action
6	KTG006	Furniture	2025-03-16T14:27:22.000000Z	2025-03-16T14:27:22.000000Z	Edit
5	KTG005	Pakaian			Edit
4	KTG004	Peralatan Rumah			Edit
3	KTG003	Elektronik			Edit
2	KTG002	Minuman			Edit
1	KTG001	Makanan			Edit

Showing 1 to 6 of 6 entries

Edit Kategori

Kode Kategori

KTG006

Nama Kategori

Furniture

Update

### Kode Kategori

KTG006

### Nama Kategori

Peralatan Masak

Update

Kategori Id	Kategori Kode	Kategori Nama	Created At	Updated At	Action
6	KTG006	Peralatan Masak	2025-03-16T14:27:22.000000Z	2025-03-18T07:25:29.000000Z	Edit
5	KTG005	Pakaian			Edit
4	KTG004	Peralatan Rumah			Edit
3	KTG003	Elektronik			Edit
2	KTG002	Minuman			Edit
1	KTG001	Makanan			Edit

## 4. Tambahkan action delete di datatables serta controllernya

### ➤ Update KategoriDataTable.php

Edit file app/DataTables/KategoriDataTable.php, perbarui bagian dataTable():

```
public function dataTable(QueryBuilder $query): EloquentDataTable
{
    return [new EloquentDataTable(model: $query)]
    ->addColumn(name: 'action', content: function ($row): string {
        return '
        <a href="'.url(path: '/kategori/'. $row->kategori_id.'/edit').'" class="btn btn-sm btn-warning mr-1">Edit</a>
        <button class="btn btn-sm btn-danger delete-btn" data-id="'. $row->kategori_id.'">Delete</button>
        ';
    })
    ->rawColumns(columns: ['action']) // Supaya tombol HTML dirender dengan benar
    ->setRowId(content: 'id');
}
```

### ➤ Update kategoriController.php

```
1 reference | 0 overrides | Codeium: Refactor | Explain | Generate Function Comment
public function delete($id): Redirector|RedirectResponse
{
    $find = KategoriModel::findOrFail(id: $id);
    $find->delete();

    return redirect(to: '/kategori');
}
```

### ➤ Update route di web.php

```
Route::post(uri: '/kategori/{id}', action: [KategoriController::class, 'update']);
Route::get(uri: '/kategori/delete/{id}', action: [KategoriController::class, 'delete']);
```

## ➤ Hasil :

10 entries per page

Search:

Kategori Id	Kategori Kode	Kategori Nama	Created At	Updated At	Action
9	KTG006	Barang	2025-03-18T07:49:10.000000Z	2025-03-18T07:49:10.000000Z	Edit Delete
5	KTG005	Pakaian			Edit Delete
4	KTG004	Peralatan Rumah			Edit Delete
3	KTG003	Elektronik			Edit Delete
2	KTG002	Minuman			Edit Delete
1	KTG001	Makanan			Edit Delete



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI  
**POLITEKNIK NEGERI MALANG**  
**JURUSAN TEKNOLOGI INFORMASI**  
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141  
Telp. (0341) 404424 – 404425, Fax (0341) 404420  
<http://www.polinema.ac.id>

---

## References

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