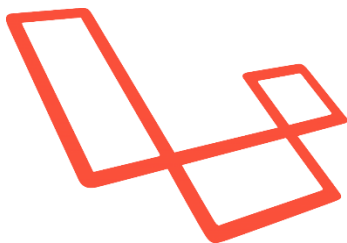




Modul 5 – Laravel MVC



HTML



CSS



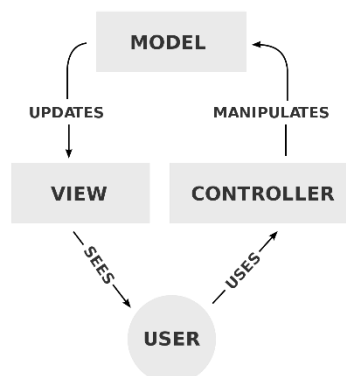
Requirement

1. Web Browser (Ex: Mozilla Firefox, Chrome, IE, dll)
2. Teks Editor (Ex: Visual Studio Code, Sublime Text, IntelliJ Idea, dll)
3. Web Server Apache (Ex: Xampp, Wamp)
4. Composer (<https://getcomposer.org/download/>)
5. Php Version Above 8.0

Penjelasan

Memahami Laravel MVC

Laravel merupakan salah satu framework PHP berkembang saat ini, dengan mendukung desain arsitektur *software* dengan menggunakan MVC atau *Model View Controller* dimana untuk memisahkan *logic* untuk manipulasi data, antarmuka pengguna dan kontrol aplikasi.




- **Model** : Pada *framework* PHP, model biasanya digunakan sebagai penghubung antara *controller* dengan *database* untuk mengambil data pada *database*. Hal ini merujuk pada konsep MVC di mana model digunakan sebagai representasi dari pengetahuan (*database*).
- **Controler** : bagian yang mengatur hubungan antara bagian model dan bagian view, *controller* berfungsi untuk menerima *request* dan data dari *user* kemudian menentukan apa yang akan diproses oleh aplikasi.
- **View** : Merupakan bagian yang menangani *presentation logic*. Pada suatu aplikasi web bagian ini biasanya berupa *file template* HTML, yang diatur oleh *controller*. View berfungsi untuk menerima dan merepresentasikan data kepada *user*. Bagian ini tidak memiliki akses langsung terhadap bagian model.

Instalasi Laravel MVC dan Membuat Project

1. Instalasi Composer

Sebelum melakukan instalasi composer pastikan mengunduh **XAMPP dengan versi php 8.0 keatas** karena laravel terbaru **hanya kompatibel dengan php 8.0 keatas**. Composer merupakan *tools dependency manager* di bahasa pemrograman PHP. Dengan menggunakan composer kita bisa melakukan instalasi atau menambahkan sebuah paket/*package* dengan lebih mudah.

Link Composer: <https://getcomposer.org/doc/00-intro.md#installation-windows>

 [Home](#) | [Getting Started](#) | [Download](#) | [Documentation](#) | [Browse Packages](#)

Download Composer Latest: v2.4.2

Windows Installer

The installer - which requires that you have PHP already installed - will download Composer for you and set up your PATH environment variable so you can simply call `composer` from any directory.

Download and run **Composer-Setup.exe** - it will install the latest composer version whenever it is executed.

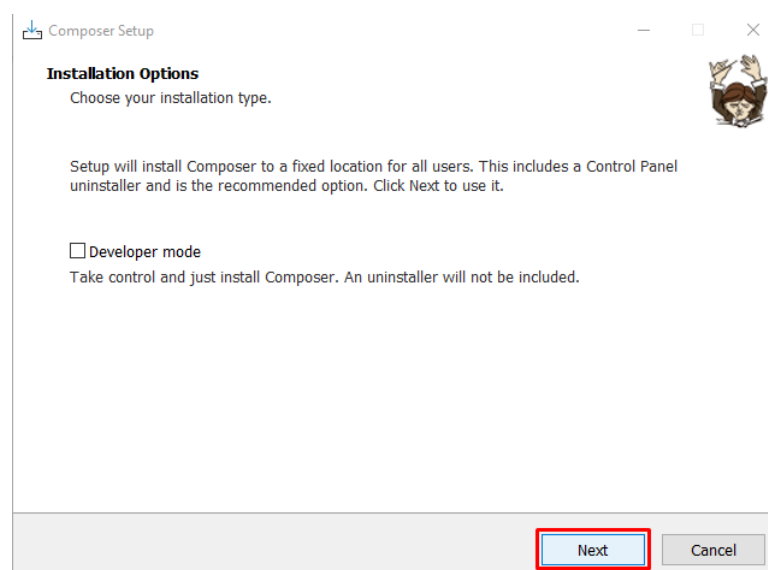
Command-line installation

[Click here to download Composer-Setup installer for Windows](#)

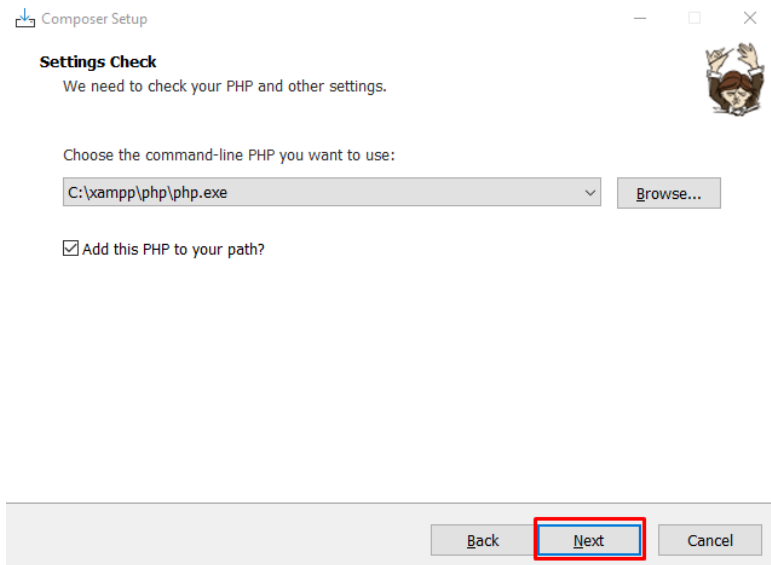
To quickly install Composer in the current directory, run the following script in your terminal. To automate the installation, use [the guide on installing Composer programmatically](#).

Gambar 1 Klik pada **kotak merah** untuk memulai mengunduh

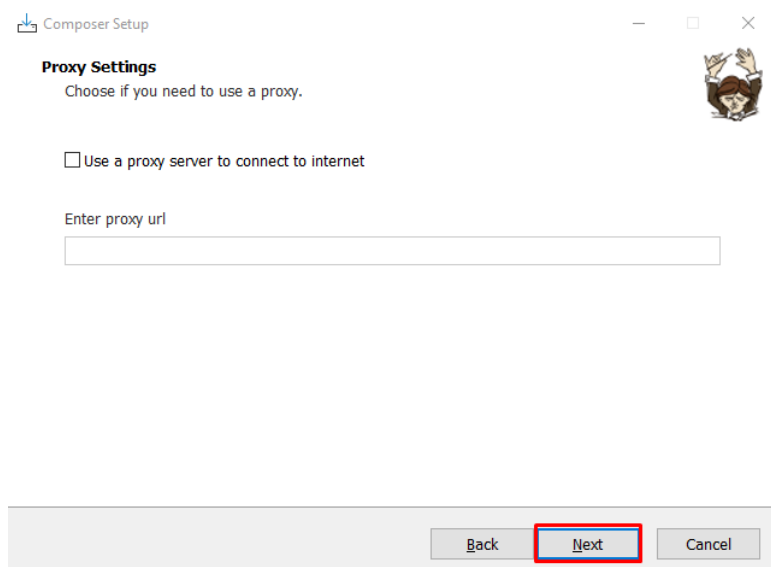
Selanjutnya, buka file **Composer-Setup.exe** untuk memulai instalasi. Ikuti langkah-langkah pada gambar berikut:



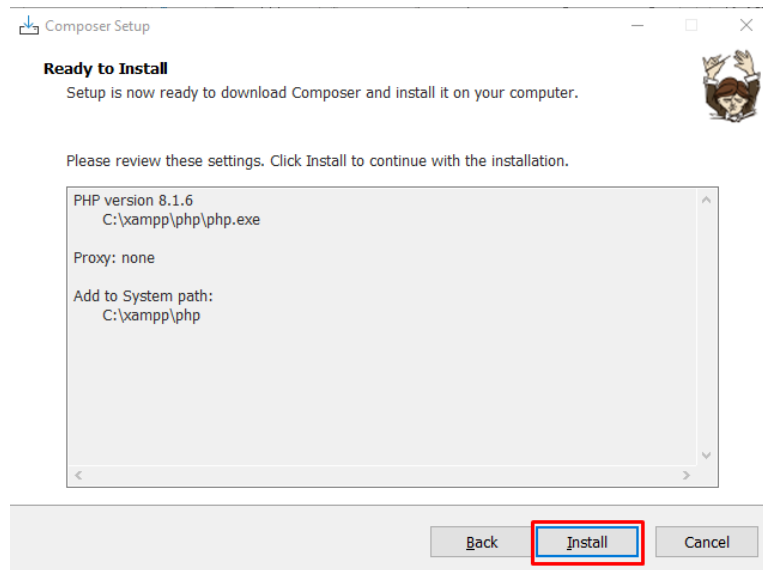
Gambar 2 Tekan tombol next **tanpa perlu** mencentang developer mode



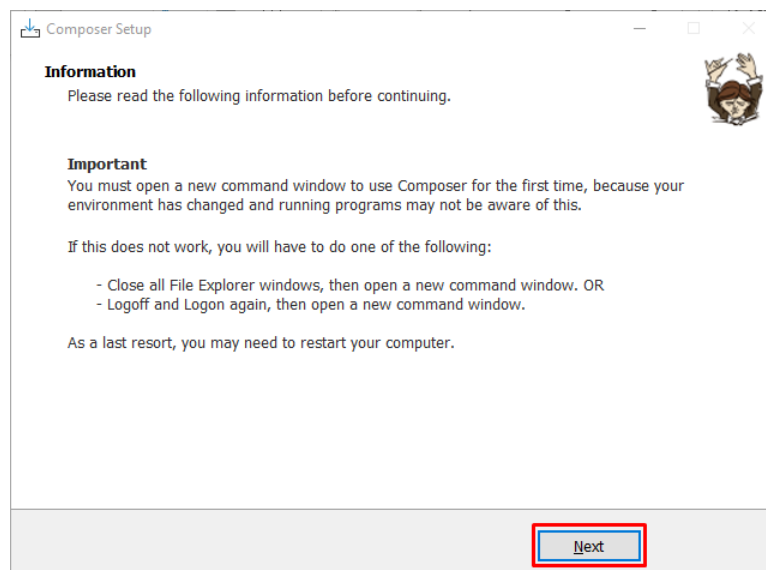
Gambar 3 Arahkan tempat instalasi pada **direktori php.exe** kemudian **centang** bagian “Add this PHP to your path”.



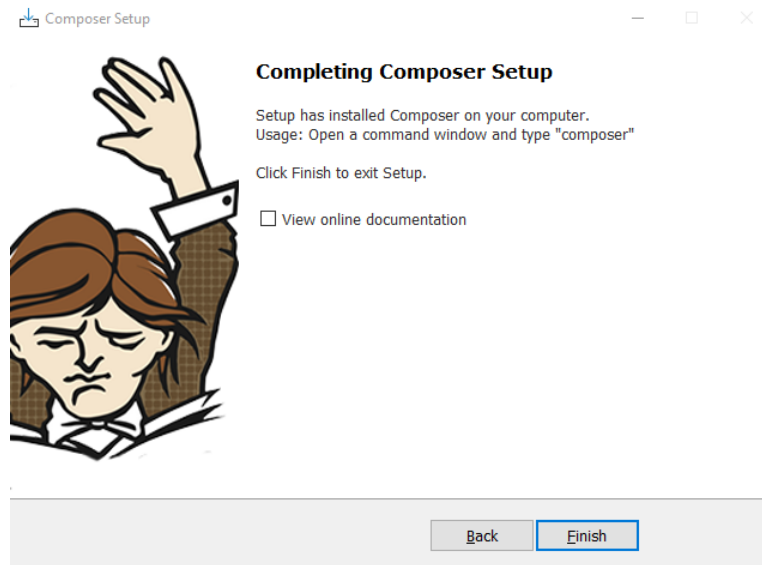
Gambar 4 Apabila instalasi meminta untuk menggunakan proxy, **tidak perlu dicentang** dan field **dibiarkan kosong**, setelah itu tekan tombol next.



Gambar 5 Pastikan kembali bahwa instalasi terarah pada **direktori php.exe**, jika sudah benar tekan tombol *next*.



Gambar 6 Tekan tombol *next*.



Gambar 7 Instalasi composer telah selesai dieksekusi, selanjutnya **tekan tombol *finish*** untuk menyelesaikan proses instalasi.

2. Membuat Project Laravel 9

Untuk membuat project Laravel 9 baru, kita bisa menggunakan perintah composer create-project. Langsung saja, silahkan teman-teman masuk ke dalam folder di mana akan menyimpan *project* Laravel-nya. Kemudian jalankan perintah berikut ini di dalam terminal/CMD :

```
composer create-project laravel/laravel:^9.0 contoh_project
```

Ganti **contoh_project** dengan format **UGD_X_YYYY**.

Perintah di atas digunakan untuk membuat *project* laravel baru dengan versi 9.x. Silakan tunggu proses instalasinya sampai selesai.

```
C:\Windows\System32\cmd.exe - composer create-project laravel/laravel:9.0 contoh_project
Microsoft Windows [Version 10.0.19044.1889]
(c) Microsoft Corporation. All rights reserved.

C:\XAMPP\htdocs> composer create-project laravel/laravel:9.0 contoh_project
Creating a "laravel/laravel:9.0" project at "../contoh_project"
Installing laravel/laravel (v9.0.0)
- Installing laravel/laravel (v9.0.0): Extracting archive
Created project in C:\XAMPP\htdocs\contoh_project
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 108 installs, 0 updates, 0 removals
- Locking asm89/stack-cors (v2.1.1)
- Locking brick/math (0.10.2)
- Locking dflydev/dot-access-data (v3.0.1)
- Locking doctrine/inflector (2.0.5)
- Locking doctrine/instantiator (1.4.1)
- Locking doctrine/lexer (1.2.3)
- Locking dragonmantank/cron-expression (v3.3.2)
- Locking egulias/email-validator (3.2.1)
- Locking fakerphp/faker (v1.20.0)
- Locking filp/whoops (2.14.5)
- Locking fruitcake/laravel-cors (v2.2.0)
- Locking fruitcake/php-cors (v1.2.0)
- Locking graham-campbell/result-type (v1.1.0)
- Locking guzzlehttp/guzzle (7.5.0)
- Locking guzzlehttp/promises (1.5.2)
- Locking guzzlehttp/psr7 (2.4.1)
- Locking hamcrest/hamcrest-php (v2.0.1)
- Locking laravel/framework (v9.51.0)
```

3. Menjalankan Project Laravel 9

Setelah berhasil melakukan proses instalasi Laravel 9, sekarang kita lanjutkan belajar bagaimana cara menjalankan *project* tersebut. Silakan jalankan perintah berikut ini di dalam terminal/CMD :

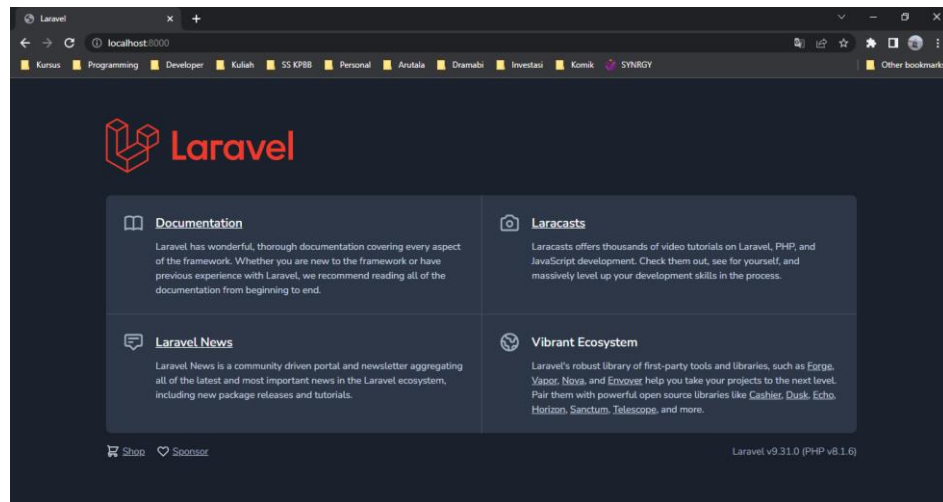
```
cd contoh_project
```

Ganti **contoh_project** dengan nama dari project yang sudah kalian buat.

Perintah **cd** di atas, digunakan untuk melakukan navigasi atau masuk ke dalam folder *project* **contoh_project**. Jika sudah masuk di dalam folder *project*, sekarang jalankan perintah berikut ini

```
php artisan serve
```

Jika berhasil, maka *project* kita akan dijalankan di dalam *localhost* menggunakan *port* 8000. Kita bisa membuaknya di dalam browser dengan `http://localhost:8000`.



Guided

1. Konfigurasi Koneksi Database

Karena akan bekerja menggunakan *database*, maka kita perlu melakukan **konfigurasi koneksi database-nya terlebih dahulu**. Silakan buka *project* kita menggunakan *Text Editor*, kemudian cari file yang bernama **.env**.

```
CONTOH_PROJECT  .env
> app
> bootstrap
> config
> database
> lang
> public
> resources
> routes
> storage
> tests
> vendor
> .editorconfig
> .env
> .env.example
> .gitattributes
> .gitignore
> .styleci.yml
> artisan
> composer.json
> composer.lock
> package.json
> phpunit.xml
> README.md
> webpack.mix.js

1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:zAZ0XSzWF5AMbcnIDNKK5eppX640yTPSYcb47aCQbc5A=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=laravel
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
```


Jika sudah ketemu, silakan **cari kode berikut ini** di dalam file `.env`.

```
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

Dan **ubahlah** menjadi seperti berikut ini agar sesuai dengan nama database yang nantinya akan dibuat di MYSQL :

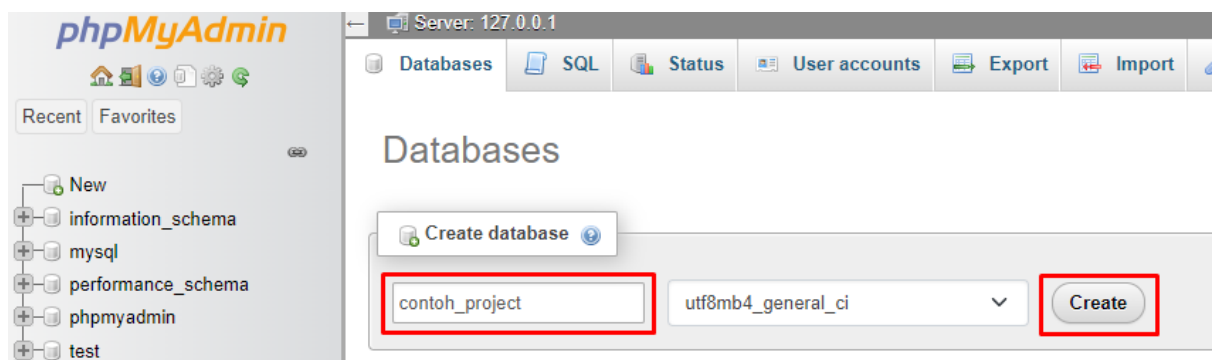
```
DB_DATABASE=contoh_project
DB_USERNAME=root
DB_PASSWORD=
```

Ganti `contoh_project` dengan format **UGD_X_YYYY**.

Di atas, kita atur untuk **DB_DATABASE** atau **nama database** yang akan kita gunakan. Kemudian untuk **DB_USERNAME** kita biarkan default, yaitu **root** dan untuk **DB_PASSWORD** silakan **disesuaikan dengan konfigurasi-nya masing-masing**. Jika menggunakan XAMPP, maka untuk *password*-nya dikosongkan saja (secara *default* kosong).

2. Membuat Database

Setelah kita berhasil melakukan konfigurasi koneksi *database*, sekarang kita lanjutkan untuk membuat *database*-nya. Silakan **buka phpmyadmin di xampp**. Kemudian **buat database baru** dengan nama dengan format **UGD_X_YYYY**. Kurang lebih seperti berikut ini :



Ganti `contoh_project` dengan format **UGD_X_YYYY**.

3. Membuat Model dan Migration

Migration adalah sebuah *version control database*, di mana akan membantu team untuk mengubah dan membagikan sebuah skema *database* dari aplikasi yang dibangun. Jika pada materi-materi sebelumnya kita membuat tabel-tabel secara manual di dalam *database*, maka dengan *migration* hal itu sudah tidak perlu dilakukan lagi.

Sekarang, kita lanjutkan belajar bagaimana cara membuat model dan *migration* di Laravel 9. Silakan jalankan perintah berikut ini di dalam terminal/CMD dan pastikan berada di dalam *project* laravel-nya

```
php artisan make:model Departemen -m
```

Perintah di atas, digunakan untuk **membuat model baru** dengan nama **Departemen** dan kita tambahkan *flag* **-m**, yang artinya *file migration*-nya juga akan ikut dibuat. Jika perintah di atas berhasil dijalankan, maka kita akan **mendapatkan 2 file baru**, yaitu :

- app/Models/Departemen.php
- database/migrations/2022_09_21_164205_create_departemens_table.php

Catatan: untuk nama file migration akan random sesuai tanggal pembuatannya.

4. Membuat Model dan Migration

Setelah berhasil membuat model dan *migration*, sekarang kita lanjutkan untuk **menambahkan field atau kolom di dalam file migration**. *field* yang kita tambahkan ini akan di *generate* di dalam table **departemen** yang ada di *database*.

Silahkan **buka file** `database/migrations/2022_09_21_164205_create_departemens_table.php`, kemudian pada **bagian function up**, ubah kode-nya menjadi seperti berikut ini :

```
public function up()
{
    Schema::create('departemens', function (Blueprint $table) {
        $table->id();
        $table->string('nama_departemen');
        $table->string('nama_manager');
        $table->integer('jumlah_pegawai');
        $table->timestamps();
    });
}
```

Dari perubahan kode di atas, kita menambahkan 3 *field* baru, yaitu :

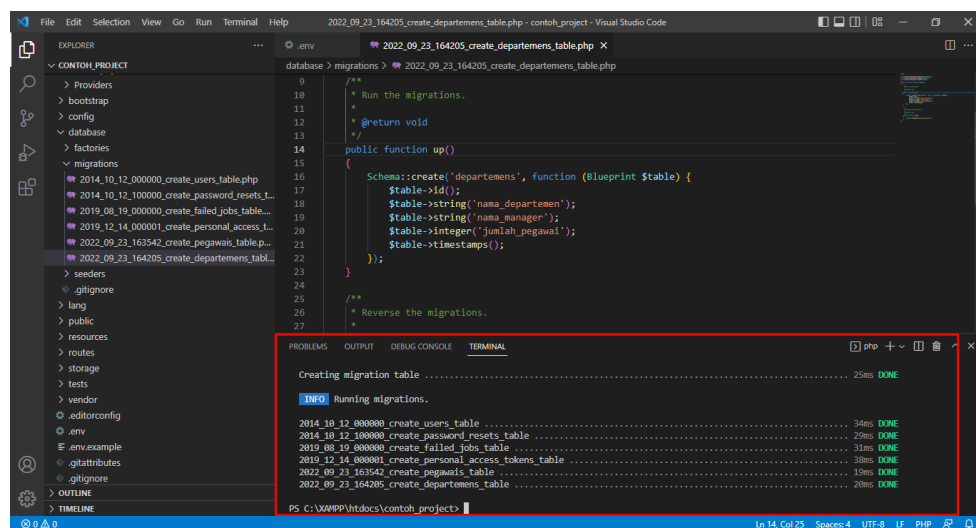
Field	Tipe Data
nama_Departemen	string
nama_manager	string
jumlah_pegawai	integer

5. Menjalankan *Migration*

Sekarang kita akan belajar menjalankan perintah *migrate*. Perintah ini akan digunakan untuk **melakukan proses *create table departemen*** beserta field yang ada di dalamnya ke dalam *database*.

```
php artisan migrate
```

Jika perintah di atas berhasil dijalankan, maka akan mendapatkan output seperti berikut ini :

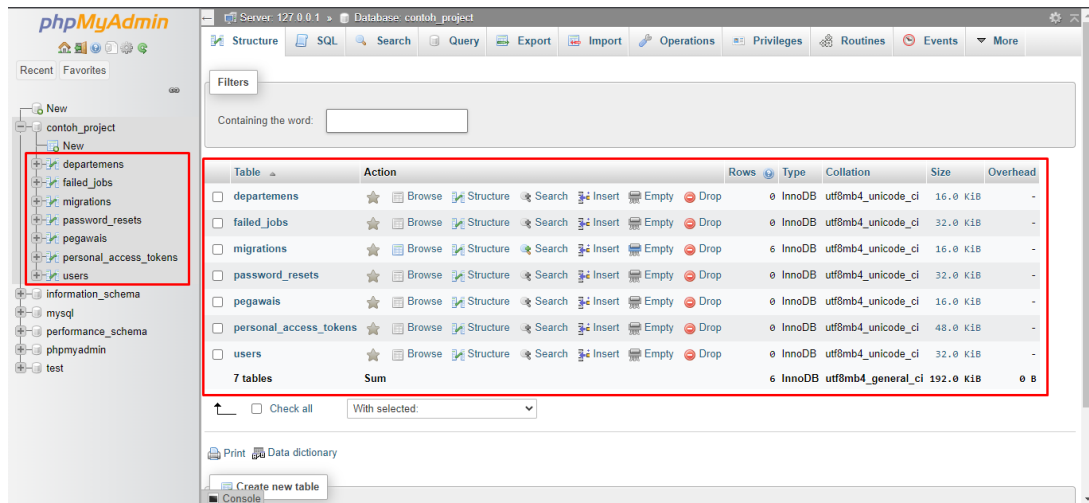


The screenshot shows a Visual Studio Code window with a file explorer on the left displaying a project structure. The main editor shows a migration file named `2022_09_23_164205_create_departemens_table.php`. The code defines a `Schema::create` method for a table named 'departemens' with columns: `id` (integer, primary key), `nama_departemen` (string), `nama_manager` (string), `jumlah_pegawai` (integer), and `timestamps`. Below the code, the terminal window shows the output of running `php artisan migrate`. The output indicates that the migration table was created successfully and the migration was run, with a list of previous migrations and their completion times.

```
2022_09_23_164205_create_departemens_table.php
9      /**
10     * Run the migrations.
11     *
12     * @return void
13     */
14     public function up()
15     {
16         Schema::create('departemens', function (Blueprint $table) {
17             $table->id();
18             $table->string('nama_departemen');
19             $table->string('nama_manager');
20             $table->integer('jumlah_pegawai');
21             $table->timestamps();
22         });
23     }
24     /**
25     * Reverse the migrations.
26     *
27     */
28     public function down()
29     {
30         Schema::dropIfExists('departemens');
31     }
32 }
```

```
Creating migration table ..... 25ms DONE
INFO Running migrations.
2014_10_12_000000_create_users_table ..... 34ms DONE
2014_10_12_100000_create_password_resets_table ..... 29ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 31ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 38ms DONE
2022_09_23_163542_create_pegawais_table ..... 19ms DONE
2022_09_23_164205_create_departemens_table ..... 20ms DONE
PS C:\xampp\htdocs\contoh_project>
```

Dan jika kita cek di dalam *database*, maka tabel-tabel beserta *field*-nya **juga otomatis ter-generate**.



6. Konfigurasi Mass Assignment

Mass Assignment di laravel memungkinkan kita untuk mengizinkan sebuah *field* dari tabel agar dapat menyimpan sebuah data. Karena tabel yang kita gunakan nantinya buat menyimpan data, maka kita perlu melakukan konfigurasi *Mass Assignment*-nya terlebih dahulu.

Silakan buka *file* `app/Models/Departemen.php`, kemudian **ubah kode-nya** menjadi seperti berikut ini :

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Departemen extends Model
{
    use HasFactory;

    /**
     * fillable
     *
     * @var array
     */
}
```

```
protected $fillable = [
    'nama_departemen',
    'nama_manager',
    'jumlah_pegawai',
];
}
```

Di atas, kita **menambahkan properti baru** yang bernama \$fillable. Properti tersebut yang dinamakan mass assignment di laravel, dimana di dalamnya kita menambahkan *field-field* yang diizinkan untuk melakukan manipulasi data.

7. Membuat *Controller Departemen*

Pertama, kita akan **membuat controller** terlebih dahulu. Dan untuk teman-teman semuanya, jika membuat *controller* **pastikan menggunakan kata tunggal atau singular**. Karena **Best Practice** dalam pembuatan *controller* menggunakan kata tunggal/singular. Silahkan jalankan perintah berikut ini di dalam terminal/CMD :

```
php artisan make:controller DepartemenController
```

Jika perintah di atas berhasil dijalankan, maka kita akan **mendapatkan 1 file controller baru** dengan nama **DepartemenController.php** yang berada di dalam folder **app/Http/Controllers**. Silahkan buka *file* tersebut dan **ubah kode-nya** menjadi seperti berikut ini:

```
<?php

namespace App\Http\Controllers;

/* Import Model */
use App\Models\Departemen;
use Illuminate\Http\Request;

class DepartemenController extends Controller
{
    /**
     * index
     *
     * @return void
     */
    public function index()
    {
```

```

        //get posts
        $departemen = Departemen::get();

        //render view with posts
        return view('departemen.index', compact('departemen'));
    }
}

```

Dari perubahan kode di atas, pertama kita **import model departemen** terlebih dahulu. Setelah itu, kita **buat 1 method** baru dengan nama **index**. Di dalam method index, pertama kita **get** data **departemen** dari *database* melalui model Departemen. Setelah itu, kita **return** ke dalam sebuah *view* dengan nama **index.blade.php** yang berada di dalam folder **resources/views/departemen**. Jika kita lihat memang belum ada, karena kita akan membuatnya nanti. Dan kita juga **mengirimkan data departemen** tersebut ke dalam *view* melalui *method* bawaan dari PHP, yaitu *compact*.

8. Menambahkan Route

Sekarang kita akan lanjutkan menambahkan sebuah *route* di Laravel 9. Pada project ini kita akan **menggunakan route dengan type resource**, yang artinya *route* tersebut akan berisi beberapa *route-route* untuk kebutuhan CRUD. Seperti *index*, *create*, *store*, *show*, *edit*, *update* dan *destroy*. Ini akan menghemat waktu kita dibandingkan membuatnya secara manual satu persatu. Sekarang, **silakan buka file routes/web.php**, kemudian **ubah keseluruhan kode-nya** menjadi seperti berikut ini:

```

<?php

use Illuminate\Support\Facades\Route;

/*
|-----
| Web Routes
|-----
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider within a group which
| contains the "web" middleware group. Now create something great!
|
*/

Route::get('/', function () {

```

```

        return view('dashboard'); /* arahkan ke halaman dashboard */
    });

//Route Resource
Route::resource('/departemen',
\App\Http\Controllers\DepartemenController::class);

```

Di atas, kita **menambahkan route baru** dengan jenis *resource*. Untuk memastikan apakah *route* yang kita tambahkan sudah berhasil, kita **bisa menjalankan perintah berikut** ini di dalam terminal/CMD :

php artisan route:list

```

PS C:\XAMPP\htdocs\contoh_project> php artisan route:list

GET|HEAD / ..... ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
POST _ignition/execute-solution ..... ignition.executeSolution > Spatie\LaravelIgnition > ExecuteSolutionController
GET|HEAD _ignition/health-check ..... ignition.healthCheck > Spatie\LaravelIgnition > HealthCheckController
POST _ignition/update-config ..... ignition.updateConfig > Spatie\LaravelIgnition > UpdateConfigController
GET|HEAD api/user .....
GET|HEAD departemen ..... departemen.index > DepartemenController@index
POST departemen ..... departemen.store > DepartemenController@store
GET|HEAD departemen/create ..... departemen.create > DepartemenController@create
GET|HEAD departemen/{departeman} ..... departemen.show > DepartemenController@show
PUT|PATCH departemen/{departeman} ..... departemen.update > DepartemenController@update
DELETE departemen/{departeman} ..... departemen.destroy > DepartemenController@destroy
GET|HEAD departemen/{departeman}/edit ..... departemen.edit > DepartemenController@edit
GET|HEAD sanctum/csrf-cookie ..... Laravel\Sanctum > CsrfCookieController@show

Showing [13] routes

PS C:\XAMPP\htdocs\contoh_project>

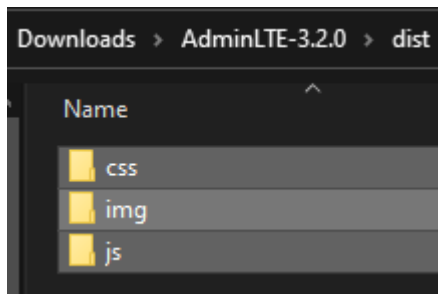
```

Gambar 8 Route hasil dari *resource*

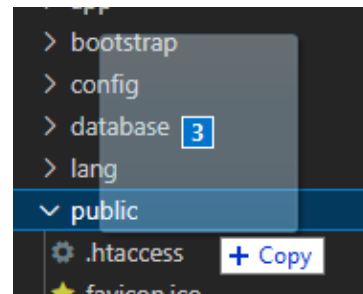
9. Membuat View dan Menampilkan Data

Untuk tampilan *dashboard*, kita akan gunakan *template* dari AdminLTE versi 3.2.0 agar proses pembuatan tampilan halaman lebih efisien. Silakan **download** asetnya melalui *link* berikut : <https://github.com/ColorlibHQ/AdminLTE/archive/refs/tags/v3.2.0.zip>

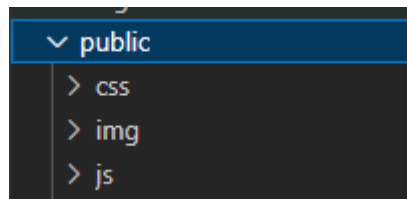
Extract file yang sudah terdownload kemudian **salin** seluruh folder yang ada di folder AdminLTE-3.2.0\dist ke dalam folder *project* dengan nama **public**.



Gambar 9 Drag semua folder *template* dari folder **dist**

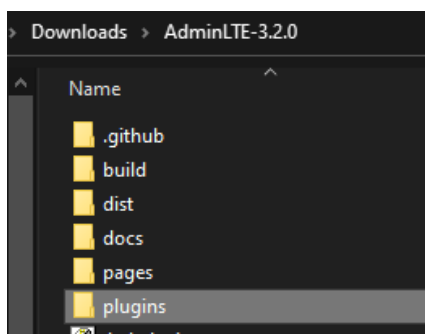


Gambar 10 Drop ke folder **public** yang ada pada *project*

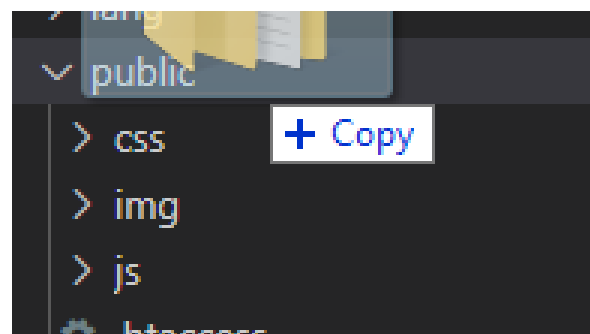


Gambar 11 Folder **css**, **img**, **js** berhasil disalin

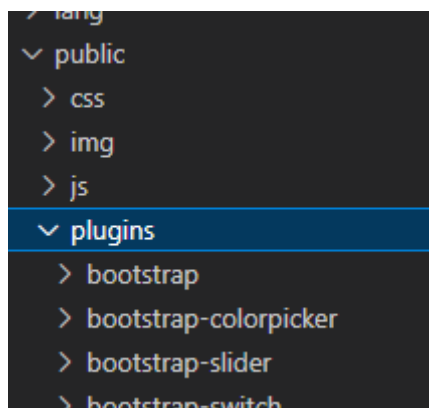
Salin juga folder **AdminLTE-3.2.0\plugins** ke dalam folder **public** pada *project*. Pastikan bahwa semua aset **berhasil tersalin** seperti pada gambar.



Gambar 12 Drag folder *plugins* dari *template*

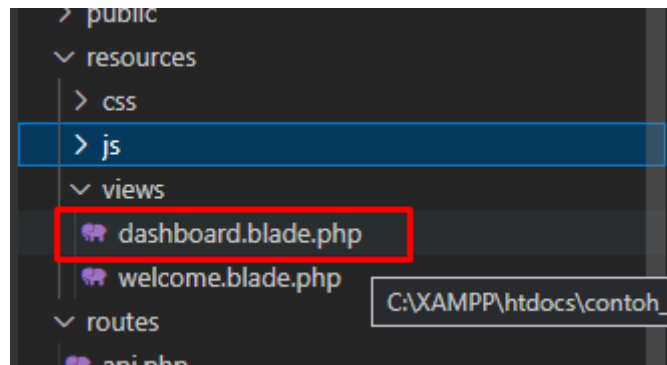


Gambar 13 Drop ke folder **public**



Gambar 14 Folder **plugins** berhasil disalin

Kita lanjutkan membuat sebuah *view* yang akan kita gunakan untuk menampilkan *navbar*. Silakan **buat file baru** dengan nama **dashboard.blade.php** di dalam folder **resources/views**.



Untuk membuat tampilan *navbar* yang akan dipakai berulang. **Masukkan kode** berikut ini di dalam **dashboard.blade.php** :

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <title>Amsang Tech | UGD_X_YYYY</title>
    <!-- Google Font: Source Sans Pro -->
    <link rel="stylesheet"
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,400i,
700&display=fallback">
    <!-- Font Awesome Icons -->
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.2.0/css/all.min.css">
    <!-- Theme style -->
    <link rel="stylesheet" href="{{ asset('css/adminlte.min.css') }}">
  </head>
  <body class="hold-transition sidebar-mini">
    <div class="wrapper">
      <!-- Navbar -->
      <nav class="main-header navbar navbar-expand navbar-white
navbar-light">
        <!-- Left navbar links -->
        <ul class="navbar-nav">
          <li class="nav-item">
            <a class="nav-link" data-widget="pushmenu" href="#"
role="button">
              <i class="fas fa-bars"></i>
            </a>
```

```

        </li>
    </ul>
    <!-- Right navbar links -->
    <ul class="navbar-nav ml-auto">
        <!-- Navbar Search -->
        <li class="nav-item">
            <a class="nav-link" data-widget="navbar-search"
href="#" role="button">
                <i class="fas fa-search"></i>
            </a>
            <div class="navbar-search-block">
                <form class="form-inline">
                    <div class="input-group input-group-sm">
                        <input class="form-control form-
control-navbar" type="search" placeholder="Search" aria-label="Search">
                        <div class="input-group-append">
                            <button class="btn btn-navbar"
type="submit">
                                <i class="fas fa-search"></i>
                            </button>
                            <button class="btn btn-navbar"
type="button" data-widget="navbar-search">
                                <i class="fas fa-times"></i>
                            </button>
                        </div>
                    </div>
                </form>
            </div>
        </li>
        <li class="nav-item">
            <a class="nav-link" data-widget="fullscreen"
href="#" role="button">
                <i class="fas fa-expand-arrows-alt"></i>
            </a>
        </li>
    </ul>
</nav>
<!-- /.navbar -->
<!-- Main Sidebar Container -->
<aside class="main-sidebar sidebar-dark-primary elevation-4">
    <!-- Brand Logo -->
    <a href="#" class="brand-link">
        
        <span class="brand-text font-weight-light">Amsang
Tech</span>
    </a>

```

```

        <!-- Sidebar -->
        <div class="sidebar">
            <!-- Sidebar user panel (optional) -->
            <div class="user-panel mt-3 pb-3 mb-3 d-flex">
                <div class="image">
                    
                </div>
                <div class="info">
                    <a href="#" class="d-block">Nama Praktikan</a>
                </div>
            </div>
            <!-- SidebarSearch Form -->
            <div class="form-inline">
                <div class="input-group" data-widget="sidebar-
search">
                    <input class="form-control form-control-
sidebar" type="search" placeholder="Search" aria-label="Search">
                    <div class="input-group-append">
                        <button class="btn btn-sidebar">
                            <i class="fas fa-search fa-fw"></i>
                        </button>
                    </div>
                </div>
            </div>
            <!-- Sidebar Menu -->
            <nav class="mt-2">
                <ul class="nav nav-pills nav-sidebar flex-column"
data-widget="treeview" role="menu" data-accordion="false">
                    <li class="nav-item">
                        <a href="{{ url('departemen') }}"
class="nav-link">
                            <i class="nav-icon far fa-circle"></i>
                            <p>Departemen</p>
                        </a>
                    </li>
                </ul>
            </nav>
            <!-- /.sidebar-menu -->
        </div>
    <!-- /.sidebar -->
</aside>
<!-- Content Wrapper. Contains page content -->
<div class="content-wrapper">
    @yield('content')
</div>
<!-- /.content-wrapper -->
<!-- Main Footer -->

```

```

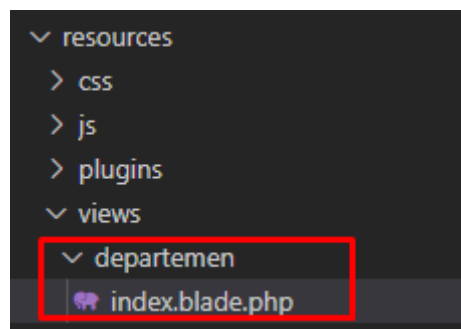
        <footer class="main-footer">
            <!-- To the right -->
            <div class="float-right d-none d-sm-inline"> NPM_PRAKTIKAN
</div>

            <!-- Default to the left -->
            <strong>Copyright &copy; {{ date('Y') }} <a
href="#">AdminLTE.io</a>. </strong> All rights reserved.
        </footer>
    </div>
    <!-- ./wrapper -->
    <!-- REQUIRED SCRIPTS -->
    <!-- jQuery -->
    <script src="{{ asset('plugins/jquery/jquery.min.js') }}"></script>
    <!-- Bootstrap 4 -->
    <script src="{{ asset('plugins/js/bootstrap.bundle.min.js')
}}"></script>
    <!-- AdminLTE App -->
    <script src="{{ asset('js/adminlte.min.js') }}"></script>
</body>
</html>

```

Ganti **UGD_X_YYY** dengan kode Kelas dan NPM kalian

Kita lanjutkan membuat sebuah *view* yang akan kita gunakan untuk menampilkan data. Silakan **buat folder departemen** di dalam folder **resources/views**, lalu **buat file baru** dengan nama **index.blade.php**.



Untuk membuat tampilan halaman departemen yang akan dipakai untuk menampilkan data. **Masukkan kode** berikut ini di dalam **departemen/index.blade.php** :

```

@extends('dashboard')

@section('content')
    <div class="content-header">
        <div class="container-fluid">
            <div class="row mb-2">

```

```

        <div class="col-sm-6">
            <h1 class="m-0">Departemen</h1>
        </div>
        <!-- /.col -->
        <div class="col-sm-6">
            <ol class="breadcrumb float-sm-right">
                <li class="breadcrumb-item">
                    <a href="{ { url('departemen') }}">Departemen</a>
                </li>
                <li class="breadcrumb-item active">Index</li>
            </ol>
        </div>
        <!-- /.col -->
    </div>
    <!-- /.row -->
</div>
<!-- /.container-fluid -->
</div>
<!-- /.content-header -->
<!-- Main content -->
<div class="content">
    <div class="container-fluid">
        <div class="row">
            <div class="col-12">
                <div class="card">
                    <div class="card-body">
                        <div class="table-responsive p-0">
                            <table class="table table-hover text-
nowrap">
                                <thead>
                                    <tr>
                                        <th class="text-center">Nama
Departemen</th>
                                        <th class="text-center">Nama
Manger</th>
                                        <th class="text-center">Jumlah
Pegawai</th>
                                    </tr>
                                </thead>
                                <tbody>
                                    @forelse ($departemen as $item)
                                        <tr>
                                            <td class="text-center">{{
$item->nama_departemen }}</td>
                                            <td class="text-center">{{
$item->nama_manager }}</td>
                                            <td class="text-center">{{
$item->jumlah_pegawai }}</td>

```

```

        </tr>
        @empty
        <div class="alert alert-danger">
            Data Departemen belum tersedia
        </div>
        @endforelse
    </tbody>
</table>
</div>
</div>
<!-- /.card-body -->
</div>
<!-- /.card -->
</div>
<!-- /.col-md-6 -->
</div>
<!-- /.row -->
</div>
<!-- /.container-fluid -->
</div>
@endsection

```

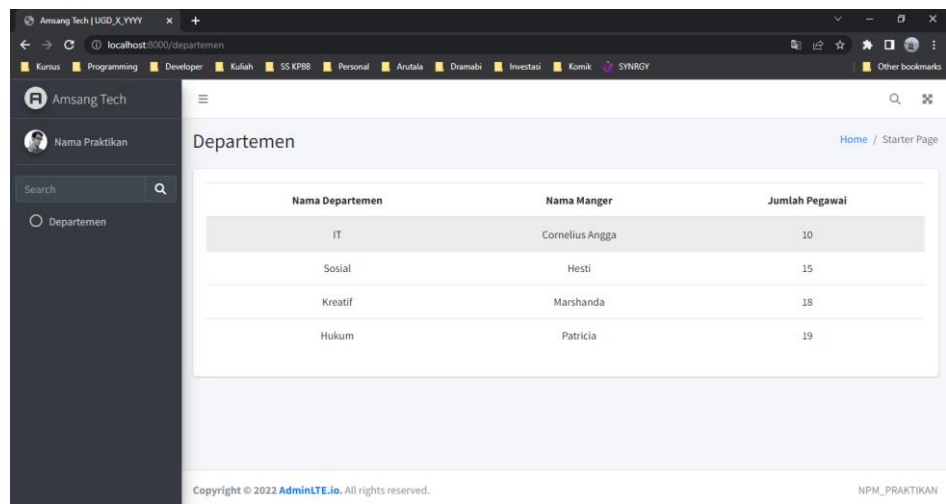
Dari penambahan kode di atas, kita melakukan perulangan untuk menampilkan data yang dikirimkan melalui *controller*. Dan untuk perulangan, kita menggunakan direktif **@forelse**, di mana jika data ada maka akan dilakukan *looping* dan jika data tidak tersedia, maka akan menampilkan pesan/*message*.

10. Uji Coba Menampilkan Data

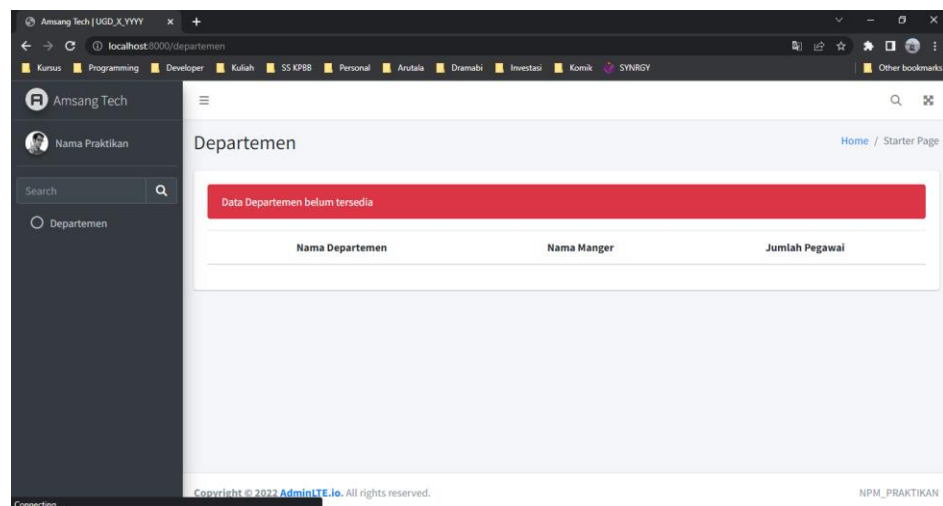
Untuk menguji apakah web bekerja dengan baik, sekarang jalankan perintah berikut ini:

```
php artisan serve
```

Kita bisa mencoba membuka website di dalam URL <http://localhost:8000/> dan jika berhasil maka akan menampilkan halaman seperti berikut ini:



Gambar 15 Tampilan apabila data departemen tersedia



Gambar 16 Tampilan apabila data tidak tersedia