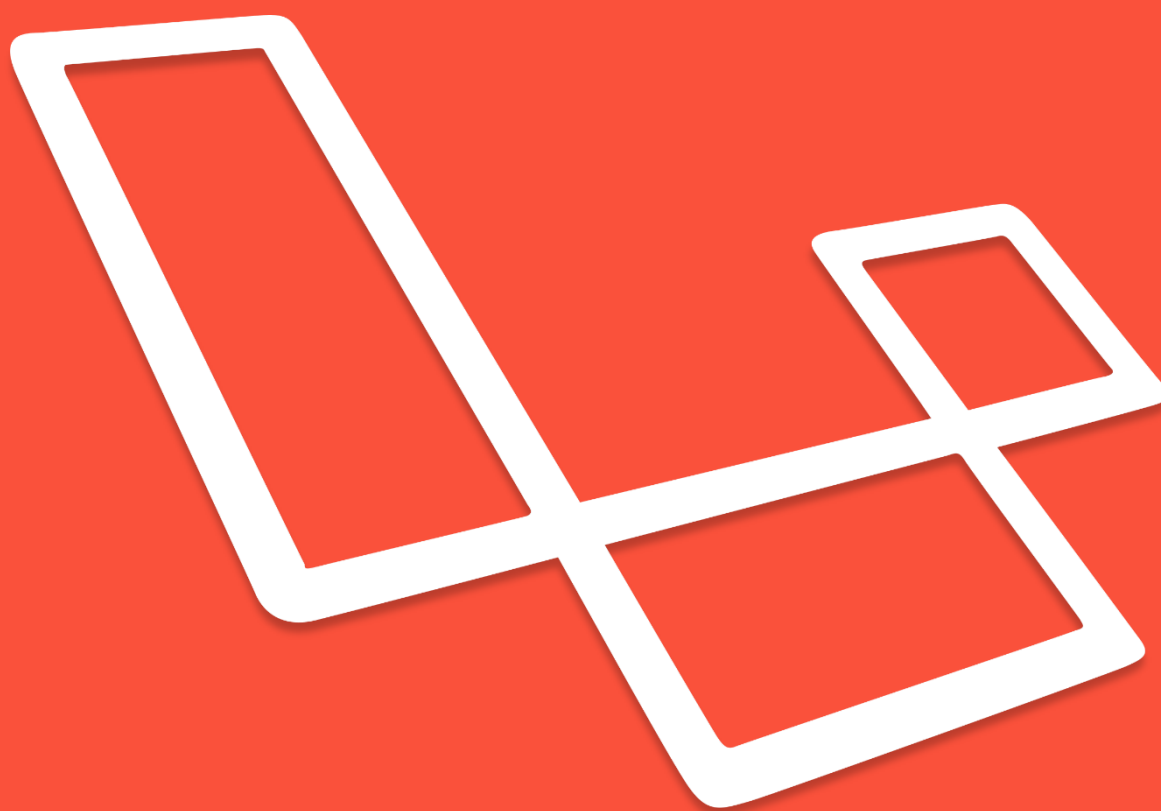


Dieka Prastya

# Beginning Laravel



Memulai Awal Perjalanan  
Baru Menggunakan Framework  
**Laravel 5.6**

## Daftar Isi

Beginning Laravel .....	4
Pembuatan Aplikasi Sederhana .....	6
1. Merancang Design Aplikasi .....	6
2. Instalasi Kebutuhan Framework Laravel.....	9
• Kebutuhan Sistem & Pengembangan.....	9
• Kebutuhan Aplikasi.....	9
3. Membuat Project Baru dengan Framework Laravel.....	9
Memastikan Composer Terinstall dengan baik .....	10
Menginstall Package Laravel Installer di Composer .....	10
Membuat Project Laravel Baru dengan Package Laravel Installer .....	11
Menjalankan & Memastikan Project Laravel Dapat Dijalankan .....	12
4. Membuat Halaman Baru di Framework Laravel.....	13
Membuat Route GET.....	14
Membuat Controller dan Fungsinya .....	16
Mengarahkan Fungsi Controller kepada View .....	17
Menyiapkan Assets (AdminLTE) untuk Membuat View .....	17
Membuat Master / Layout View .....	18
Membuat View Bagian-Bagian Layout.....	21
Membuat View Tiap Halaman.....	24
Memastikan Setiap Halaman telah Berhasil Diakses .....	29
5. Membuat Database dengan Laravel Database Migration .....	30
Membuat Schema Migration .....	31
Menyiapkan Database untuk di Migrasikan .....	33
Mengatur Koneksi Database dari Aplikasi .....	33
Memulai Proses Migration.....	33

6. Menyelesaikan Halaman Mahasiswa Tambah (Create / Insert).....	34
Membuat Fungsi Controller untuk Menyimpan ke Database .....	35
7. Menyelesaikan Fungsi Pengambilan Data Halaman Daftar Mahasiswa .....	37
Memproses View Jurusan & Jenis Kelamin.....	39
8. Menyelesaikan Fungsi Penghapusan Data pada Halaman Daftar Mahasiswa .....	40
Membuat Fungsi Controller untuk Menghapus Mahasiswa .....	41
9. Menyelesaikan Fungsi Halaman Perubahan Data Mahasiswa .....	42
Menyelesaikan Fungsi Update pada Halaman Perubahan Data Membuat Route POST Update.....	45
Membuat Fungsi Controller untuk Update ke Database .....	45
Penutup .....	47



## Beginning Laravel

Laravel adalah framework yang paling banyak digunakan oleh PHP Developer di dunia saat ini (per tanggal 29/06/2018), terlihat dari bagaimana tren keyword pencarian di Google yang jika dibandingkan dengan framework-framework PHP lain semacam CodeIgniter, Yii, Symphony, CakePHP, dll nya, Laravel masihlah menempati posisi puncak pencarian, hingga banyaknya perusahaan-perusahaan yang merekrut Laravel Developer, selain itu terlihat juga dari bagaimana komunitas nya yang semakin besar dan aktif. Di Indonesia sendiri framework Laravel terus meningkat perkembangannya, setelah sebelumnya framework CodeIgniter yang mendominasi perkembangan pengguna bagi PHP Developer di Indonesia.

Popularitas Laravel ini tidak terlepas dari selaku pihak developer framework Laravel nya sendiri yang terus memberikan banyak kemudahan bagi penggunanya, selain itu ekosistem nya yang terus berkembang mengikuti perkembangan teknologi, serta dari penggunanya yang senantiasa aktif memberikan dukungan berupa referensi belajar hingga berbagai package library pelengkap nya, semua itu membuat Laravel sangatlah luas & indah untuk dipelajari dimasa sekarang ini.

Oleh karena itu penulis tertarik untuk membuat modul belajar framework Laravel bagi siapapun yang ingin memulai untuk mempelajarinya, dan merasakan berbagai kemudahan dari framework Laravel, tidak terkhusus bagi PHP Developer yang masih menggunakan native PHP tetapi juga bagi PHP Developer yang sudah terbiasa menggunakan framework lain dan ingin beralih ke framework Laravel.

Akhir kata, harapan penulis semoga dengan hadirnya modul “Beginning Laravel” ini dapat menjadi manfaat bagi Anda yang ingin sedikit mengenal hingga mulai mempelajari & memahami tentang framework Laravel.

## Kenapa Framework?

Seringkali bagi developer PHP Native yang belum pernah menggunakan framework bertanya-tanya, kenapa kita harus beralih menggunakan framework? Saya tetap bisa membuat website dengan PHP tanpa harus menggunakan framework, kenapa juga

sekarang saya harus berlelah-lelah untuk belajar lagi? Apa keharusannya saya beralih dari PHP Native ke PHP Framework?

Berikut beberapa manfaat yang bisa Anda dapatkan jika mengembangkan sebuah aplikasi berbasis web menggunakan framework:

1. **Rapid Development**, proses pembuatan aplikasi jauh lebih cepat dibandingkan pengembangan tanpa menggunakan framework, hal ini disebabkan karena banyaknya dukungan *library* yang disediakan oleh framework tersebut yang dimana hal itu sangat mempermudah developer untuk mengembangkan aplikasinya, misalnya saja seorang developer yang ingin membuat sebuah form isian yang memiliki validasi, maka ia akan sangat dipermudah dengan *library* yang bernama Form Validation yang dengan itu membuat kita dapat memvalidasi isian hanya dengan beberapa baris kode, atau jika kita ingin membuat sebuah fitur *Authentication*, maka kita juga hanya perlu menambahkan *library Authentication* pada proyek aplikasi kita, maka dengan hanya konfigurasi beberapa hal pada framework maka fitur *Authentication* pun dapat kita gunakan, dan banyak lagi *library* yang dapat kita tambahkan untuk menunjang kecepatan dalam pengembangan aplikasi. Hal ini pun menarik minat penggiat bisnis *web development*, yang dimana banyak perusahaan yang cenderung lebih mencari developer yang paham framework dibandingkan dengan yang native dikarenakan cepatnya proses develop yang berefek pada semakin rendahnya biaya pembuatan aplikasi.
2. **More Secure**, penggunaan framework membuat aplikasi kita jauh lebih aman dibandingkan dengan Native, alasan ini disebabkan karena pada Framework Open Source terdapat update berkala pada vendor pembuat Framework, Contohnya Laravel selalu melakukan update version, update tersebut tidak hanya karena adanya penambahan fitur, tetapi juga perbaikan disebabkan security issues, hal ini menyebabkan keamanan aplikasi web yang menggunakan framework terus berkembang, selain itu framework pun senantiasa menyediakan library-library yang menunjang untuk meningkatkan keamanan.
3. **Easier Maintenance**, framework yang ketika pembangunannya wajib mengikuti struktur pengembangan dari pemilik framework menyebabkan framework jauh lebih mudah di maintenance, contohnya antara satu developer Laravel dengan

developer Laravel yang lain dapat lebih mudah untuk saling mengerjakan satu project yang sama menggunakan framework yang sama, hal itu memudahkan juga ketika suatu saat nanti anda akan memaintain, selain itu adapula fitur-fitur pada framework seperti unit testing untuk melakukan uji test pada fungsional aplikasi dengan lebih mudah.

4. **Stronger Teamwork**, dengan design pattern yang sudah terstruktur dengan baik, banyaknya dukungan library, dan dokumentasi library yang lengkap membuat seorang developer bisa fokus pada pengembangan aplikasinya tanpa sibuk pada pembuatan shared function yang sudah ada pada framework, hal ini akan mengurangi ketergantungan berkaitan shared function antara satu developer dengan developer yang lain, dan tentu saja itu akan lebih memperkuat team pada kerja utamanya.

## Pembuatan Aplikasi Sederhana

Agar lebih dalam memahaminya lagi, mari kita langsung mulai dengan langkah langkah membuat aplikasi yang akan mewakili sebagian besar kebutuhan dasar dalam membuat aplikasi menggunakan framework Laravel.

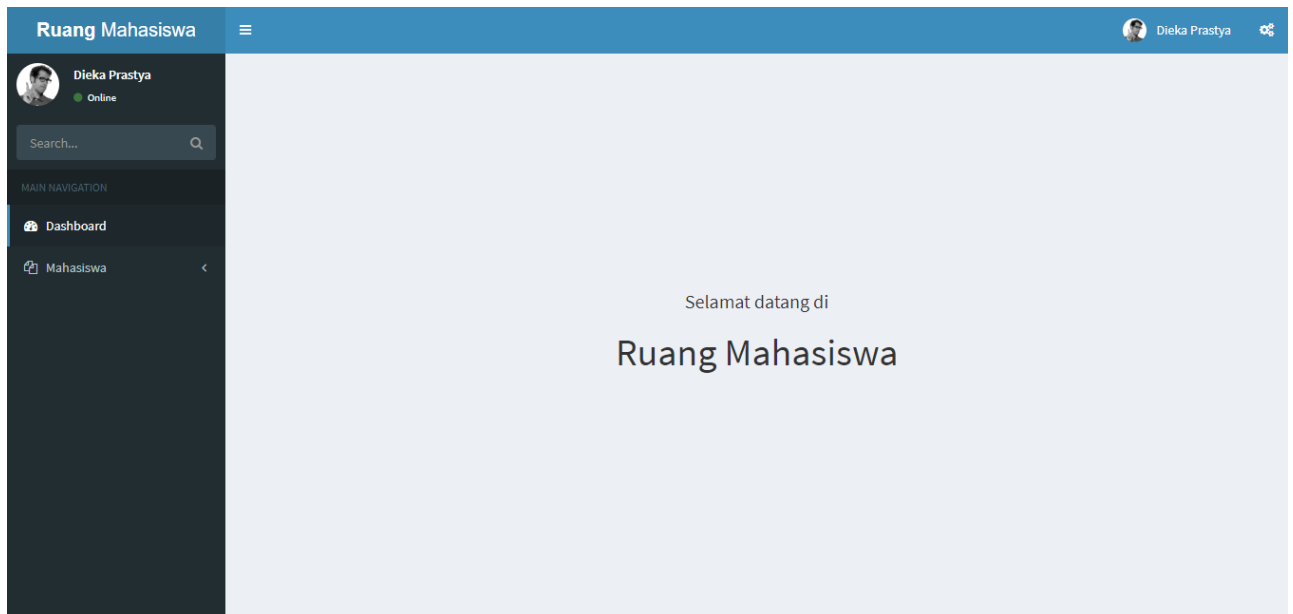
### 1. Merancang Design Aplikasi

Sebelum masuk ke bagian teknis Laravel, mari kita dahulukan dengan merancang aplikasi yang akan kita buat supaya proses pembuatannya nanti akan jauh lebih mudah & terstruktur dengan baik. Proses perancangan aplikasi biasanya dimulai dari Analisa, perancangan Database, hingga terakhir perancangan UI/UX, namun kali ini akan kita singkat dengan dimulai dari perancangan designnya UI aplikasi nya agar lebih mudah untuk dipelajari.

Langkah pertama buatlah mockup atau design dari setiap halaman web atau aplikasi yang ingin kita buat beserta dengan Path URL dari setiap halamannya, berikut contoh design UI rancangan web aplikasi yang ingin kita buat:

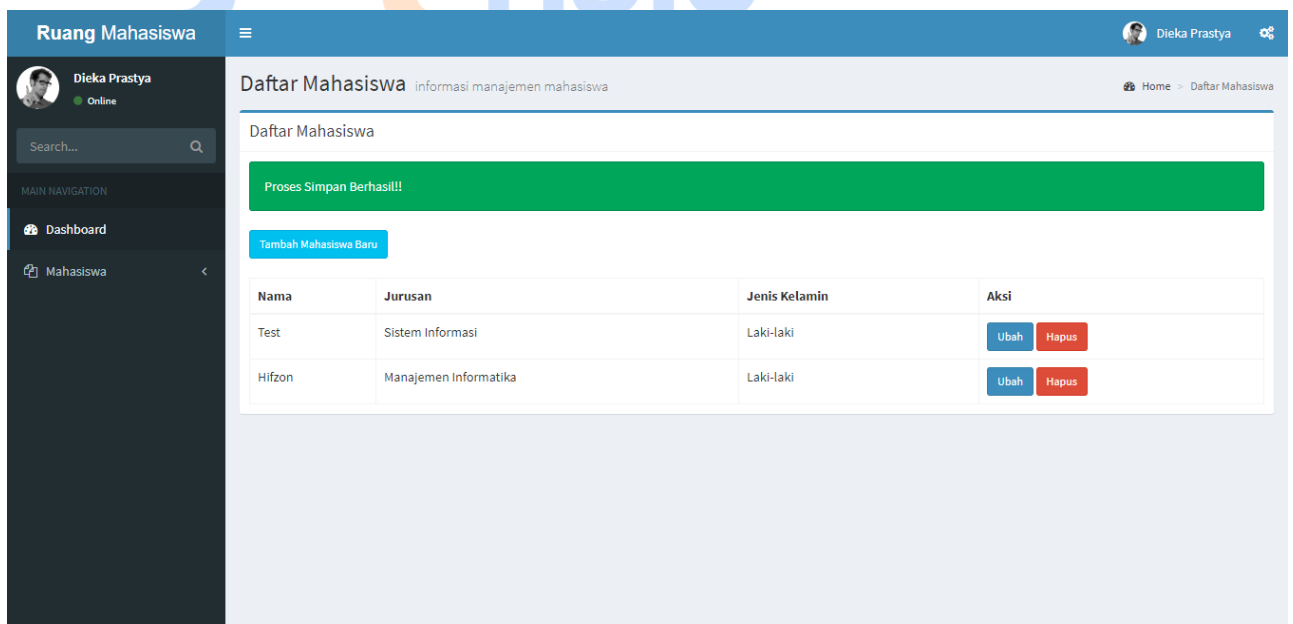
a. Halaman Home

Path URL: <http://contoh.com/>



b. Halaman Daftar Mahasiswa

Path URL: <http://contoh.com/mahasiswa>



c. Halaman Tambah Data Mahasiswa

Path URL: <http://contoh.com/mahasiswa/tambah>

The screenshot shows a web application interface for adding a new student. The header bar is blue with the text 'Ruang Mahasiswa' on the left and a user profile 'Dieka Prasty' on the right. A dark sidebar on the left contains a search bar and navigation links for 'Dashboard' and 'Mahasiswa'. The main content area is titled 'Tambah Mahasiswa' and contains a form with the following fields: 'Nama' (text input with placeholder 'Masukkan nama anda..'), 'Jurusan' (dropdown menu with 'Sistem Informasi' selected), and 'Jenis Kelamin' (radio buttons for 'Laki-laki' and 'Perempuan', with 'Laki-laki' selected). A blue 'Daftarkan' button is at the bottom of the form.

d. Halaman Ubah Data Mahasiswa

Path URL: <http://contoh.com/mahasiswa/ubah>

The screenshot shows the 'Ubah Mahasiswa' (Edit Student) page. The interface is identical to the previous one, but the form fields are pre-filled: 'Nama' contains 'Test', 'Jurusan' is 'Sistem Informasi', and 'Jenis Kelamin' has 'Laki-laki' selected. The blue button at the bottom is labeled 'Simpan' (Save).



## 2. Instalasi Kebutuhan Framework Laravel

Ada beberapa prasayarat untuk dapat memulai project framework Laravel yang ingin kita buat, berikut prasayarat yang dibutuhkan:

### Kebutuhan Sistem & Pengembangan

1. XAMPP >= 7.1.3 (<https://www.apachefriends.org/>)
  - a. PHP >= 7.1.3
  - b. Apache (Web Server)
  - c. MySQL (Database)
2. Composer >= 1.6.5 (<https://getcomposer.org/>)
3. IDE (Visual Studio Code / Sublime Text / Atom Text Editor)
4. Browser (Chrome / Mozilla Firefox)

### Kebutuhan Aplikasi

1. JQuery v3.3.1 (<https://jquery.com/>)
2. Bootstrap v4.1.1 (<https://getbootstrap.com/>)
3. AdminLTE (Admin Template)

Lakukan instalasi pada tools-tools diatas, dianjurkan untuk melakukan Instalasi XAMPP terlebih dahulu sebelum menginstall Composer, dikarenakan Composer akan membutuhkan PHP untuk konfigurasinya diawal ketika melakukan instalasi Composer pertama kali.

## 3. Membuat Project Baru dengan Framework Laravel

Terdapat berbagai macam cara untuk membuat project baru dengan Framework Laravel, yaitu:

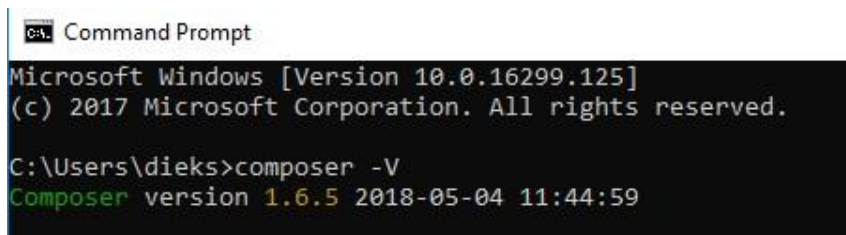
- a. Menginstall paket Composer (<https://packagist.org/packages/laravel/laravel>)
- b. Melakukan clone Git (<https://github.com/laravel/laravel>)

Namun kali ini kita akan memilih salah satu cara saja, yaitu menggunakan cara yang pertama (menginstall paket Composer).

**Composer** adalah alat manajemen dependency pada PHP seperti npm (Node.js) dan Bundler (Ruby). Composer memungkinkan untuk membuat library pada project dan dengan composer kita dapat menginstall atau mengupdate paket library project secara otomatis tanpa harus menginstall secara manual.

### Memastikan Composer Terinstall dengan baik

1. Jalankan Command Prompt terlebih dahulu pada Windows anda.
2. Lalu ketik & jalankan perintah **composer -V** yang dimana perintah tersebut digunakan untuk melihat versi dari Composer.



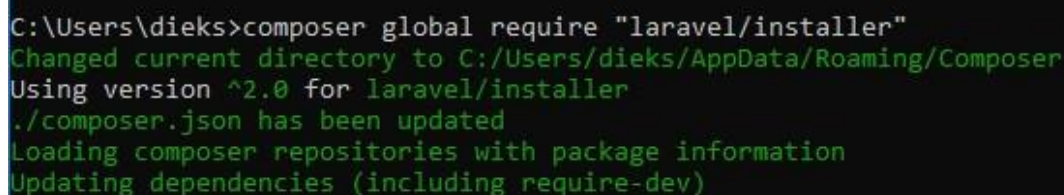
```
Command Prompt
Microsoft Windows [Version 10.0.16299.125]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\dieks>composer -V
Composer version 1.6.5 2018-05-04 11:44:59
```

3. Jika muncul versi dari Composer, maka artinya Composer anda telah mampu berjalan dan terinstall dengan baik.

### Menginstall Package Laravel Installer di Composer

1. Jalankan perintah **composer global require "laravel/installer"** pada Command Prompt anda.



```
C:\Users\dieks>composer global require "laravel/installer"
Changed current directory to C:/Users/dieks/AppData/Roaming/Composer
Using version ^2.0 for laravel/installer
./composer.json has been updated
Loading composer repositories with package information
Updating dependencies (including require-dev)
```

2. Lalu setelah itu pastikan instalasi package Laravel Installer telah berhasil terinstall dengan baik dengan menjalankan perintah **laravel -V**.

```
C:\Users\dieks>laravel -V
Laravel Installer 2.0.1
```

“**Package Laravel Installer** adalah sebuah paket composer dari Laravel untuk kita membuat project Laravel baru tanpa perlu kita download manual dan mengkonfigurasi secara manual, dengan hanya dengan menjalankan perintah pada Package Laravel Installer, maka project Laravel baru bisa langsung terbuat, terdownload, dan terkonfigurasi dengan baik. “

### Membuat Project Laravel Baru dengan Package Laravel Installer

1. Proses pembuatan project akan membuat folder/directory baru di drive kita, maka pastikan diarahkan terlebih dahulu letak directory di Command Prompt ke directory project yang kita inginkan.
2. Jika sudah sesuai maka jalankan perintah **laravel new <nama\_directory>** untuk memulai perintah pembuatan project Laravel baru.

```
C:\Users\dieks\Projects>laravel new laravel_batch
Crafting application...
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
Package operations: 70 installs, 0 updates, 0 removals
- Installing doctrine/inflector (v1.3.0): Loading from cache
- Installing doctrine/lexer (v1.0.1): Loading from cache
- Installing dragonmantank/cron-expression (v2.2.0): Loading from cache
- Installing erusev/parsedown (1.7.1): Loading from cache
- Installing vlucas/phpdotenv (v2.5.0): Downloading (100%)
- Installing symfony/css-selector (v4.1.1): Loading from cache
- Installing tijsverkoyen/css-to-inline-styles (2.2.1): Loading from cache
- Installing symfony/polyfill-php72 (v1.8.0): Loading from cache
- Installing symfony/polyfill-mbstring (v1.8.0): Loading from cache
```

3. Proses instalasi membutuhkan koneksi internet, karena secara otomatis composer akan mendownload semua package yang dibutuhkan oleh project Laravel, dan akan mengekstraknya pada directory project layaknya sebuah template. Jika telah selesai membuat project maka akan tampil seperti dibawah ini dan akan terbentuk folder baru di lokasi yang telah dibuat.

```
sebastian/global-state suggests installing ext-uopz (*)
phpunit/php-code-coverage suggests installing ext-xdebug (^2.6.0)
phpunit/phpunit suggests installing ext-soap (*)
phpunit/phpunit suggests installing ext-xdebug (*)
phpunit/phpunit suggests installing phpunit/php-invoker (^2.0)
Generating optimized autoload files
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
> @php artisan key:generate
Application key [base64:2H4vQjnkBnA2+By3i0BIJ7RCx+Oc5n4KI03zusTfUTQ=] set successfully.
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover
Discovered Package: fideloper/proxy
Discovered Package: laravel/tinker
Discovered Package: nunomaduro/collision
Package manifest generated successfully.
Application ready! Build something amazing.
```

4. Untuk mendapatkan keterangan yang lebih lengkap & detail bisa didapatkan di dokumentasi bagian instalasi milik Laravel (<https://laravel.com/docs/5.6/installation>).

### Menjalankan & Memastikan Project Laravel Dapat Dijalankan

1. Masuk ke folder project dengan perintah **cd <nama\_directory\_project>**.
2. Lalu jalankan perintah **php artisan serve** yang berfungsi untuk meminta php untuk menjalankan fitur **artisan server**, yang dimana fitur tersebut digunakan untuk memulai web server php menjalankan/menghost project Laravel yang telah kita buat di PC yang kita jalankan.

```
C:\Users\dieks\Projects\laravel_batch>php artisan serve  
Laravel development server started: <http://127.0.0.1:8000>
```

**Artisan** adalah sebuah fitur CLI yang dihadirkan oleh Laravel sebagai shortcut programmer untuk membantu memudahkan developer ketika membangun aplikasi. Contohnya untuk membuat Controller, Model, Migration, menjalankan aplikasi, dan berbagai macam kemudahan lainnya, yang dimana semua perintah tersebut dapat dilakukan hanya dengan satu perintah menggunakan CLI.

3. Secara default project akan dihost pada alamat <http://127.0.0.1:8000>, maka bukalah alamat tersebut untuk memastikan proses install & konfigurasi awal berjalan dengan sempurna. Jika berhasil maka akan muncul halaman pertama bawaan dari template project Laravel seperti dibawah ini.



Laravel

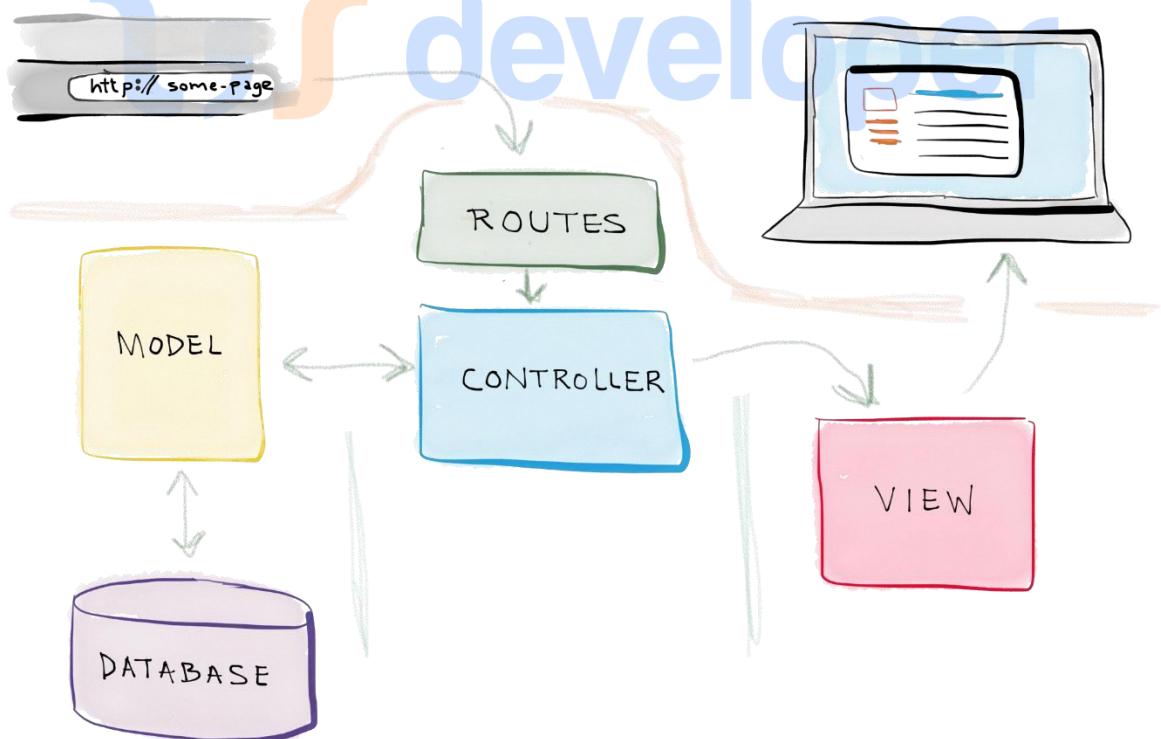
DOCUMENTATION LARACASTS NEWS FORGE GITHUB

#### 4. Membuat Halaman Baru di Framework Laravel

Tidak seperti PHP Native yang hanya dengan membuat file native PHP untuk membuat sebuah halaman website, Laravel menerapkan konsep MVC pada frameworknya, sehingga untuk membuat sebuah halaman sangatlah wajib bagi kita menggunakan kaidah tersebut (MVC) di framework Laravel ini.

**Model-View-Controller** atau **MVC** sendiri adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (Model) dari tampilan (View) dan cara bagaimana memprosesnya (Controller). Dalam implementasinya kebanyakan framework dalam aplikasi website adalah berbasis arsitektur MVC.

Simplenya untuk membuat aplikasi yang menggunakan kaidah MVC, kita diharuskan untuk membuat antara tampilan (View), hubungan data (Model), dan logika proses (Controller) terpisah satu dengan yang lainnya, yang dimana teknisnya View dipakai untuk meletakkan baris kode design frontend (HTML, CSS, JS), Controller dipakai untuk melakukan logika pemrosesan (PHP), dan Model untuk menghasilkan data & strukturnya yang kelak dikirimkan ke Controller, kemudian diteruskan ke View.



Diatas adalah gambaran sederhana bagaimana kaidah MVC di framework Laravel bekerja, sebagaimana User melakukan http request ke website, lalu http request tersebut

dikenali oleh Route, kemudian Route mengarahkannya kepada Controller untuk dilakukan pemrosesan, pada pemrosesan di Controller melibatkan Model untuk koneksi ke Database jika dibutuhkan, dan setelah hasil pemrosesan Controller selesai maka Controller akan menentukan View (Tampilan) yang akan dikembalikan kepada User yang merequest.

Langkah pertama untuk membuat halaman berkonsep MVC di framework Laravel untuk aplikasi yang ingin kita buat adalah sebagai berikut:

1. Buatlah sebuah Route (Pengenal Rute HTTP Request)
2. Buatlah sebuah Controller (Pemroses/Pengontrol)
3. Buatlah sebuah View (Tampilan HTML)

### **Membuat Route GET**

Pada tahap rancangan aplikasi sebelumnya terdapat 4 url aktif yang dapat diakses oleh user dengan menggunakan web browsernya, yaitu halaman home, daftar mahasiswa, penambahan mahasiswa, dan terakhir perubahan mahasiswa, oleh karena itu maka disimpulkan kita akan membuat 4 route berbentuk GET yang dapat diakses langsung oleh user dengan web browsernya, dan penulis pun berencana membuat 4 route tersebut mengarah kepada hanya 2 controller dengan 4 fungsi yang berbeda, yang dimana nama controller diklasifikasikan berdasarkan nama entitynya (Home & Mahasiswa), dan nama fungsinya disesuaikan dengan fungsi dari halamannya (index, tambah, ubah), maka pendefinisianannya yaitu:

URL: <http://domain.com>

- Nama Route: home
- Nama Controller: HomeController
- Nama Fungsi: getIndex()

URL: <http://domain.com/mahasiswa>

- Nama Route: mahasiswa
- Nama Controller: MahasiswaController
- Nama Fungsi: getIndex()

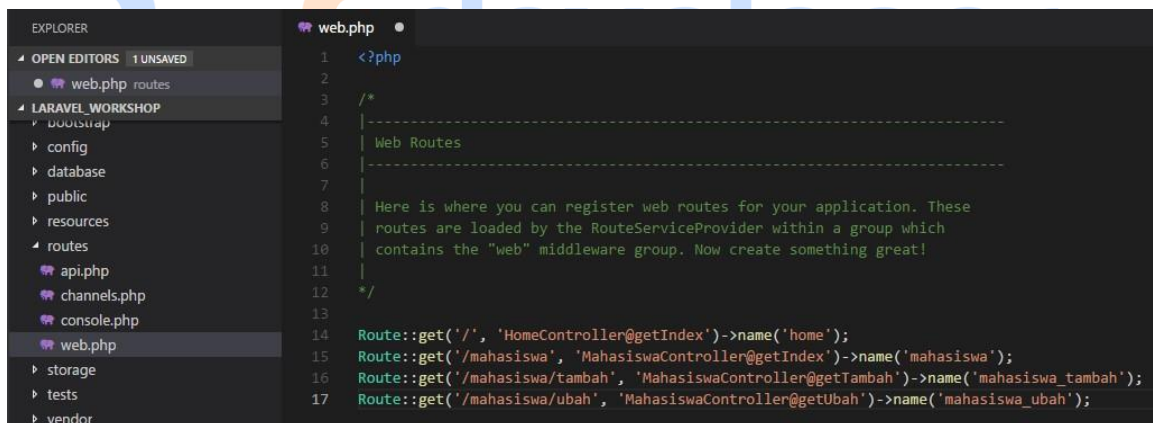
URL: <http://domain.com/mahasiswa/tambah>

- Nama Route: tambah
- Nama Controller: MahasiswaController
- Nama Fungsi: getTambah()

URL: <http://domain.com/mahasiswa/ubah>

- Nama Route: ubah
- Nama Controller: MahasiswaController
- Nama Fungsi: getUbah()

1. Bukalah file untuk pengaturan Route yang terdapat pada “<project\_folder>/routes/web.php”, dan hapuslah fungsi Route bawaan dari Laravel terlebih dahulu.
2. Tambahkan baris kode baru yang berfungsi untuk mendefinisikan Pengenal HTTP Request atau Route baru untuk halaman-halaman yang ingin kita buat.



```
1 <?php
2
3 /*
4 |-----
5 | Web Routes
6 |-----
7 |
8 | Here is where you can register web routes for your application. These
9 | routes are loaded by the RouteServiceProvider within a group which
10 | contains the "web" middleware group. Now create something great!
11 |
12 */
13
14 Route::get('/', 'HomeController@getIndex')->name('home');
15 Route::get('/mahasiswa', 'MahasiswaController@getIndex')->name('mahasiswa');
16 Route::get('/mahasiswa/tambah', 'MahasiswaController@getTambah')->name('mahasiswa_tambah');
17 Route::get('/mahasiswa/ubah', 'MahasiswaController@getUbah')->name('mahasiswa_ubah');
```

Fungsi **Route::get(<url>, <fungsi\_controller>)->name(<route\_name>)** adalah fungsi untuk mendefinisikan routing http request GET sebagai pengenalan, dan mengarahkannya ke Fungsi Controller yang kita inginkan, Contohnya sebuah http request GET dengan url <http://domain.com/mahasiswa> ingin diarahkan ke Controller dengan nama HomeController dengan fungsi getIndex(), maka didefinisikan dengan baris kode:

**Route::get('/mahasiswa', 'MahasiswaController@getIndex')->name('mahasiswa');**

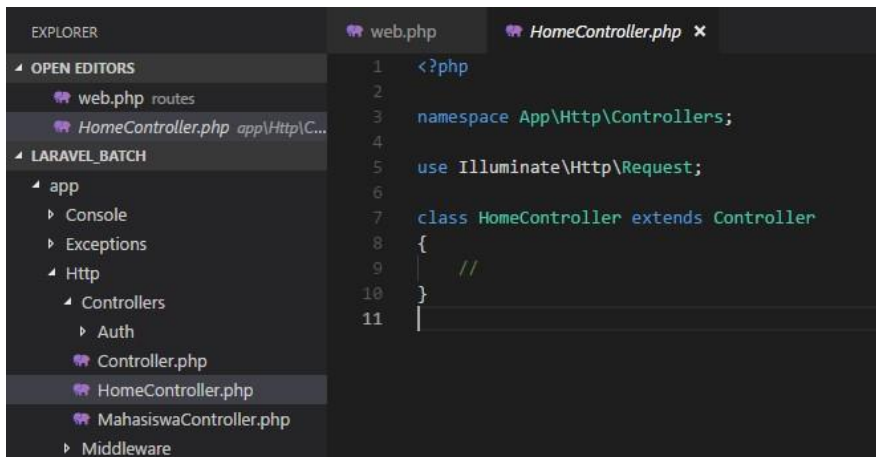


## Membuat Controller dan Fungsinya

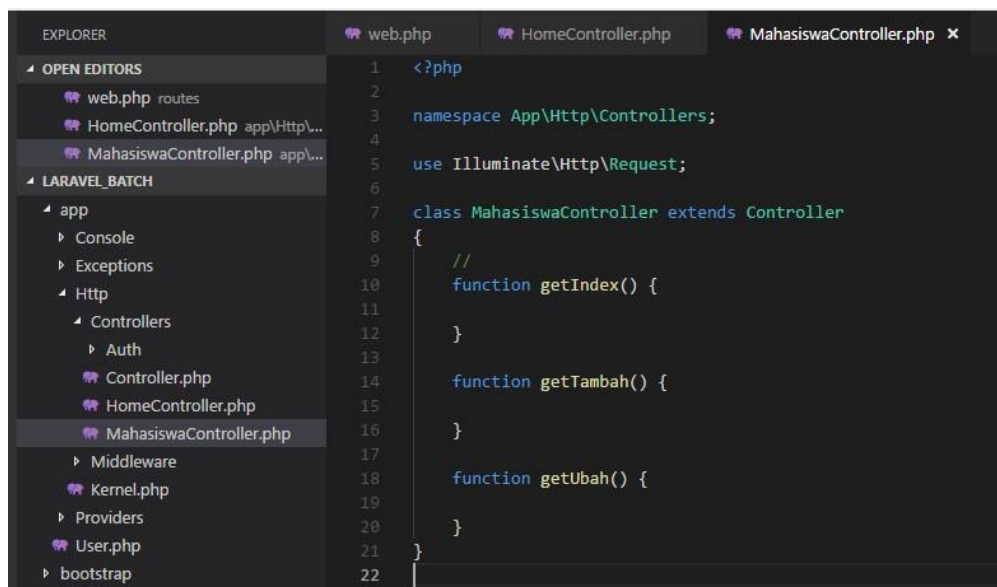
1. Buka Command Prompt, kemudian masuk ke directory project.
2. Jalankan perintah **php artisan make:controller <nama\_controller>** yang berfungsi untuk membuat controller secara otomatis. Controller akan tergenerate otomatis pada directory <project>/app/Http/Controllers/<nama\_controller>.php

```
C:\Users\dieks\Projects\laravel_batch>php artisan make:controller HomeController  
Controller created successfully.
```

3. Buatlah 2 controller, yaitu HomeController, dan MahasiswaController.



4. Buatlah 4 fungsi sesuai dengan rancangan route yang telah dibuat, yaitu pada HomeController terdapat fungsi getIndex(), dan pada MahasiswaController terdapat fungsi getIndex(), getTambah(), dan getUbah().





## Mengarahkan Fungsi Controller kepada View

Sebelumnya kita mendefinisikan 4 fungsi baru pada setiap controller, namun semua fungsi pada controller tersebut belumlah diberitahu tampilan apa yang akan diberikan kepada user yang telah mengakses route hingga diarahkan ke fungsi tersebut, maka dari itu kita juga diharuskan mengarahkan fungsi-fungsi tersebut agar memberikan / mengembalikan kembali sebuah tampilan (View) yang kelak akan dilihat oleh sang user website kita.

1. Tambahkan fungsi `view(<nama_view>)` pada setiap fungsi di controller.
2. Isilah nama view pada parameter fungsi view tersebut.

*MahasiswaController.php*

```
class MahasiswaController extends Controller
{
    function getIndex() {
        return view('mahasiswa');
    }
    function getTambah() {
        return view('mahasiswa_tambah');
    }
    function getUbah() {
        return view('mahasiswa_ubah');
    }
}
```

*HomeController.php*

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;

class HomeController extends Controller
{
    function getIndex() {
        return view('home');
    }
}
```

## Menyiapkan Assets (AdminLTE) untuk Membuat View

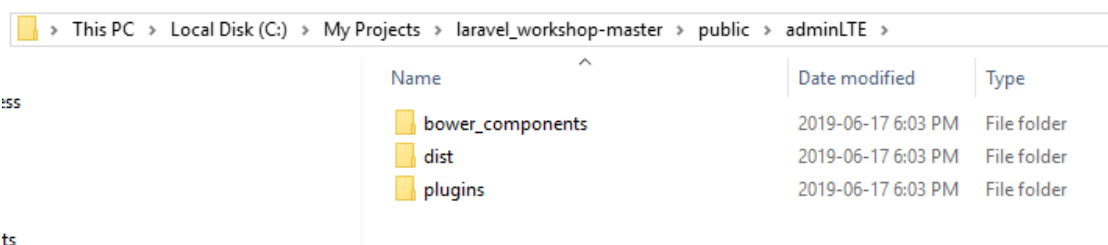
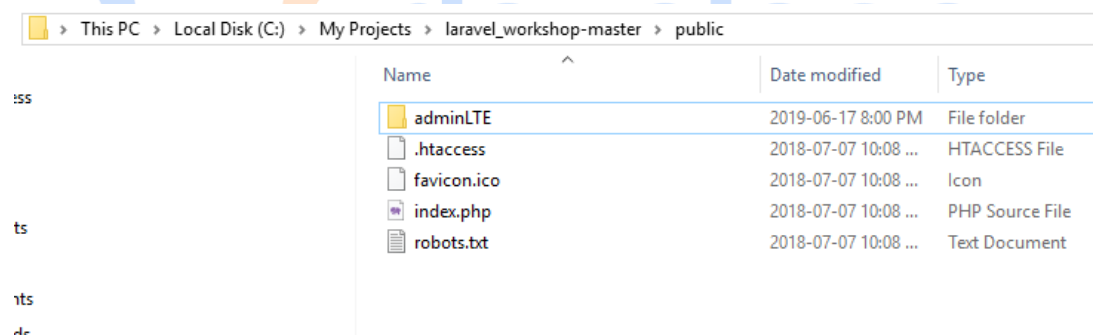
Terdapat assets eksternal (js, css, images, etc) yang harus disiapkan untuk membuat tampilan atau view yang kita inginkan, yaitu terutama assets yang berkaitan dengan

AdminLTE, didalam template AdminLTE yang kita gunakan mengandung berbagai macam assets terkait yang dibutuhkan agar template dapat berjalan dengan baik, untuk persiapan terlebih dahulu assets tersebut agar View yang kita buat nanti dapat menggunakannya.

Untuk dapat menggunakan eksternal assets (javascript atau css) tersebut maka kita diharuskan untuk meletakkan file javascript atau css external ke static file project Laravel kita, dan tempat meletakkan static file pada project Laravel adalah pada directory

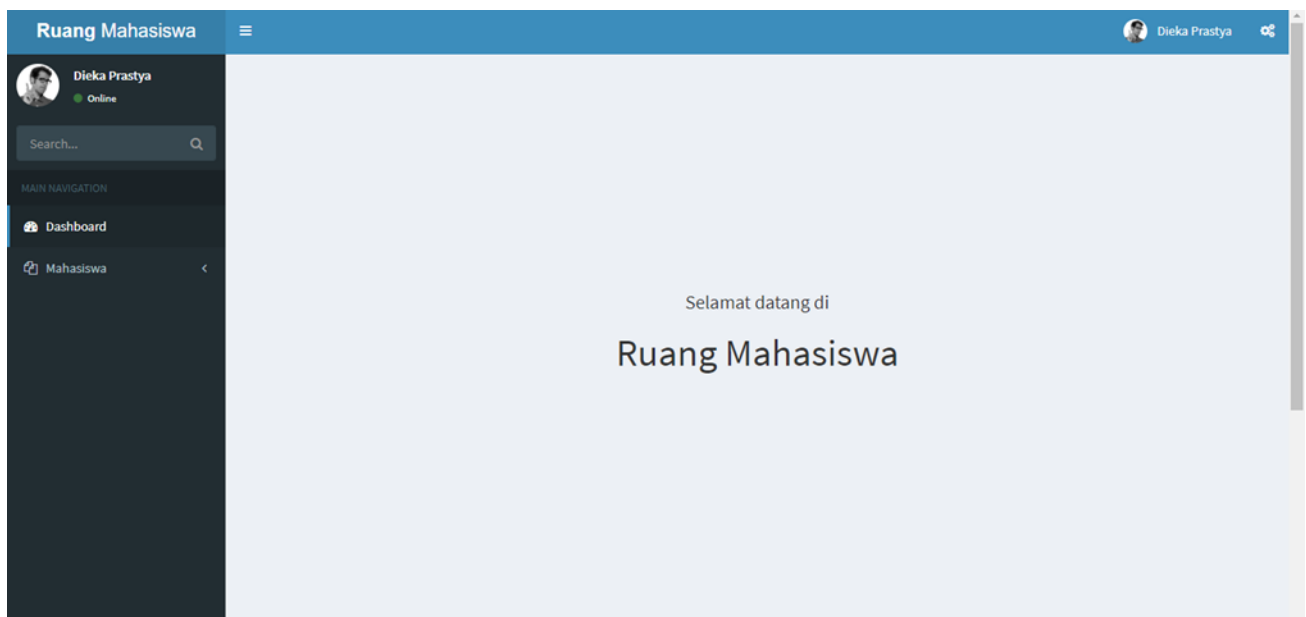
**"<directory\_project>/public/"**. Berikut langkah-langkahnya:

1. Download zip file AdminLTE (<http://adminlte.io>)
2. Copy folder dibawah dari hasil extract, kemudian pindahkan ke folder static file project Laravel
  - a. bower\_components
  - b. dist
  - c. plugins
3. Agar terlihat lebih rapih, maka kelompokkan assets pada folder khusus yang bernama adminLTE seperti dibawah ini.



### **Membuat Master / Layout View**

Pada template AdminLTE terdapat kesamaan antara setiap halamannya yang dapat dijadikan sebagai sebuah Layout dari setiap halaman, seperti Sidebar ataupun Navbar.



Navbar, sidebar, hingga footer (layout) digunakan di semua halaman pada aplikasi yang ingin kita bangun, agar layout tidak di design berulang-ulang pada setiap view (tampilan html), maka kita akan membuat view khusus yang bertugas untuk menangani layouting pada semua halaman, yang dimana layout tersebut akan menjadi global layout untuk semua view dibawahnya.

1. Untuk membuat file view, maka buatlah sebuah file php yang bernama **layout.blade.php** pada directory **"<project\_path>/resources/views/layout.blade.php"**.
2. Definisikan isi file **layout.blade.php** dengan tampilan html yang dijadikan view global, seperti dibawah ini:

```
<!DOCTYPE html>
<html>

<head>

    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Ruang Mahasiswa</title>

    <!-- Tell the browser to be responsive to screen width -->
    <meta content="width=device-width, initial-scale=1, maximum-scale=1, user-scalable=no" name="viewport">

    <!-- Bootstrap 3.3.7 -->
    <link rel="stylesheet" href="{{asset('adminLTE/bower_components/bootstrap/dist/css/bootstrap.min.css')}}">

    <!-- Font Awesome -->
    <link rel="stylesheet" href="{{asset('adminLTE/bower_components/font-awesome/css/font-awesome.min.css')}}">
```

```

<!-- Ionicons -->
<link rel="stylesheet" href="{{ asset('adminLTE/bower_components/Ionicons/css/ionicons.min.css') }}">
<!-- Theme style -->
<link rel="stylesheet" href="{{ asset('adminLTE/dist/css/AdminLTE.min.css') }}">
<!-- AdminLTE Skins. Choose a skin from the css/skins
      folder instead of downloading all of them to reduce the load. -->
<link rel="stylesheet" href="{{ asset('adminLTE/dist/css/skins/_all-skins.min.css') }}">
<!-- Google Font -->
<link rel="stylesheet"
      href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:300,400,600,700,300italic,400italic,600italic">
</head>

<body class="hold-transition skin-blue sidebar-mini">
  <div class="wrapper">
    @include('header')
    @include('sidebar')

    <!-- Content Wrapper. Contains page content -->
    <div class="content-wrapper">
      <!-- Content Header (Page header) -->
      <section class="content-header">
        @yield('content-header')
      </section>

      <!-- Main content -->
      <section class="content">
        @yield('content')
      </section>

      <!-- /.content -->
    </div>
    <!-- /.content-wrapper -->
    @include('footer')
  </div>
  <!-- /.wrapper -->

  <!-- jQuery 3 -->
  <script src="{{ asset('adminLTE/bower_components/jquery/dist/jquery.min.js') }}"></script>
  <!-- jQuery UI 1.11.4 -->
  <script src="{{ asset('adminLTE/bower_components/jquery-ui/jquery-ui.min.js') }}"></script>
  <!-- Resolve conflict in jQuery UI tooltip with Bootstrap tooltip -->
  <script>
    $.widget.bridge('uibutton', $.ui.button);
  </script>
  <!-- Bootstrap 3.3.7 -->
  <script src="{{ asset('adminLTE/bower_components/bootstrap/dist/js/bootstrap.min.js') }}"></script>
  <!-- Slimscroll -->
  <script src="{{ asset('adminLTE/bower_components/jquery-slimscroll/jquery.slimscroll.min.js') }}"></script>

```

```

<!-- FastClick -->
<script src="{{ asset('adminLTE/bower_components/fastclick/lib/fastclick.js') }}"></script>
<!-- AdminLTE App -->
<script src="{{ asset('adminLTE/dist/js/adminlte.min.js') }}"></script>
</body>
</html>

```

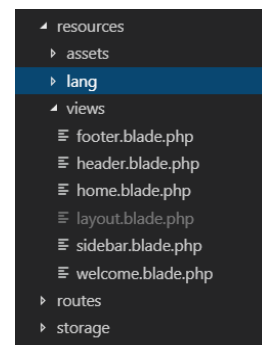
Pada view layout sendiri kita menggunakan beberapa Blade Templates, yaitu **asset()**, **route()**, **yield()**, dan **include()**.

- **asset(<static\_file\_path:string>)** : fungsi ini digunakan untuk membuat url otomatis dari path directory yang kita letakan pada directory public.
- **route(<route\_name:string>)**: fungsi ini digunakan untuk membuat url otomatis dari route yang telah dibuat dan diberi nama.
- **yield(<section\_name:string>)**: fungsi ini digunakan untuk menurunkan view kepada anaknya, yang nanti anaknya akan mengganti/mengisi bagian yang di yield dengan fungsi section.
- **include(<view\_name:string>)**: fungsi ini digunakan untuk menginclude sebuah view lain atau juga bisa dikatakan menggabungkan dengan view lain, dan diletakan pada tempat dimana fungsi include tersebut dipanggil.

### Membuat View Bagian-Bagian Layout

Pada tahap sebelumnya kita telah membuat layout master atau utama yang bernama **layout.blade.php**, didalam view layout tersebut menginclude bagian-bagian layout berupa **header**, **sidebar**, dan **footer**. Pada tahap ini kita akan membuat bagian-bagian dari terpisah dari layout tersebut:

1. Buatlah 3 view bagian-bagian dari layout, yaitu view **header.blade.php**, view **sidebar.blade.php**, dan view **footer.blade.php** pada directory views ("**<project\_path>/resources/views/**").



2. Definiskan view **header.blade.php** dengan bagian header yang akan digantikan pada **include('header')** pada **layout.blade.php**.

```
<header class="main-header">

    <!-- Logo -->
    <a href="{{ route('home') }}" class="logo">
        <span class="logo-mini"><b>R</b>M</span>
        <span class="logo-lg"><b>Ruang</b> Mahasiswa</span>
    </a>

    <!-- Header Navbar: style can be found in header.less -->
    <nav class="navbar navbar-static-top">
        <!-- Sidebar toggle button-->
        <a href="#" class="sidebar-toggle" data-toggle="push-menu" role="button">
            <span class="sr-only">Toggle navigation</span>
        </a>

        <div class="navbar-custom-menu">
            <ul class="nav navbar-nav">

                <!-- User Account: style can be found in dropdown.less -->
                <li class="dropdown user user-menu">
                    <a href="#" class="dropdown-toggle" data-toggle="dropdown">
                        
                        <span class="hidden-xs">Dieka Prastya</span>
                    </a>
                    <ul class="dropdown-menu">
                        <!-- User image -->
                        <li class="user-header">
                            
                            <p>
                                Dieka Prastya - Web Developer
                                <small>Member since Nov. 2012</small>
                            </p>
                        </li>
                        <!-- Menu Footer-->
                        <li class="user-footer">
                            <div class="pull-left">
                                <a href="#" class="btn btn-default btn-flat">Profile</a>
                            </div>
                            <div class="pull-right">
                                <a href="#" class="btn btn-default btn-flat">Sign out</a>
                            </div>
                        </li>
                    </ul>
                </li>

            </ul>
        </div>
    </nav>
</header>
```

3. Definiskan view **sidebar.blade.php** dengan bagian sidebar yang akan digantikan pada **include('sidebar')** pada **layout.blade.php**.

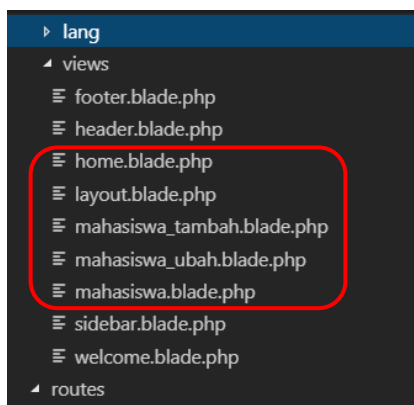
```
<aside class="main-sidebar">
  <section class="sidebar">
    <div class="user-panel">
      <div class="pull-left image">
        
      </div>
      <div class="pull-left info">
        <p>Dieka Prasty</p>
        <a href="#"><i class="fa fa-circle text-success"></i> Online</a>
      </div>
    </div>
    <form action="#" method="get" class="sidebar-form">
      <div class="input-group">
        <input type="text" name="q" class="form-control" placeholder="Search...">
        <span class="input-group-btn">
          <button type="submit" name="search" id="search-btn" class="btn btn-flat">
            <i class="fa fa-search"></i>
          </button>
        </span>
      </div>
    </form>
    <ul class="sidebar-menu" data-widget="tree">
      <li class="header">MAIN NAVIGATION</li>
      <li class="active treeview">
        <a href="{{ route('home') }}">
          <i class="fa fa-dashboard"></i> <span>Dashboard</span>
        </a>
      </li>
      <li class="treeview">
        <a href="{{ route('home') }}">
          <i class="fa fa-files-o"></i>
          <span>Mahasiswa</span>
          <span class="pull-right-container">
            <i class="fa fa-angle-left pull-right"></i>
          </span>
        </a>
        <ul class="treeview-menu">
          <li><a href="{{ route('mahasiswa') }}"><i class="fa fa-circle-o"></i> Daftar Mahasiswa</a></li>
          <li><a href="{{ route('mahasiswa_tambah') }}"><i class="fa fa-circle-o"></i> Tambah Mahasiswa</a></li>
        </ul>
      </li>
    </ul>
  </section>
</aside>
```

4. Definiskan view **footer.blade.php** dengan bagian footer yang akan digantikan pada **include('footer')** pada **layout.blade.php**.

```
<footer class="main-footer">
  <div class="pull-right hidden-xs">
    <b>Version</b> 1.0.0
  </div>
  <strong>Copyright &copy; 2019.</strong> All rights
  reserved.
</footer>
```

### Membuat View Tiap Halaman

1. Buatlah 4 view lain untuk mewakili tiap halaman pada directory views ("**<project\_path>/resources/views/**"), yaitu view **home**, **mahasiswa\_tambah**, **mahasiswa\_ubah**, dan **mahasiswa**.



2. Definiskan design html dari tiap halaman ke view yang telah dibuat, berikut design html nya:

[home.blade.php](#)

```
@extends('layout')
@section('title', 'Ruang Mahasiswa')

@section('content')
<br />
<h5 style="text-align:center;font-size: 20px; margin-top: 200px;">Selamat datang di</h5>
<h2 style="text-align:center;font-size: 40px;"> Ruang Mahasiswa </h2>
@endsection
```



```
@extends('layout')
@section('title', 'Tambah Mahasiswa')

@section('content')
<div class="row">
    <div class="col-md-12">
        <div class="box box-primary">
            <div class="box-header with-border">
                <h3 class="box-title">Tambah Mahasiswa</h3>
            </div>
            <div class="box-body">
                <form action="#" method="POST">
                    <div class="form-group row">
                        <label for="nama" class="col-sm-2 col-form-label">Nama</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" id="nama" name="nama"
                                placeholder="Masukkan nama anda..">
                        </div>
                    </div>
                    <div class="form-group row">
                        <label for="jurusan" class="col-sm-2 col-form-label">Jurusan</label>
                        <div class="col-sm-10">
                            <select class="form-control" name="jurusan">
                                <option value="SI">Sistem Informasi</option>
                                <option value="MI">Manajemen Informatika</option>
                                <option value="EN">Sastra Inggris</option>
                            </select>
                        </div>
                    </div>
                    <div class="form-group row">
                        <label class="col-form-label col-sm-2 pt-0">Jenis Kelamin</label>
                        <div class="col-sm-10">
                            <div class="form-check">
                                <input class="form-check-input" type="radio" name="jenis_kelamin" value="L" checked>
                                <label class="form-check-label">
                                    Laki-laki
                                </label>
                            </div>
                            <div class="form-check">
                                <input class="form-check-input" type="radio" name="jenis_kelamin" value="P">
                                <label class="form-check-label">
                                    Perempuan
                                </label>
                            </div>
                        </div>
                    </div>
                </form>
            </div>
        </div>
    </div>
</div>
```

```

        <div class="col-sm-2"></div>
        <div class="col-sm-10">
            <button type="submit" class="btn btn-primary">Daftarkan</button>
        </div>
    </div>
</Form>
</div>
</div>
</div>
</div>
@endsection

```

*mahasiswa\_ubah.blade.php*

```

@extends('layout')
@section('title', 'Ubah Mahasiswa')

@section('content')
<div class="row">
    <div class="col-md-12">
        <div class="box box-primary">
            <div class="box-header with-border">
                <h3 class="box-title">Ubah Mahasiswa</h3>
            </div>
            <div class="box-body">
                <form action="#" method="POST">
                    <input name="id" type="hidden" value="" />
                    <div class="form-group row">
                        <label for="nama" class="col-sm-2 col-form-label">Nama</label>
                        <div class="col-sm-10">
                            <input type="text" class="form-control" id="nama" name="nama"
                                placeholder="Masukkan nama anda.." value="">
                        </div>
                    </div>
                </div>
                <div class="form-group row">
                    <label for="jurusan" class="col-sm-2 col-form-label">Jurusan</label>
                    <div class="col-sm-10">
                        <select class="form-control" name="jurusan">
                            <option value="SI">
                                Sistem Informasi
                            </option>
                            <option value="MI">
                                Manajemen Informatika
                            </option>
                            <option value="EN">
                                Sastra Inggris
                            </option>
                        </select>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

```

```

        </div>
        <div class="form-group row">
            <label class="col-form-label col-sm-2 pt-0">Jenis Kelamin</label>
            <div class="col-sm-10">
                <div class="form-check">
                    <input class="form-check-input" type="radio" name="jenis_kelamin" value="L">
                    <label class="form-check-label">
                        Laki-laki
                    </label>
                </div>
                <div class="form-check">
                    <input class="form-check-input" type="radio" name="jenis_kelamin" value="P">
                    <label class="form-check-label">
                        Perempuan
                    </label>
                </div>
            </div>
        </div>
        <div class="form-group row">
            <div class="col-sm-2"></div>
            <div class="col-sm-10">
                <button type="submit" class="btn btn-primary">Simpan</button>
            </div>
        </div>
    </Form>
</div>
</div>
</div>
</div>
@endsection

```

*mahasiswa.blade.php*

```

@extends('layout')
@section('title', 'Daftar Mahasiswa')
@section('content-header')
<h1>
    Daftar Mahasiswa
    <small>informasi manajemen mahasiswa</small>
</h1>
<ol class="breadcrumb">
    <li><a href="#"><i class="fa fa-dashboard"></i> Home</a></li>
    <li class="active">Daftar Mahasiswa</li>
</ol>
@endsection
@section('content')
<div class="row">
    <div class="col-md-12">

```

```

<div class="box box-primary">
    <div class="box-header with-border">
        <h3 class="box-title">Daftar Mahasiswa</h3>
    </div>
    <div class="box-body">
        <div class="alert alert-success">
            Proses simpan berhasil
        </div>
        <a href="{{ route('mahasiswa_tambah') }}" class="btn btn-sm btn-info">Tambah Mahasiswa Baru</a>
        <br />
        <br />
        <table class="table table-bordered">
            <thead>
                <tr>
                    <th>Nama</th>
                    <th>Jurusan</th>
                    <th>Jenis Kelamin</th>
                    <th>Aksi</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>Dieka Prastya</td>
                    <td>Sistem Informasi</td>
                    <td>Laki-lak</td>
                    <td>
                        <a href="{{ route('mahasiswa_ubah') }}"
                            class="btn btn-sm btn-primary">Ubah</a>
                        <button name="id" value=""
                            class="btn btn-sm btn-danger">Hapus</a>
                    </td>
                </tr>
            </tbody>
        </table>
    </div>
</div>
</div>
@endsection

```

### Keterangan Blade Templates:

- **@extends(<nama\_view\_layout\_parent>):** fungsi ini digunakan untuk menjadikan view tersebut menjadi child dari view yang didefinisikan pada fungsi extends, contohnya: terdapat `extends('layout')` pada baris kode file view

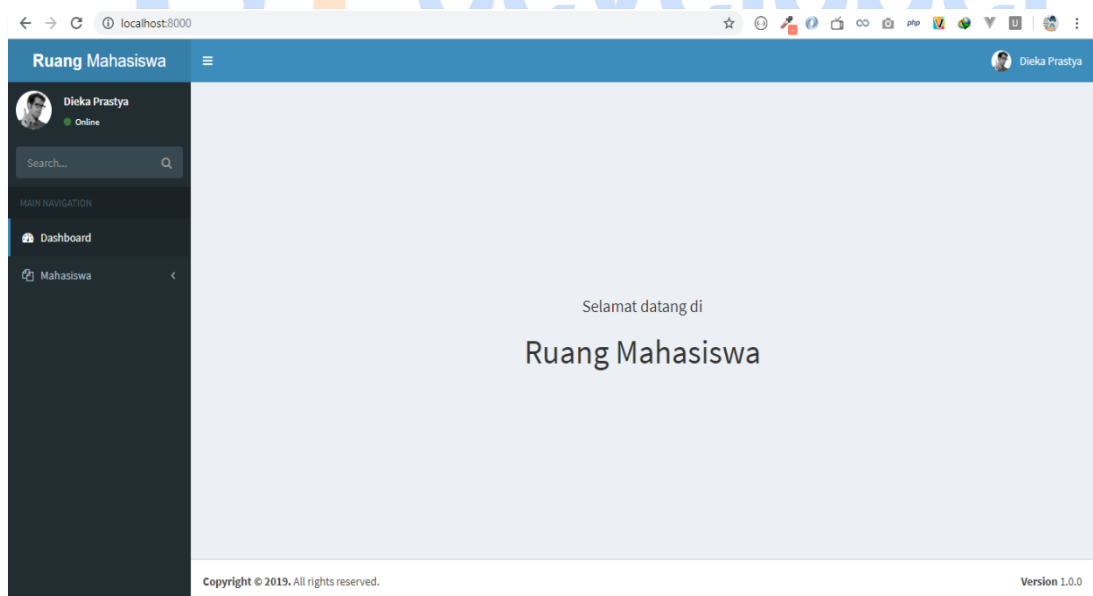
mahasiswa.blade.php, yang artinya view mahasiswa.blade.php adalah anak dari view layout.blade.php.

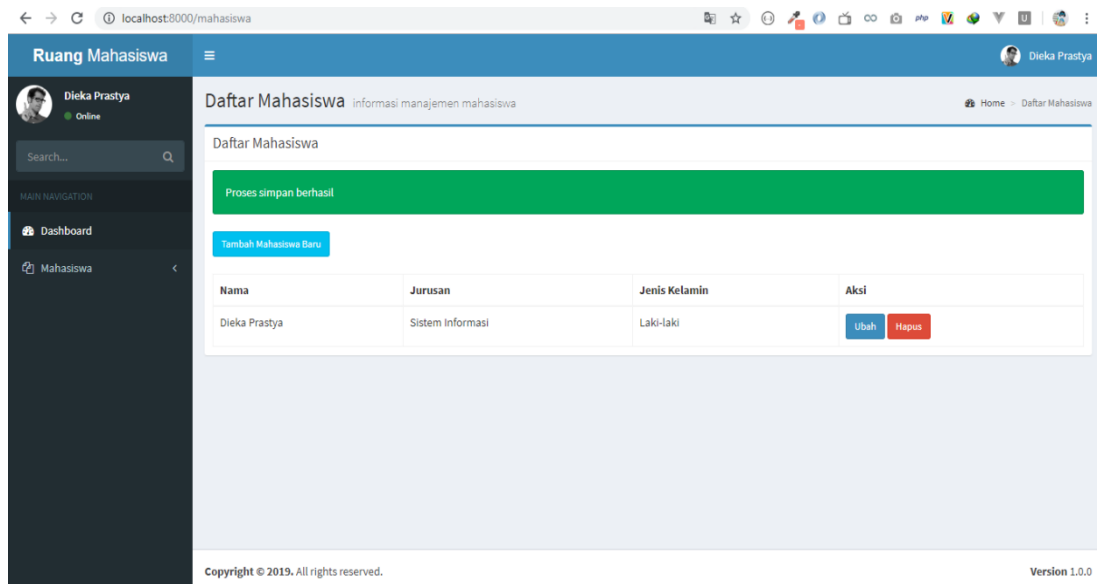
- **@section(<nama\_section>) & endsection:** fungsi ini digunakan untuk menggantikan fungsi yield yang telah didefinisikan sebelumnya pada halaman parent, contohnya @section('content') yang artinya isi dari section tersebut akan menggantikan @yield('content') pada parent view yang telah di extends sebelumnya.

### Memastikan Setiap Halaman telah Berhasil Diakses

Sampai sejauh ini mengikuti kaidah dari MVC, kita telah membuat routes, kemudian controller & fungsinya, dan yang terakhir adalah view, maka secara kaidah MVC seharusnya telah terbentuk halaman-halaman berdasarkan route, fungsi, dan view yang telah kita buat tersebut, dan pastinya jika halaman telah berhasil dibuat maka user sudah dapat mengakses halaman-halaman tersebut.

1. Pastikan **php artisan serve (Web Server PHP)** telah berjalan sebelumnya.
2. Jika sudah berjalan maka bukalah halaman home dan cobalah untuk menjelajahi setiap halaman lain pada aplikasi.





3. Jika halaman telah berhasil terbuka & tampilan pun berjalan dengan baik, mari lanjutkan ke tahap berikutnya. Namun jika masih ada kekurangan cobalah untuk periksa kembali tutorial sebelumnya.

## 5. Membuat Database dengan Laravel Database Migration

Pada Laravel terdapat library khusus berkenaan dengan perancangan database, library ini disebut dengan **Laravel Database Migration**, Laravel Database Migration sama seperti version control untuk database, membuat anda dapat membuat, berbagi, ataupun memodifikasi skema database melalui framework Laravel, dengan menggunakan library ini akan sangat terasa sekali kemudahannya ketika kalian bekerja secara team, salah satunya kalian dapat melakukan perubahan database secara bersama-sama tanpa harus melakukan perubahan pada masing-masing database lokal, dan setelah perubahan dilakukan anda tinggal mensinkronisasikan database lokal anda dengan script schema yang telah didefinisikan menggunakan library Laravel Database Migration.

Schema Database yang akan dibuat:

Table: mahasiswa

Field Name	Type	Attribute
id	int	primary_key, auto_increments

nama	varchar(255)	
jurusan	varchar(2)	
jenis_kelamin	char(1)	

## Membuat Schema Migration

Berikut langkah-langkah untuk membuat schema database melalui Laravel Database Migration:

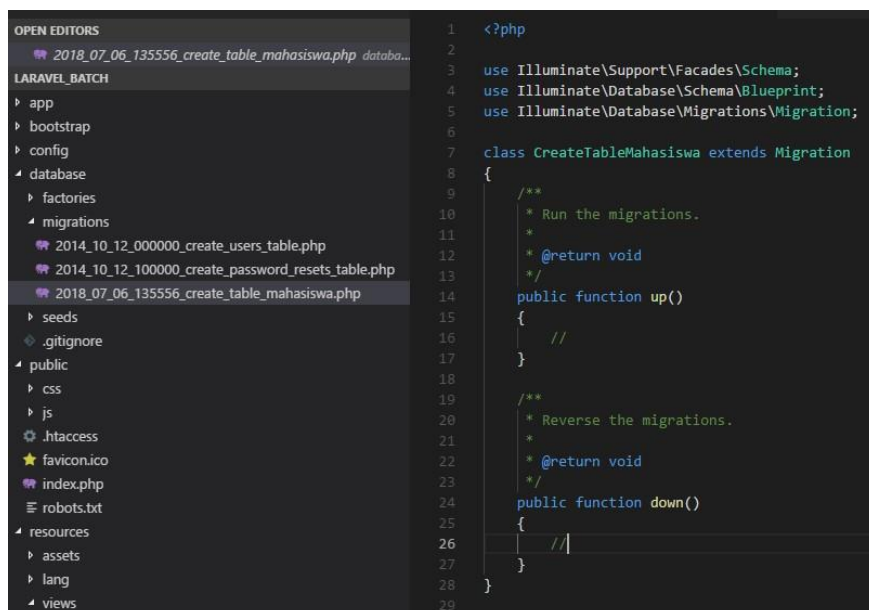
1. Buka Command Prompt, dan masuk ke directory project Laravel.
2. Jalankan perintah **php artisan make:migration create\_table\_mahasiswa**.

```
C:\Users\dieks\Projects\laravel_batch>php artisan make:migration create_table_mahasiswa
Created Migration: 2018_07_06_135556_create_table_mahasiswa
```

Perintah **php artisan make:migration <nama\_migration>** adalah perintah shortcut dari Laravel CLI untuk membuat schema migration baru pada project Laravel, file migration akan otomatis terbuat pada directory

“<project\_path>/database/migrations/<tanggal\_nama\_migration>”

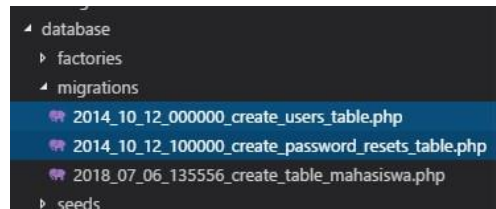
3. Jika berhasil, maka akan terbentuk migration baru pada directory migration, dengan nama create\_table\_mahasiswa.



```

1  <?php
2
3  use Illuminate\Support\Facades\Schema;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Database\Migrations\Migration;
6
7  class CreateTableMahasiswa extends Migration
8  {
9      /**
10       * Run the migrations.
11       *
12       * @return void
13       */
14      public function up()
15      {
16          //
17      }
18
19      /**
20       * Reverse the migrations.
21       *
22       * @return void
23       */
24      public function down()
25      {
26          //
27      }
28  }
29  
```

4. Hapuslah terlebih dahulu file migration bawaan dari Laravel dikarenakan pada aplikasi ini tidak akan menggunakan schema tersebut, file bawaan tersebut adalah **2014\_10\_12\_000000\_create\_users\_table.php** dan **2014\_10\_12\_100000\_create\_password\_resets\_table.php**.



5. Ubah dan tambahkan baris kode dibawah ini pada file migration create\_table\_mahasiswa fungsi up().

```
public function up()
{
    Schema::create('mahasiswa', function(Blueprint $table) {
        $table->increments('id');
        $table->string('nama');
        $table->string('jurusan', 2);
        $table->char('jenis_kelamin', 1);
    });
}
```

**Library atau class Schema** adalah library yang digunakan Laravel Migration untuk membuat hingga memodifikasi table pada database, mulai dari create, alter, drop, dan lain-lainnya yang berkaitan akan schema table pada database.

Salah satu fungsi pada schema yang kita gunakan saat ini adalah fungsi **create**, fungsi create adalah fungsi yang terkhusus untuk membuat sebuah table beserta field-field nya pada database. Contoh untuk mendefinisikannya adalah sebagai berikut:

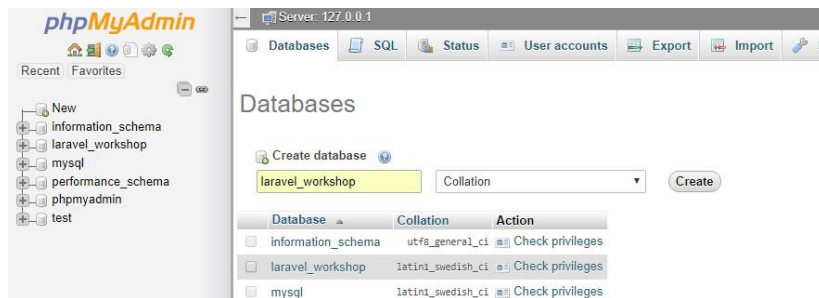
```
public function up()
{
    // Membuat Table Baru
    Schema::create('<nama_table>', function(Blueprint $table) {
        $table->increments('id'); // field id, auto increments, primary
        $table->string('nama'); // field nama, varchar(255)
        $table->string('jurusan', 2); // field jurusan, varchar(2)
        $table->char('jenis_kelamin', 1); // field jenis_kelamin char(1)
    });
}
```



## Menyiapkan Database untuk di Migrasikan

Proses migrasi tidaklah otomatis membuat database, sebelum melakukan proses migrasi database diperlukan database kosong sebagai wadah Laravel Migration untuk membuat table-table sesuai schema yang telah dirancang pada aplikasi. Berikut langkahnya:

1. Buatlah database kosong pada server MySQL anda dengan nama **laravel\_workshop**.



## Mengatur Koneksi Database dari Aplikasi

Sebelum memulai migration, pastikan koneksi database telah terkonfigurasi dengan benar mengarah ke alamat server mysql local kita, dan kearah database **laravel\_workshop**.

1. Bukalah file **.env** pada directory "`<project_path>/`**.env**".
2. Atur **DB\_CONNECTION**, **DB\_HOST**, **DB\_PORT**, **DB\_DATABASE**, **DB\_USERNAME**, dan **DB\_PASSWORD** dengan benar sesuai dengan alamat server & database mysql yang ingin dimigrasikan.

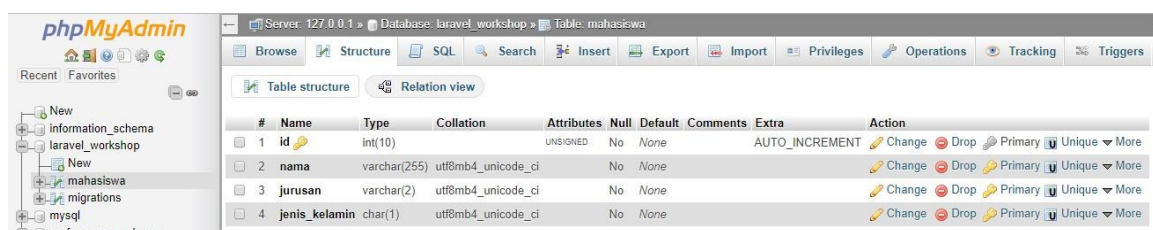
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel_workshop
DB_USERNAME=root
DB_PASSWORD=
```

## Memulai Proses Migration

Untuk memulai migration, kita hanya perlu menjalankan perintah pada Laravel CLI seperti biasanya, dengan menjalankan perintah **php artisan migrate**.

```
C:\Users\dieks\Projects\laravel_batch>php artisan migrate
Migration table created successfully.
Migrating: 2018_07_06_135556_create_table_mahasiswa
Migrated: 2018_07_06_135556_create_table_mahasiswa
```

Jika proses migration berhasil akan muncul informasi seperti diatas, dan kita juga dapat melihat langsung pada server mysql nya dengan terbentuknya table mahasiswa dengan field yang telah didefinisikan pada schema, dan terdapat table tambahan dari Laravel Migration yang bernama table migrations, yang dimana itu berfungsi untuk mencatat log version control migrations.



#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(10)	utf8mb4_unicode_ci	UNSIGNED	No	None		AUTO_INCREMENT	Change Drop Primary Unique More
2	nama	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop Primary Unique More
3	jurusan	varchar(2)	utf8mb4_unicode_ci		No	None			Change Drop Primary Unique More
4	jenis_kelamin	char(1)	utf8mb4_unicode_ci		No	None			Change Drop Primary Unique More

Proses migration selesai!!

## 6. Menyelesaikan Halaman Mahasiswa Tambah (Create / Insert)

Pada tahap keenam ini kita akan menyelesaikan halaman mahasiswa tambah, sejauh ini halaman mahasiswa tambah hanyalah baru bisa dapat dibuka pada halamannya saja, namun masih tidak dapat digunakan untuk melakukan penambahan data mahasiswa.

Proses penambahan mahasiswa dilakukan setelah user menginput data isian berupa nama, jurusan, dan jenis kelamin, baru setelah itu user melakukan proses submit dengan menekan tombol “Daftarkan”. Pada proses klik daftarkan itu, akan terjadi pengiriman data berupa request input yang telah diinput oleh user, data yang diinput akan dikirimkan dengan http request dengan method “POST” ke server aplikasi kita.

Http request dari user selalu mengikuti kaidah dari MVC, yaitu harus selalu melalui Route, kemudian diproses oleh Controller, lalu kemudian user diberikan kepada sebuah View untuk nanti dikembalikan kepada user kembali. Untuk kita menerima dan menyimpan data input dari user maka kita diwajibkan untuk membuat Route & Fungsi Controller lain yang terkhusus hanya untuk menerima input & menyimpan ke database dari apa yang user inputkan itu dengan method “POST” nya. **Membuat Route POST**

1. Buka file konfigurasi route (“<project\_path>/routes/web.php”).

2. Tambahkan baris kode baru untuk menambahkan route khusus untuk menerima POST.

```
Route::post('/mahasiswa/tambah', 'MahasiswaController@postTambah')->name('mahasiswa_tambah_post');
```

### Membuat Fungsi Controller untuk Menyimpan ke Database

1. Buka file MahasiswaController pada  
“<project\_path>/app/Http/Controllers/MahasiswaController.php”.
2. Tambahkan fungsi baru didalam class yang bernama **postTambah()**.
3. Lalu tambahkan baris kode seperti dibawah ini untuk melakukan penyimpanan ke database didalam fungsi **postTambah()**.

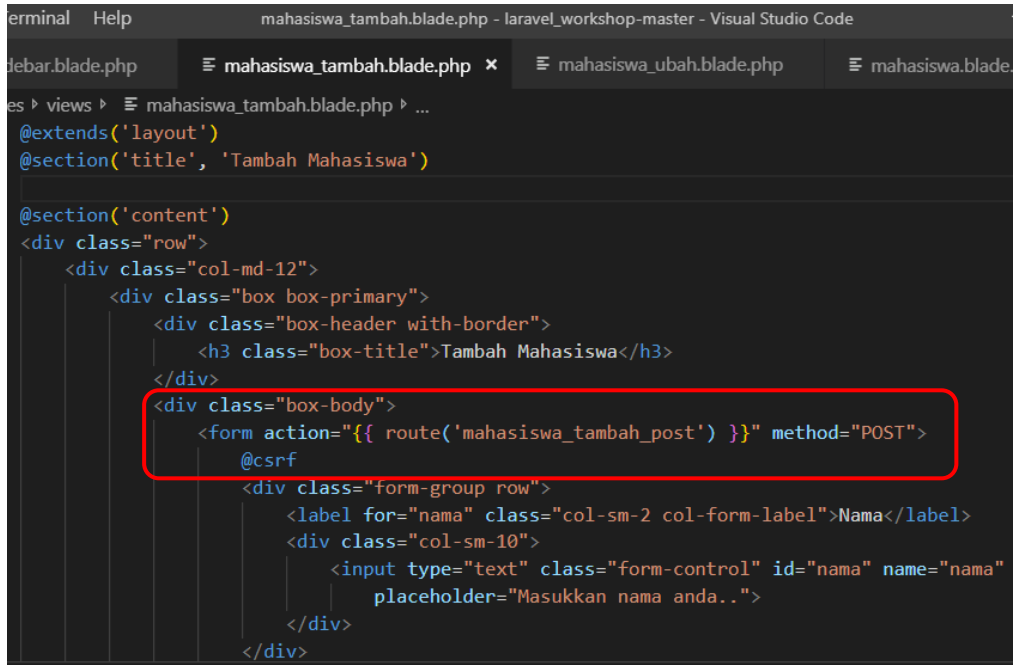
```
function postTambah() {  
    DB::table('mahasiswa')->insert([  
        'nama' => $request->input('nama'),  
        'jenis_kelamin' => $request->input('jenis_kelamin'),  
        'jurusan' => $request->input('jurusan')  
    ]);  
}
```

Kode baris diatas adalah untuk melakukan penyimpanan atau insert ke table mahasiswa, dengan field-field yang diisi dengan request input dari user. Pada proses penyimpanan tersebut memerlukan library DB untuk menjalankannya, untuk itu perlu dipanggil terlebih dahulu pula library DB pada file tersebut untuk menggunakannya.

4. Tambahkan baris kode “**use Illuminate\Support\Facades\DB**” untuk syarat penggunaan atau pemanggilan library DB.

```
<?php  
  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
use Illuminate\Support\Facades\DB;  
  
class MahasiswaController extends Controller  
{  
    //
```

- Ubah view **mahasiswa\_tambah.php** agar mengarahkan post action formnya ke route **MahasiswaController@postTambah()**, dan tambahkan blade template **@csrf** untuk menggenerate field input csrf demi keamanan yang lebih baik.



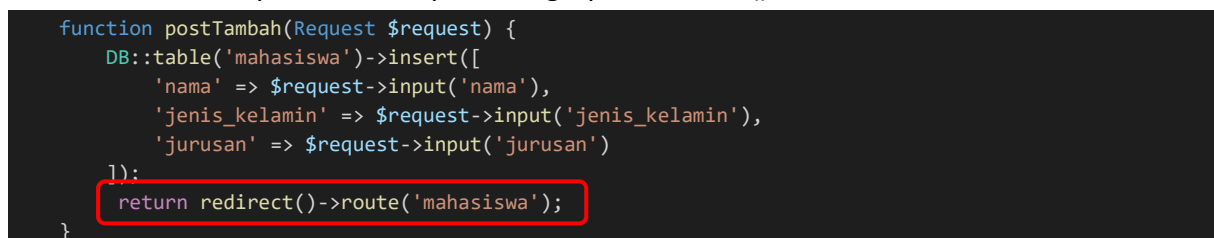
```
terminal Help mahasiswa_tambah.blade.php - laravel_workshop-master - Visual Studio Code

debar.blade.php  mahasiswa_tambah.blade.php x  mahasiswa_ubah.blade.php  mahasiswa.blade.p

es ▸ views ▸ mahasiswa_tambah.blade.php ▸ ...
@extends('layout')
@section('title', 'Tambah Mahasiswa')

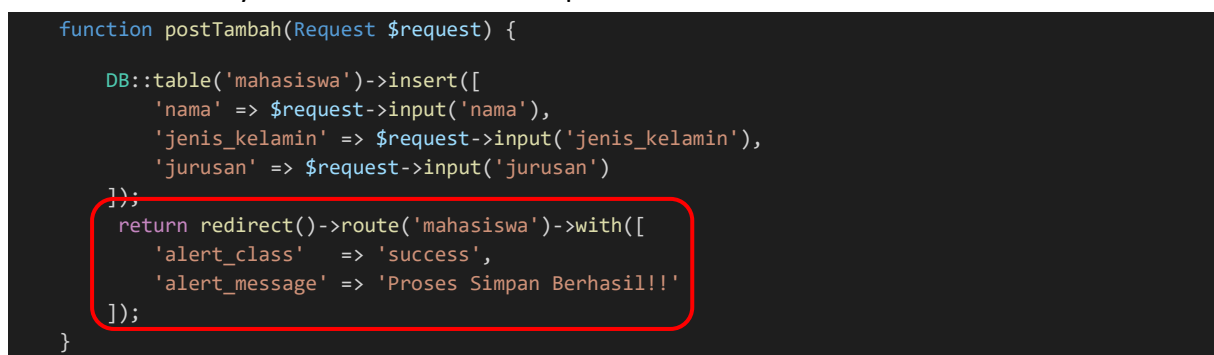
@section('content')
<div class="row">
  <div class="col-md-12">
    <div class="box box-primary">
      <div class="box-header with-border">
        <h3 class="box-title">Tambah Mahasiswa</h3>
      </div>
      <div class="box-body">
        <form action="{{ route('mahasiswa_tambah_post') }}" method="POST">
          @csrf
          <div class="form-group row">
            <label for="nama" class="col-sm-2 col-form-label">Nama</label>
            <div class="col-sm-10">
              <input type="text" class="form-control" id="nama" name="nama"
                placeholder="Masukkan nama anda..">
            </div>
          </div>
        </div>
      </div>
    </div>
  </div>
</div>
```

- Sampai sini proses fungsi penyimpanan sudah berjalan, dan tereksekusi.
- Namun user masih belum menerima feedback setelah melakukan penyimpanan, maka tambahkan proses Redirect yang mengarah ke halaman daftar mahasiswa setelah proses insert pada fungsi postTambah().



```
function postTambah(Request $request) {
    DB::table('mahasiswa')->insert([
        'nama' => $request->input('nama'),
        'jenis_kelamin' => $request->input('jenis_kelamin'),
        'jurusan' => $request->input('jurusan')
    ]);
    return redirect()->route('mahasiswa');
}
```

- Agar proses simpan memiliki alert yang muncul setelah redirect, maka tambahkan flash session yang dikirim ke halaman daftar mahasiswa, yang nantinya alert tersebut muncul pada view daftar mahasiswa.



```
function postTambah(Request $request) {
    DB::table('mahasiswa')->insert([
        'nama' => $request->input('nama'),
        'jenis_kelamin' => $request->input('jenis_kelamin'),
        'jurusan' => $request->input('jurusan')
    ]);
    return redirect()->route('mahasiswa')->with([
        'alert_class' => 'success',
        'alert_message' => 'Proses Simpan Berhasil!!'
    ]);
}
```

*mahasiswa.blade.php*

```
<h2> Daftar Mahasiswa </h2>
<br />
@if(session('alert_message'))
<div class="alert alert-{{session('alert_class')}}">
    {{ session('alert_message') }}
</div>
@endif
```

## 9. Fungsi Halaman Tambah Selesai!!

## 7. Menyelesaikan Fungsi Pengambilan Data Halaman Daftar Mahasiswa

Pada halaman daftar mahasiswa akan langsung menampilkan daftar mahasiswa yang ada di database ketika awal user membuka halaman daftar mahasiswa via web browser-nya, maka proses pengambilan data daftar mahasiswa kita simpulkan tetaplah berada fungsi getIndex() di MahasiswaController. Berikut langkah-langkah untuk mengambil data dari database pada Controller, yang kemudian ditampilkan pada view.

### Mengambil Data dari Database dan Menampilkannya

1. Buka file **MahasiswaController**.
2. Tambahkan pada fungsi getIndex() baris kode untuk mengambil data dari database table mahasiswa, kemudian Lakukan pengiriman data yang diambil dari database ke view mahasiswa untuk ditampilkan pada table.

*MahasiswaController.php*

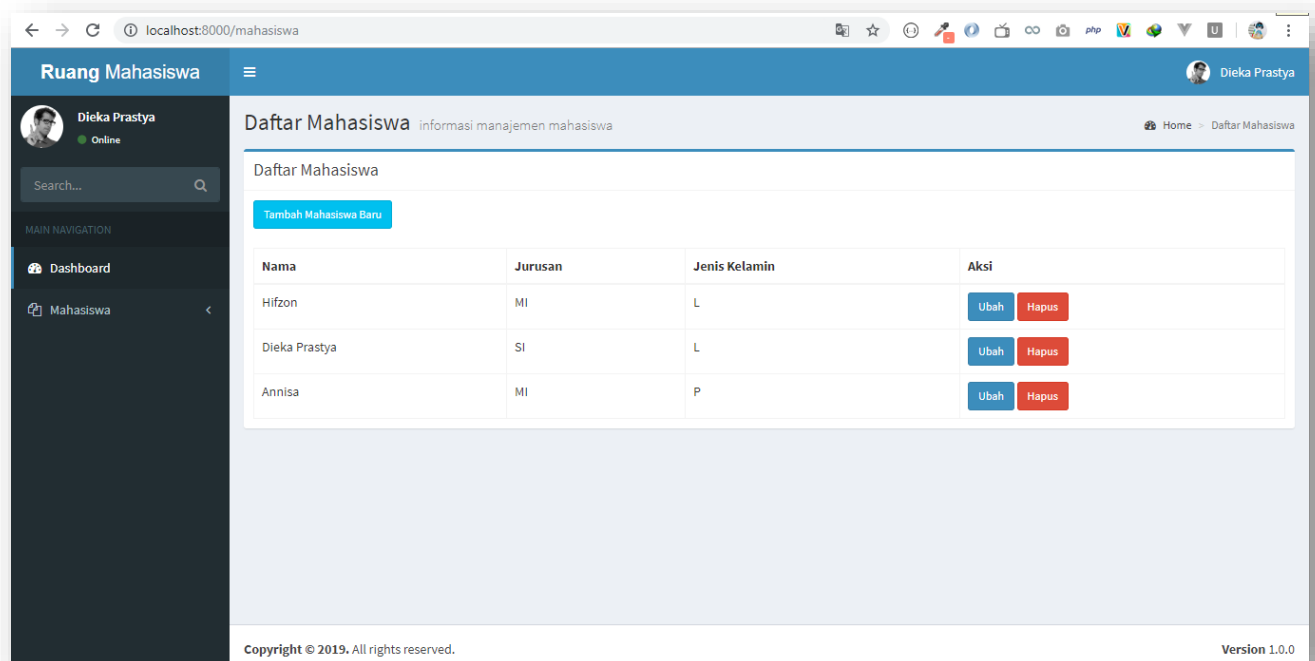
```
function getIndex() {
    $lists = DB::table('mahasiswa')->get();
    return view('mahasiswa', [
        'lists' => $lists
    ]);
}
```

3. Kemudian pada view **mahasiswa.blade.php** gunakan fungsi @foreach() untuk agar tabel pada view dapat menampilkan data yang dikirimkan dari controller.

```
<table class="table table-bordered">
    <thead>
        <tr>
            <th>Nama</th>
            <th>Jurusan</th>
            <th>Jenis Kelamin</th>
            <th>Aksi</th>
        </tr>
    </thead>
    <tbody>
```

```
@foreach($lists as $row)
<tr>
<td>{{ $row->nama }}</td>
<td>{{ $row->jurusan }}</td>
<td>{{ $row->jenis_kelamin }}</td>
<td>
<a href="{{ route('mahasiswa_ubah') }}"
class="btn btn-sm btn-primary">Ubah</a>
<button name="id" class="btn btn-sm btn-danger">Hapus</a>
</td>
</tr>
@endforeach
</tbody>
</table>
```

4. Dengan ini maka data telah berhasil dimunculkan pada halaman daftar mahasiswa.



5. Hanya saja pada data yang ditampilkan pada table adalah data mentah yang belum sama sekali diproses, sehingga pada field jurusan & jenis kelamin masihlah menampilkan data mentah (L/P) pada kolom data jenis kelamin dan (SI/MI/EN) pada kolom data jurusan.

## Memproses View Jurusan & Jenis Kelamin

1. Buatlah variabel array jurusan & jenis kelamin yang mendefinisikan nilai (key) dan keterangannya pada fungsi **getIndex()** di **MahasiswaController**, lalu kirimkan array \$jurusan dan \$jenis\_kelamin ke view mahasiswa.

```
function getIndex() {
    // mengambil data dari database table mahasiswa
    $lists = DB::table('mahasiswa')->get();

    //membuat variabel array dengan key & keterangan jurusan
    $jurusan = [
        'SI' => 'Sistem Informasi',
        'MI' => 'Manajemen Informatika',
        'EN' => 'Sastra Inggris',
    ];

    //membuat variabel array dengan key & keterangan jenis kelamin
    $jenisKelamin = [
        'L' => 'Laki-laki',
        'P' => 'Perempuan'
    ];

    //memberikan feedback ke user berupa tampilan (view)
    //beserta data variabel list mahasiswa, jenis kelamin kedalam view
    return view('mahasiswa', [
        'lists' => $lists,
        'jenisKelamin' => $jenisKelamin,
        'jurusan' => $jurusan
    ]);
}
```

2. Pada view “mahasiswa” ubah kolom jurusan & jenis kelamin pada table sehingga menggunakan array yang mengambil key dari variabel jurusan & jenis kelamin seperti dibawah ini:

```
@foreach($lists as $row)
<tr>
    <td>{{ $row->nama }}</td>
    <td>{{ $jurusan[$row->jurusan] }}</td>
    <td>{{ $jenisKelamin[$row->jenis_kelamin] }}</td>
    <td>
        <a href="{{ route('mahasiswa_ubah') }}"
            class="btn btn-sm btn-primary">Ubah</a>
        <button name="id">
```

```

class="btn btn-sm btn-danger">Hapus</a>

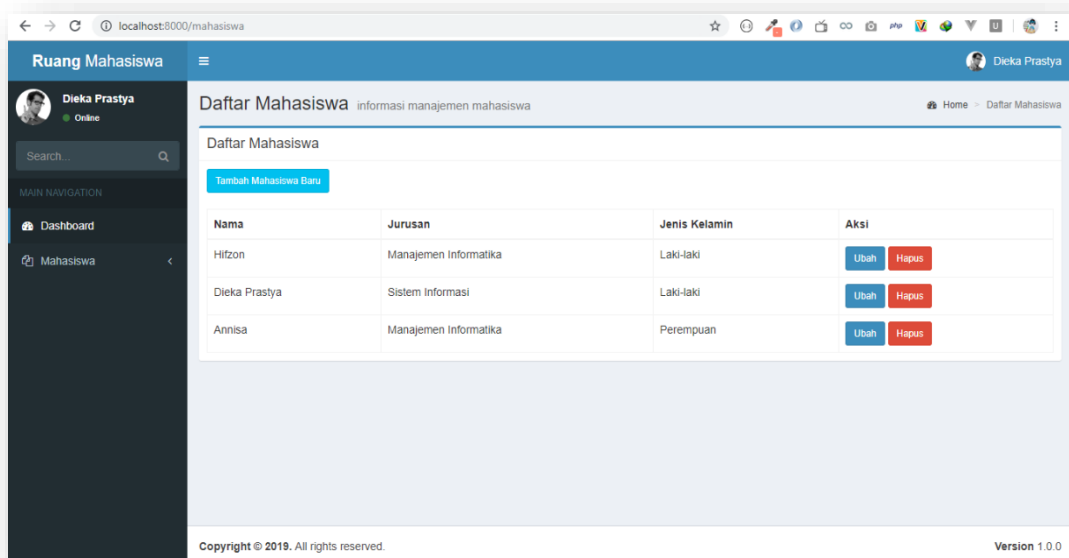
</td>

</tr>

@endforeach

```

4. Sampai pada tahap ini kita telah menyelesaikan halaman list atau daftar mahasiswa.



## 8. Menyelesaikan Fungsi Penghapusan Data pada Halaman Daftar Mahasiswa

Sama seperti fungsi tombol “Daftarkan” pada halaman penambahan mahasiswa, tombol “Hapus” pada halaman daftar mahasiswa juga menggunakan POST untuk proses mengirim kode id yang akan dihapus, oleh karena itu dibutuhkan sebuah fungsi controller & route baru untuk melakukan penghapusan data mahasiswa. **Membuat Route POST Hapus**

1. Buka file konfigurasi route (“<project\_path>/routes/web.php”).
2. Tambahkan baris kode baru untuk menambahkan route khusus menerima POST untuk menghapus.

```

Route::post('/mahasiswa/tambah', 'MahasiswaController@postTambah')->name('mahasiswa_tambah_post');
Route::post('/mahasiswa/hapus', 'MahasiswaController@postHapus')->name('mahasiswa_hapus_post');

```



## Membuat Fungsi Controller untuk Menghapus Mahasiswa

1. Buka file **MahasiswaController** pada  
“<project\_path>/app/Http/Controllers/MahasiswaController.php”.
2. Tambahkan fungsi baru yang bernama **postHapus()**.
3. Lalu tambahkan baris kode seperti dibawah ini untuk menjadikan fungsi postHapus() dapat melakukan penghapusan ke database berdasarkan input id, dan dapat memberikan feedback ke user berupa redirect ke halaman dan pesan sukses.

```
function postHapus(Request $request) {  
    //untuk melakukan proses penghapusan ke database berdasarkan input (id)  
    DB::table('mahasiswa')  
        ->where('id', $request->input('id'))  
        ->delete();  
  
    //untuk mengembalikan feedback ke user berupa redirect & message  
    return redirect()->route('mahasiswa')->with([  
        'alert_class' => 'success',  
        'alert_message' => 'Proses Hapus Berhasil!!'  
    ]);  
}
```

4. Ubah view **mahasiswa.blade.php**, dan buat agar ketika button diklik maka fungsi postHapus() pada MahasiswaController berjalan sesuai dengan tombol pada baris data yang diklik.

```
<form action="{{ route('mahasiswa_hapus_post') }}" method="post" id="formHapus">  
    @csrf  
</form>
```

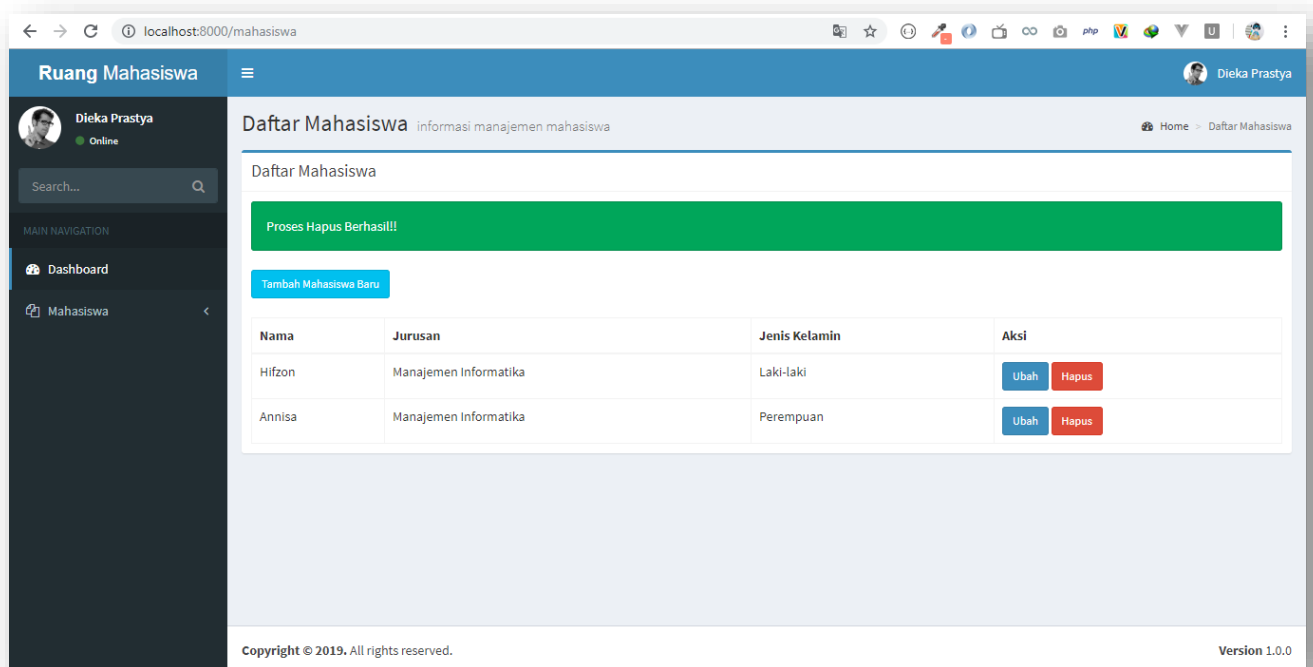
```
<table class="table table-bordered">  
    <thead>  
        <tr>  
            <th>Nama</th>  
            <th>Jurusan</th>  
            <th>Jenis Kelamin</th>  
            <th>Aksi</th>  
        </tr>  
    </thead>  
    <tbody>  
        @foreach($lists as $row)  
            <tr>  
                <td>{{ $row->nama }}</td>  
                <td>{{ $jurusan[$row->jurusan] }}</td>
```

```

<td>{{ $jenisKelamin[$row->jenis_kelamin] }}</td>
<td>
    <a href="{{ route('mahasiswa_ubah') }}"
        class="btn btn-sm btn-primary">Ubah</a>
    <button name="id" value="{{ $row->id }}" form="formHapus"
        class="btn btn-sm btn-danger">Hapus</a>
</td>
</tr>
@endforeach
</tbody>
</table>

```

5. Sampai sini proses fungsi penghapusan data mahasiswa di database seharusnya sudah dapat dilakukan.



## 9. Menyelesaikan Fungsi Halaman Perubahan Data Mahasiswa

Pada halaman perubahan data mahasiswa sejauh ini hanyalah baru bisa diakses saja tanpa ada fungsional untuk dapat melakukan perubahan mahasiswa, untuk itu mari di tahap ini kita selesaikan fungsional daripada halaman perubahan data mahasiswa agar bekerja sesuai dengan fungsi yang kita inginkan.

Berdasarkan mekanismenya, langkah awal untuk dapat menerapkan fungsi perubahan data mahasiswa ini kita menggunakan 2 langkah yang dimana ke-2 langkah tersebut memerlukan akses ke database dalam prosesnya.

Yang **pertama** ketika user sedang berada pada halaman daftar mahasiswa, kemudian sang user mengklik tombol ubah pada salah satu data mahasiswa, maka seketika browser akan diarahkan ke halaman perubahan data mahasiswa yang menampilkan inputan yang sudah beserta dengan data mahasiswa yang sesuai dengan yang dipilih, pada tahap tersebut diperlukan fungsi tersendiri yang saat ini menjadi asal usul dari fungsi `getUbah()` pada `MahasiswaController`.

Yang **kedua** ketika user klik tombol simpan pada halaman perubahan user setelah menginput data yang ingin diubah. Pada proses ini akan menjalankan proses update data mahasiswa ke dalam database.

### Proses 1: Mengambil Data Mahasiswa dan Menampilkannya pada Input

1. Buka file `MahasiswaController`.
2. Tambahkan baris kode dibawah ini pada fungsi `getUbah()` untuk mengambil data table mahasiswa pada database.

```
function getUbah(Request $request) {  
    //Mengambil data mahasiswa berdasarkan id  
    $data = DB::table('mahasiswa')  
        ->where('id', $request->query('id'))  
        ->first();  
  
    //Memberikan response feedback ke user berupa view mahasiwa_ubah  
    //beserta dengan data variabel $mahasiswa yang diisi dengan data  
    return view('mahasiswa_ubah', [  
        'mahasiswa' => $data  
    ]);  
}
```

1. Lakukan pengiriman data yang diambil dari database (\$mahasiswa) ke view mahasiswa untuk ditampilkan pada inputan.
2. Pada view **mahasiswa\_ubah.blade.php**, echo atau isikan tiap value inputan sesuai dengan fieldnya masing-masing.

- Input Id (Hidden)

```
<input name="id" type="hidden" value="{{ $mahasiswa->id }}" />
```

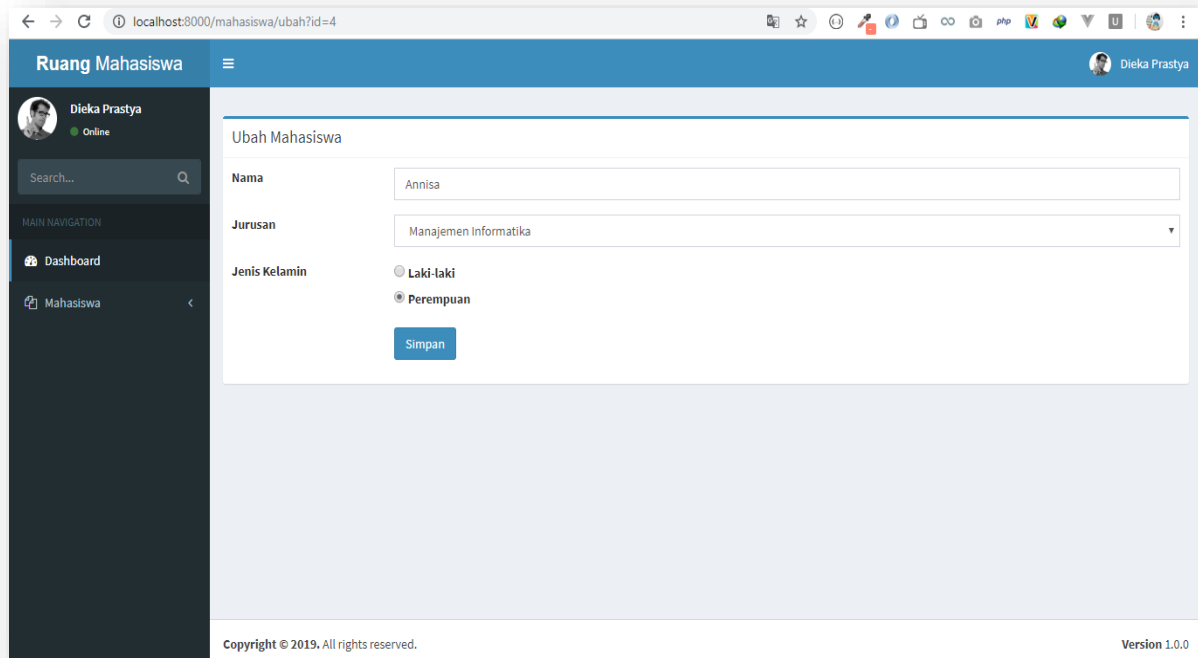
- Input Jurusan (Select)

```
<select class="form-control" name="jurusan">
  <option value="SI" {{ $mahasiswa->jurusan == 'SI' ? 'selected' : '' }}>
    Sistem Informasi
  </option>
  <option value="MI" {{ $mahasiswa->jurusan == 'MI' ? 'selected' : '' }}>
    Manajemen Informatika
  </option>
  <option value="EN" {{ $mahasiswa->jurusan == 'EN' ? 'selected' : '' }}>
    Sastra Inggris
  </option>
</select>
```

- Input Jenis Kelamin (Radio)

```
<div class="form-group row">
  <label class="col-form-label col-sm-2 pt-0">Jenis Kelamin</label>
  <div class="col-sm-10">
    <div class="form-check">
      <input class="form-check-input" type="radio" name="jenis_kelamin" value="L"
        {{ $mahasiswa->jenis_kelamin == 'L' ? 'checked' : '' }}>
      <label class="form-check-label">
        Laki-laki
      </label>
    </div>
    <div class="form-check">
      <input class="form-check-input" type="radio" name="jenis_kelamin" value="P"
        {{ $mahasiswa->jenis_kelamin == 'P' ? 'checked' : '' }}>
      <label class="form-check-label">
        Perempuan
      </label>
    </div>
  </div>
</div>
```

5. Pada tahap ini proses pengambilan data ketika perubahan halaman dibuka dari daftar mahasiswa sudah selesai dilakukan.



## Proses 2: Menyelesaikan Fungsi Update pada Halaman Perubahan

### Data Membuat Route POST Update

1. Buka file konfigurasi route (“<project\_path>/routes/web.php”).
2. Tambahkan baris kode baru untuk menambahkan route khusus menerima POST untuk mengupdate.

```
Route::post('/mahasiswa/tambah', 'MahasiswaController@postTambah')->name('mahasiswa_tambah_post');  
Route::post('/mahasiswa/hapus', 'MahasiswaController@postHapus')->name('mahasiswa_hapus_post');  
Route::post('/mahasiswa/ubah', 'MahasiswaController@postUbah')->name('mahasiswa_ubah_post');
```

### Membuat Fungsi Controller untuk Update ke Database

1. Buka file MahasiswaController pada “<project\_path>/app/Http/Controllers/MahasiswaController.php”.
2. Tambahkan fungsi baru yang bernama **postUbah()**.
3. Lalu tambahkan baris kode seperti dibawah ini untuk melakukan penyimpanan update ke database didalam fungsi **postUbah()**.

```
function postUbah(Request $request) {
    DB::table('mahasiswa')->where('id', $request->input('id'))
    ->update([
        'nama' => $request->input('nama'),
        'jenis_kelamin' => $request->input('jenis_kelamin'),
        'jurusan' => $request->input('jurusan')
    ]);
    return redirect()->route('mahasiswa')

    ->with([
        'alert_class' => 'success',
        'alert_message' => 'Proses Perubahan Berhasil!!'
    ]);
}
```

- Ubah view **mahasiswa\_ubah.blade.php** agar mengarahkan post action formnya ke route MahasiswaController@postUbah(), dan tambahkan blade template @csrf untuk memberikan keamanan ketika melakukan post.

```
<div class="box-body">
    <form action="{{ route('mahasiswa_ubah_post') }}" method="POST">
        @csrf
        <input name="id" type="hidden" value="{{ $mahasiswa->id }}" />
        <div class="form-group row">
```

- Pada tahap ini proses update seharusnya sudah dapat dilakukan.

Universitas Cyber   Home   Mahasiswa

## Daftar Mahasiswa

Proses Perubahan Berhasil!!

Tambah Mahasiswa Baru

Nama	Jurusan	Jenis Kelamin	Aksi
Hifzon	Sastra Inggris	Laki-laki	Ubah Hapus
Annisa	Manajemen Informatika	Perempuan	Ubah Hapus

## Penutup

Sekian tutorial pembelajaran Beginning Laravel ini, semoga dapat menjadi manfaat untuk kawan-kawan semua, atas segala kekurangan yang ada pada tutorial ini penulis mohon maaf, jika ada yang sulit dipahami ataupun saran dan kritik dari tutorial ini, silahkan ditanyakan atau disampaikan kepada penulis. Terima kasih.

Wassalamu'alaikum Wr. Wb.

Dieka Prastya

