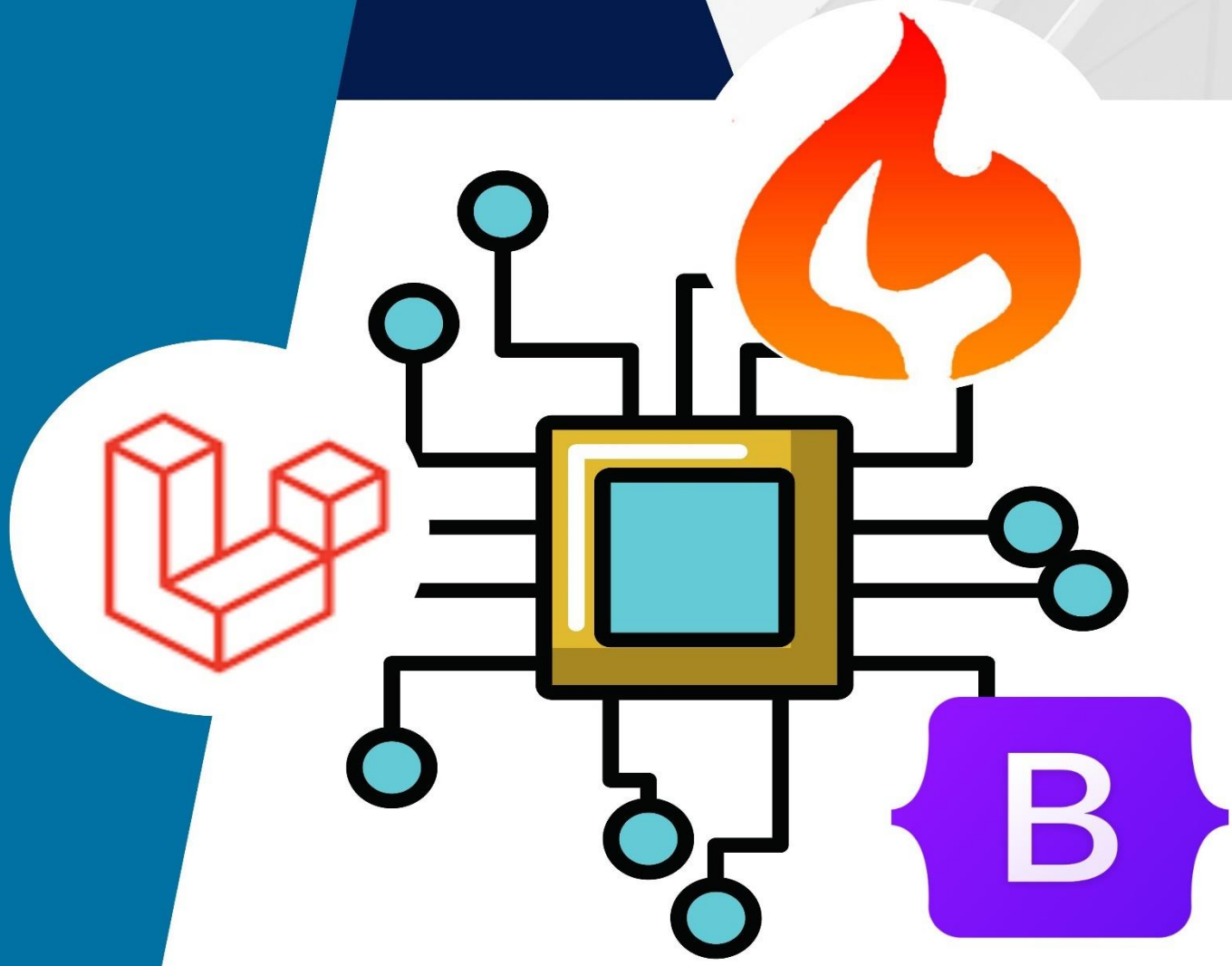
MODUL MATA KULIAH
FRAMEWORK PROGRAMMING

ALLIN JUNIKHAH, M.T.



TEKNIK INFORMATIKA
UIN MAULANA MALIK IBRAHIM
MALANG

KATA PENGANTAR

Puji syukur penulis panjatkan ke hadirat Allah SWT atas segala rahmat dan karuna yang dilimpahkan sehingga penulis dapat menyelesaikan Modul Mata Kuliah *Framework Programming* ini. Sholawat serta salam tercurahkan kepada Rasulullah Muhammad SAW beserta keluarganya.

Selama melaksanakan penyusunan modul ini, penulis mendapat bantuan dan dukungan dari berbagai pihak. Untuk itu penulis mengucapkan terima kasih kepada pihak-pihak berikut ini.

1. Bapak Dr. Fachrul Kurniawan ST., M.MT., IPM selaku Kepala Program Studi Teknik Informatika UIN Maulana Malik Ibrahim Malang.
2. Bapak Dr. Muhammad Faisal, S.Kom., M.T. yang telah memberikan dukungan saran dan bimbingan dalam bidang multimedia pengembangan e-module *Framework Programming*.
3. Rekan-rekan dosen Teknik Informatika UIN Maulana Malik Ibrahim Malang, yang telah memberikan dukungan sehingga penulis dapat menyelesaikan penyusunan modul *Framework Programming* ini dengan baik.
4. Seluruh mahasiswa mata kuliah *Framework Programming* UIN Maulana Malik Ibrahim Malang yang meberikan dorongan kuat penulis untuk dapat menyusun dan menyelesaikan modul ini.
5. Orang tua dan keluarga yang senantiasa memberikan semangat dan do'anya.
6. Seluruh staf dan laboran Teknik Informatika UIN Maulana Malik Ibrahim Malang
7. Semua pihak yang membantu, yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa modul ini bukanlah tanpa kelemahan, baik dari segi ilmu yang disampaikan maupun Teknik penulisannya. Oleh karena itu, kritik dan saran akan sangat membantu dalam membenahi kekurangan-kekurangan tersebut. Harapan penulis modil ini dapat memberikan ilmu dan manfaat bagi pembacanya terkhusus mahasiswa Teknik Informatika kelas *Framework Programming* UIN Maulana Malik Ibrahim Malang.

Penulis

Allin Junikhah, M.T.

DAFTAR ISI

BAB I

PENDAHULUAN

1. Deskripsi Singkat

Framework programming merupakan salah satu mata kuliah pilihan pada program studi Teknik Informatika. Pada mata kuliah pilihan ini, mahasiswa diharapkan dapat memiliki kompetensi yang mendalam dalam kelompok keahlian *software engineering*. Mata kuliah *Framework Programming* terdiri dari mata kuliah teori 3 sks yang tidak diikuti dengan mata kuliah praktikum. Materi dalam mata kuliah yang didominasi dengan materi *coding* sangat membutuhkan banyak pemahaman dan praktik pemrograman. Modul *Framework Programming* ini disusun penulis dengan harapan menjadi salah satu sumber belajar yang dapat digunakan dalam mata kuliah *Framework Programming* bersamaan dengan materi yang diberikan di kelas.

Modul *Frameworks Programming* disusun untuk meningkatkan pemahaman *programming* yang baik sehingga akan menciptakan ketercapaian mata kuliah yang diikuti yang sekaligus juga mendukung ketercapaian profil lulusan *Software Engineer* yang menjadi salah satu profil lulusan Teknik Informatika UIN Malang.

Modul ajar merupakan bahan penting dalam pembelajaran peserta didik. Sebagai bahan utama pemahaman konsep materi perkuliahan, modul ajar sangat berperan dalam kesuksesan mahasiswa memahami materi baik konseptual maupun praktikum. Proses belajar yang tidak didukung dengan penyediaan sarana dan prasarana yang memadai sangat menentukan keberhasilan proses pendidikan.

Ketercapaian mata kuliah akan sangat mempengaruhi ketercapaian profil lulusan. *Framework Programming* diharapkan dapat meningkatkan kompetensi *Backend development, Frontend development,* dan *Fullstack development* mahasiswa yang kelak akan mengisi profil lulusan *Software Engineer*, yang merupakan 1 dari 6 profil lulusan Teknik Informatika, disamping *Information System Analyst, Game & Multimedia Engineer, Data Scientist & Artificial Intelligent Engineer, Network & Cloud Engineer, Assistant Researcher & Academician*. Maka dari itu sangat dibutuhkan media pembelajaran berupa modul dan e-module untuk mengoptimalkan ketercapaian matakuliah.

Selain akan sangat mempengaruhi kompetensi mahasiswa dalam *programming*, pengembangan modul dan e-module secara tidak langsung juga mendukung proses evaluasi materi ajar dan

kurikulum untuk menindaklanjuti masalah kemutakhiran materi Proses Belajar Mengajar (PBM) oleh Lembaga Penjaminan Mutu (LPM).

2. Tujuan Pembelajaran

Setelah mengikuti pembelajaran ini, mahasiswa diharapkan mampu mencapai capaian mata kuliah *Framework Programming* yang tertuang pada Rancangan Pembelajaran Semester, yaitu:

1. Mahasiswa mampu menjelaskan konsep MVC dalam framework.
2. Mahasiswa mampu membangun dan mengembangkan aplikasi web aktif menggunakan framework.

Dengan rincian sub capaian mata kuliah sebagai berikut:

1. Mampu memahami, menjelaskan, dan mengimplementasikan konsep website
2. Mampu memahami, menjelaskan, dan mengimplementasikan konsep PHP
3. Mampu memahami dan menjelaskan konsep framework
4. Mampu memahami dan menjelaskan konsep framework Codeigniter
5. Mampu memahami, menjelaskan, dan mengimplementasikan konsep struktur direktori framework Codeigniter
6. Mampu memahami dan mengimplementasikan konsep CRUD menggunakan framework Codeigniter
7. Mampu mengetahui dan memahami implementasi framework selain Codeigniter
8. Mampu membangun aplikasi berbasis website menggunakan framework

3. Materi Pokok

Pokok bahasan pada materi modul ini meliputi: Pengenalan *Framework Programming*, tujuan penggunaan *Framework Programming*, contoh dan kebutuhan software dalam pembangunan aplikasi web berbasis *Framework*, Memperdalam konsep PHP dan OOP dalam *Framework*, Installasi *Framework* serta membuat aplikasi sederhana yang mengimplementasikan CRUD (Create, Read, Update, Delete).

4. Petunjuk Belajar

Mahasiswa dapat memanfaatkan modul ini dalam mengikuti perkuliahan Framework Programming, bersamaan dengan materi yang diajarkan di kelas. Mahasiswa dapat mengikuti langkah-langkah dalam praktik mandiri sesuai yang telah disediakan modul. Mahasiswa juga dapat memanfaatkan *e-module* yang dikembangkan bersamaan dengan modul ini untuk memperkuat pemahaman materi perkuliahan *Framework Programming* dengan pengalaman pembelajaran yang lebih menarik.

BAB II
FRAMEWORK PROGRAMMING

Tujuan

1. Mahasiswa mampu memahami, menjelaskan, dan mengimplementasikan konsep website
2. Mahasiswa memahami, menjelaskan, dan mengimplementasikan konsep PHP
3. Mahasiswa memahami dan menjelaskan konsep framework

1. Definisi *Framework Programming*

Framework atau dalam bahasa indonesia dapat diartikan sebagai "kerangka kerja". ***Framework Programming*** merupakan kumpulan program yang terdiri dari fungsi-fungsi/prosedur-prosedur dan class-class untuk tujuan tertentu yang sudah siap digunakan sehingga bisa lebih mempermudah dan mempercepat pekerjaan seorang programer, tanpa harus membuat fungsi atau class dari awal.

2. Tujuan Penggunaan *Framework*

1. Mempercepat dan mempermudah pembangunan sebuah aplikasi web.
2. Relatif memudahkan dalam proses maintenance karena sudah ada pola tertentu dalam sebuah *framework* (dengan syarat programmermengikuti pola standar yang ada)
3. Umumnya *framework* menyediakan fasilitas-fasilitas yang umum dipakai sehingga kita tidak perlu membangun dari awal (misalnya validasi, ORM, pagination, multiple database, scaffolding, pengaturan session, error handling, dll
4. Lebih bebas dalam pengembangan jika dibandingkan CMS
5. Meskipun pada dasarnya sama-sama memudahkan dalam pembuatan website. Apabila menggunakan CMS, kita tidak perlu repot-repot dan pusing-pusing menulis script maupun sekumpulan kode. Dengan fitur CMS, semuanya telah dibuat instan dan kita hanya perlu sedikit mengatur bagian konten dan interface-nya saja.

3. Contoh *Framework*

1. CodeIgniter
2. Laravel
3. Yii
4. Symfony
5. Zend *Framework*
6. CakePHP

7. FuelPHP
8. PhalconPHP
9. Slim
10. Lumen dll.

4. Kebutuhan Software dalam Implementasi *Framework Programming*

1. *Framework* Codeigniter / CI (download pada http://www.codeigniter.com/download)
2. Siapkan Editor. Seperti: Visual Studio Code, Sublime Text, Notepad ++, Codelobster, Netbean, Dreamweaver dll.
3. Database dan PHP (Xampp/ Wamp)
4. Internet Browser, digunakan untuk menampilkan aplikasi dan berinteraksi dengan antarmukanya. Sebagai contoh : Internet Explorer, Opera,Google Chrome, Mozilla Firefox dll. Sebagai contoh: Windows 10, Windows 7, Windows XP, Ubuntu, Fedora, Slackware, BlankOn dll

5. Mengenal PHP dan OOP

- PHP (Hypertext Preprocessor) adalah bahasa pemprograman yang digunakan untuk membuat website, atau bisa disebut bahasa pemprograman yang ada disisi server.
- Object Oriented Programming (OOP) merupakan paradikma pemprograman yang berorientasi pada obyek.
- Dalam konsep OOP, semua masalah dibagi dalam obyek obyek karena konsep data dan fungsi – fungsi yang akan mengoperasikan digabungkan menjadi satu kesatuan yang dapat disebut sebagai obyek.
- Secara sederhana obyek sendiri adalah kumpulan variable dan fungsi yang dibungkus menjadi satu tempat atau kesatuan.
- Dalam sebuah objek diciptakan melalui sebuah kelas atau dikenal dengan istilah instan of class. Didalam obyek sendiri mempunyai 2 elemen utama :
    1. **Attributes** : merupakan nilai – nilai yang tersimpan dalam obyek tersebut dan secara langsung maupun tidak langsung untuk menentukan karakteristik dari obyek tersebut.
    2. **Method :** ini merupakan suatu aksi yang akan dijalankan atau dikerjakan obyek tersebut.

3. **Enkapsulasi**

   Didalam OOP, semua elemen program merupakan objek-objek yang saling berinteraksi. satu sama lain. Dalam interaksinya, tiap objek mempunyai informasi yang bisa diakses oleh objek lain ataupun yang disembunyikan dari objek lain. Sebagai contoh, dalam dunia nyata, objek mobil memiliki steer, pedal, yang langsung berinteraksi dengan sopir.Tetapi sopir tidak secara langsung berinteraksi dengan mesin.

4. **Class**

   adalah struktur dari suatu objek yang didalamnya terdapat Attribute dan juga Method, atau bisa disebut sebuah struktur yang mendefinisikan sebuah variabel dan juga method umum pada semua object. Class juga merupakan grup suatu object dengan kemiripan attributes/properties, behaviour dan relasi ke object lain.

   Contoh: class artikel, memiliki attribute: judul, nama_penulis, tanggal_terbit, kategori, dll. Artikel juga memiliki method terbitkan(), arsipkan(), hapus(), edit(), dll.

```
class artikel {

    // attribute
     public $tanggal_terbit;

    // method
    function terbitkan() {
       echo 'artikel berhasil diterbitkan ';
    }
}
```

5. **Object** adalah instance dari class, ini berarti class harus di-instantiate (dibuat objeknya) terlebih dahulu agar bisa digunakan dalam program (kecuali static class).

   contoh pembuatan objek dalam php:

```
// deklarasi class
class artikel {
    public $tanggal_terbit;
    function terbitkan() {
      echo 'artikel berhasil diterbitkan ';
  }
}
// instantiate objek baru
$artikel_a = new artikel();
// instantiate objek lainnya
$artikel_b = new artikel();
```

Contoh untuk mengakses suatu atribut/method dari objek, maka kita bisa menggunakan tanda -> (strip, lebih besar dari) :

```
// mengakses attribute
$artikel_a->tanggal_terbit
// mengakses method
$artikel_a->terbitkan()
```

6. I**nheritance**

   Sebuah class dapat di-inherit (diwariskan) ke class turunannya, sehingga class turunannya memiliki attribute dan method yang sama dengan class induknya. Sebagai contoh, class artikel bisa diwariskan menjadi class berita. Class artikel disebut parent class, dan class berita disebut child class.

```
// parent class
class artikel {
    public $tanggal_terbit;
    function terbitkan() {
      echo 'artikel berhasil diterbitkan ';
    }
}
// child class
class berita extends artikel {
    // attribute tambahan
      public $nama_wartawan;
      public $nara_sumber;
    // method tambahan
     function cek_nara_sumber() {
        echo 'nara sumber berhasil dicek ';
    }
}
```

6. Persiapan Perangkat Aplikasi Framework

a. Installasi Database dan PHP – MySQL (XAMPP)

Codeigniter merupakan *framework* PHP, karena itu ia pasti membutuhkan web server. Berikut ini requirement server untuk Codeigniter 4:

- PHP Versi 7.2+
- MySQL Versi 5.1+
- Phpmyadmin

Jika kamu sudah menginstal XAMPP, maka ketiga aplikasi server ini sudah terpenuhi. Tapi jika kamu pengguna Linux, maka ini bisa diinstal satu-per-satu.

Setelah menginstal webserver, kita harus mengaktifkan beberapa ekstension yang dibutuhkan untuk pengembangan Codeigniter 4.

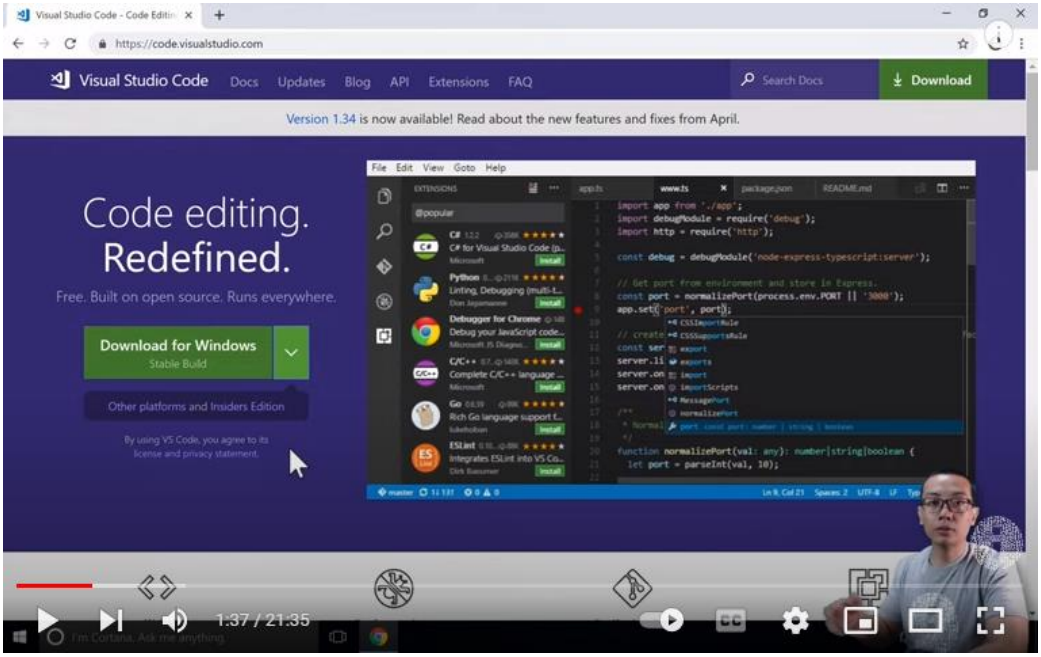
Untuk pengguna Windows dan XAMPP. Silahkan buka XAMPP Control Panel, lalu pada bagian apache klik Config->PHP.

Setelah itu, cari di bagian extension dan hapus **;** yang ada di depan nama extension untuk mengaktifkannya.

php-json	**ekstension untuk bekerja dengan JSON;**
php-mysqlnd	**native driver untuk MySQL;**
php-xml	**ekstension untuk bekerja dengan XML;**
php-intl	**ekstensi untuk membuat aplikasi multibahasa;**

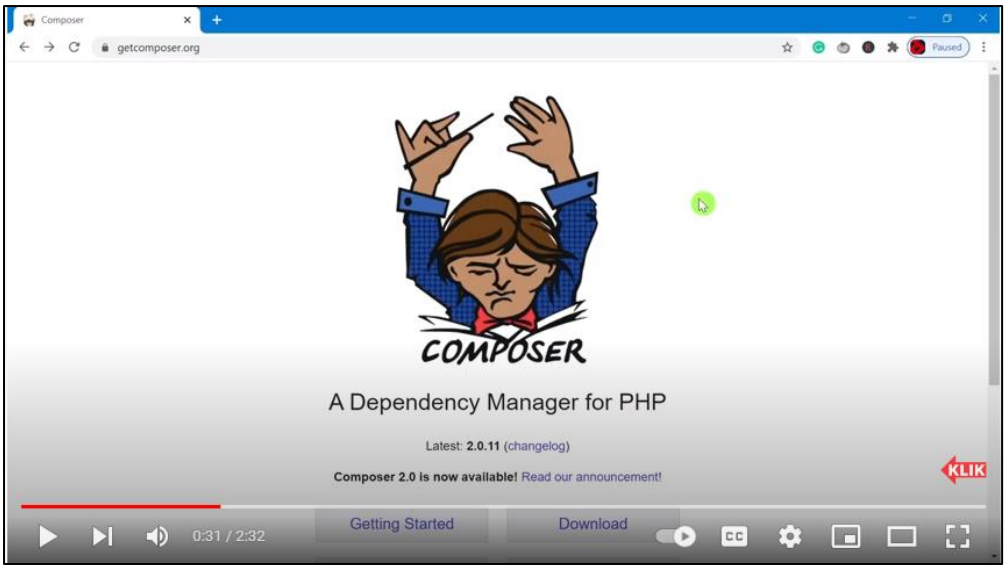
b. Installasi Editor – Visual Studio Code

Link youtube : https://www.youtube.com/watch?v=NnLVIY3a2es&t=114s



c. Installasi Composer

Link youtube : https://www.youtube.com/watch?v=NOwLcV05Lz8



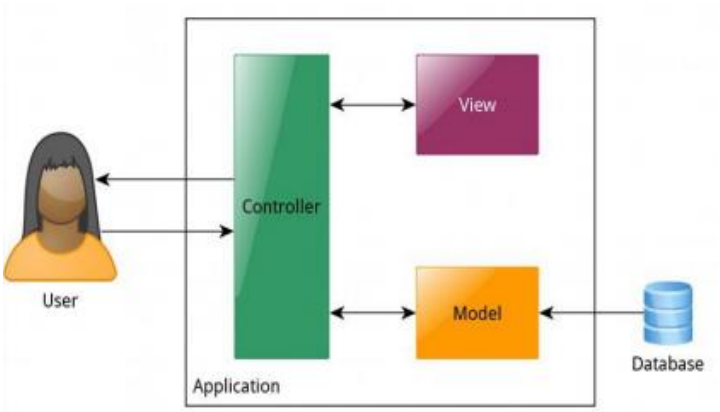
BAB III
FRAMEWORK CODEIGNITER

Tujuan

1. Mahasiswa mampu memahami dan menjelaskan konsep *framework* Codeigniter
2. Mahasiswa mampu memahami, menjelaskan, dan mengimplementasikan konsep struktur direktori *framework* Codeigniter
3. Mampu memahami dan mengimplementasikan konsep CRUD menggunakan *framework Codeigniter*
4. Mampu membangun aplikasi berbasis website menggunakan *framework* codeigniter

1. *Framework* Codeigniter

- CodeIgniter adalah sebuah web application network yang bersifat open source yang digunakan untuk membangun aplikasi php dinamis.
- CodeIgniter menjadi sebuah *framework* PHP dengan model MVC (*Model, View, Controller*) untuk membangun website dinamis dengan menggunakan PHP yang dapat mempercepat pengembang untuk membuat sebuah aplikasi web.
- Selain ringan dan cepat, CodeIgniter juga memiliki dokumentasi yang super lengkap disertai dengan contoh implementasi kodenya. Dokumentasi yang lengkap inilah yang menjadi salah satu alasan kuat mengapa banyak orang memilih CodeIgniter sebagai *framework* pilihannya. Karena kelebihan-kelebihan yang dimiliki oleh CodeIgniter, pembuat PHP Rasmus Lerdorf memuji CodeIgniter di frOSCon (Agustus 2008) dengan mengatakan bahwa dia menyukai CodeIgniter karena "*it is faster, lighter and the least like a framework.*"



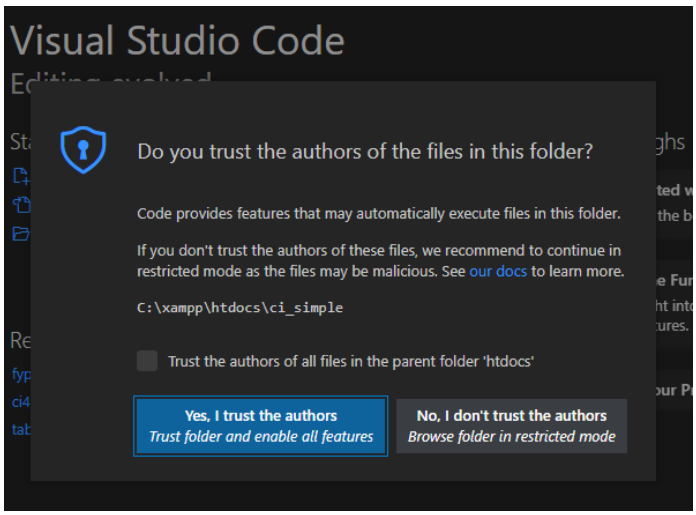
Cara Kerja MVC

2. Installasi Codeigniter 4.1.9 Melalui Aplikasi Composer

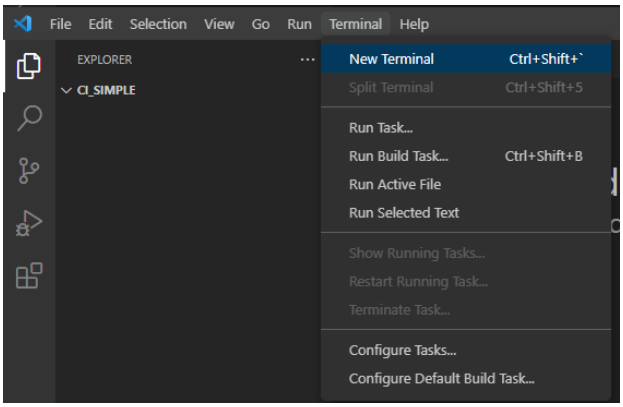
Pastikan terlebih dahulu beberapa software pendukung telah terinstall

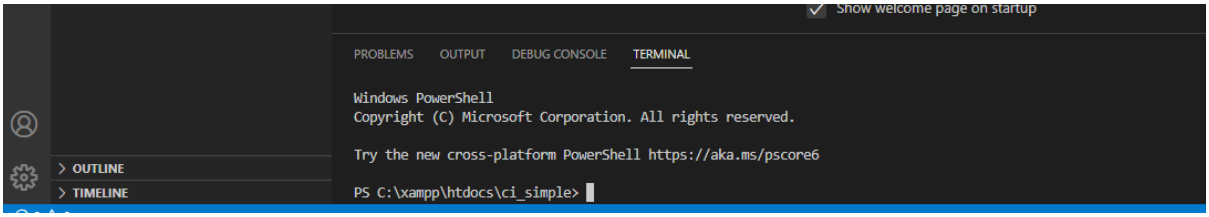
1. Xampp / Wamp (ex: Xampp)
2. Text Editor (ex: Visual Studio Code)

1. Langkah pertama buat folder untuk meletakkan file project codeigniter. Dengan cara open folder, buat folder baru (ex: **ci_simple**), letakkan pada direktori folder tersebut.
2. Muncul dialog box, pilih yes I trust the author

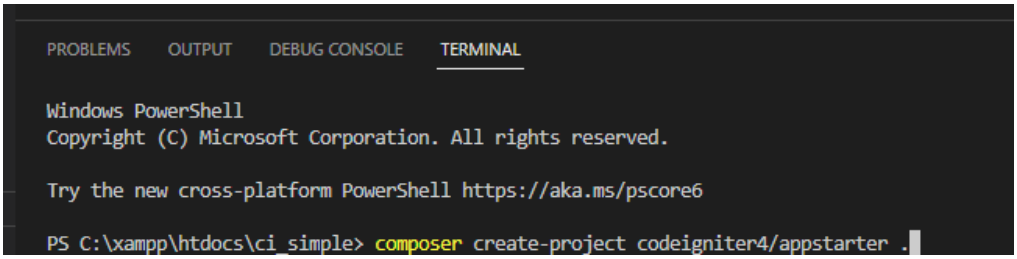


3. Pada direktori ci_simple kita akan meletakkan seluruh file project codeigniter kita.
4. Buka window terminal, sebelumnya pastikan sudah terinstall Composer. Masuk terminal dengan cara Open menu terminal – New Terminal

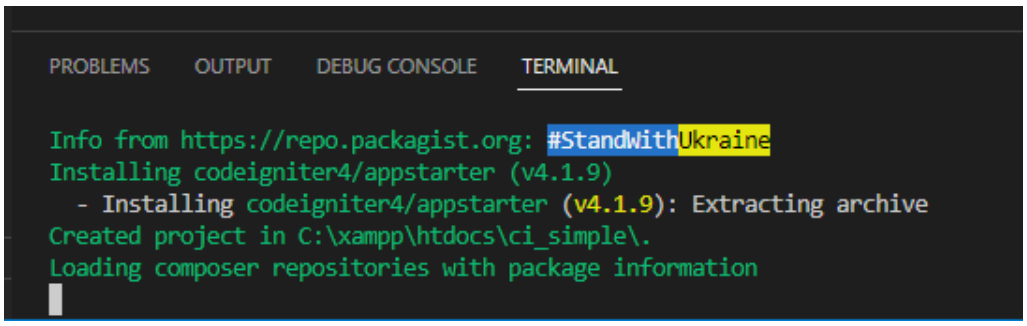




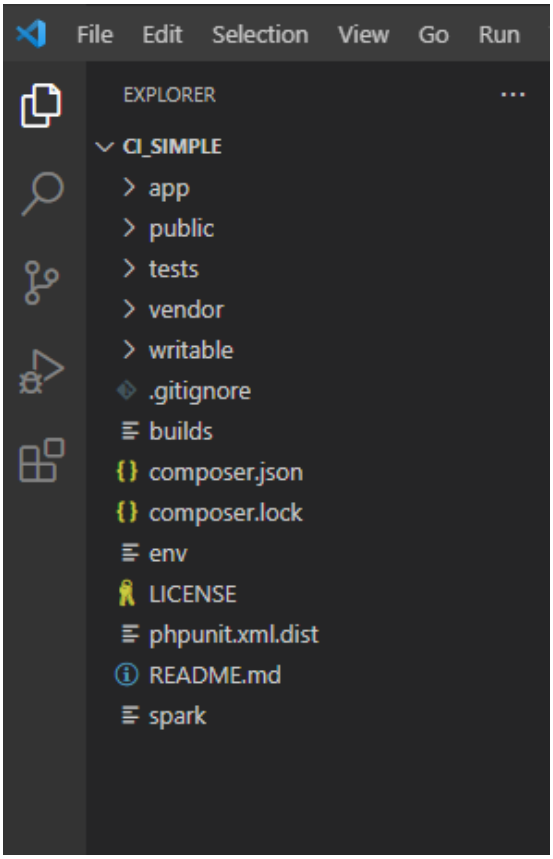
5. Lakukan proses installasi Codeigniter dengan menuliskan script berikut **Composer create-project codeigniter4/appstarter.** Yang ditulis pada window terminal pada direktori yang kita pilih.



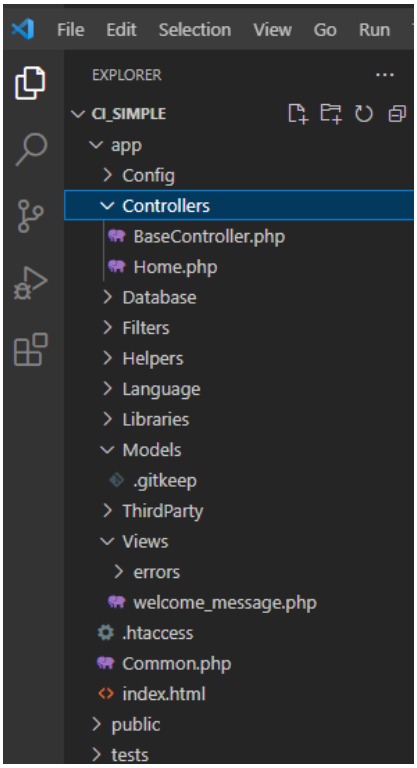
6. Tunggu hingga direktori berhasil terisi file codeigniter.



7. Berikut jika file codeigniter sudah berhasil dibuat.

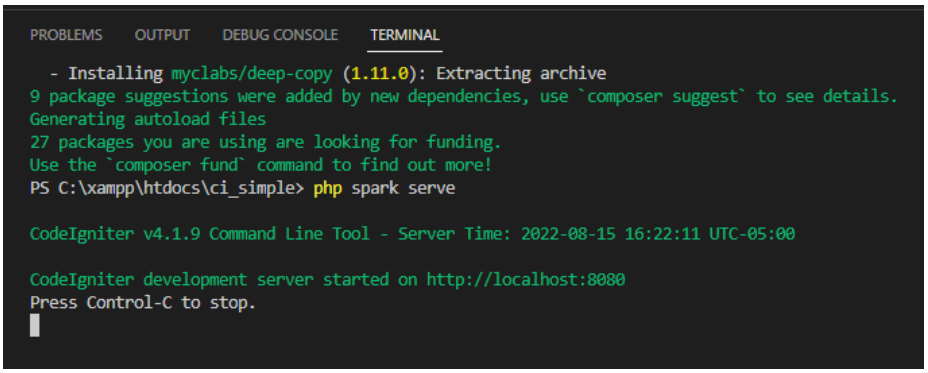


8. Berikut struktur direktori pada Codeigniter. Pada project yang berbasis MVC (*model, View, Controller*) ini kita akan focus pada 3 direktori utama yaitu**, Controllers, Models, dan Views.**

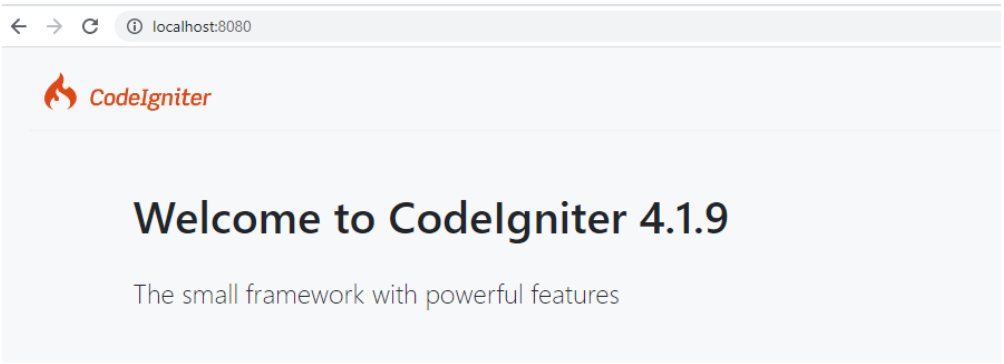


3. Menjalankan Codeigniter

Untuk menjalankan codeigniter yang telah kita install, tulis script berikut pada terminal **php spark serve**

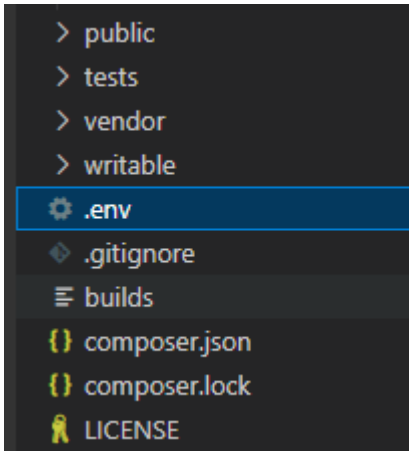


Kemudian buka browser ketikkan alamat base url http://localhost:8080/, aplikasi web codeigniter telah dapat dijalankan



4. Setting Environment Codeigniter

Secara default project codeigniter 4 yang telah di download belum mengaktifkan environtemnt file. Untuk mengaktifkan environment file, silahkan buka file **env** pada project. Kemudian rename menjadi **.env** (dot env). Selanjutnya terlihat gambar di atas icon env berubah menjadi icon setting setelah di ubah menjadi **.env**.



Hal penting yang perlu dilakukan setelah instalasi codeigniter 4 adalah mensetting jenis CI_ENVIRONMENT.

Terdapat tiga mode CI_ENVIRONMENT codeigniter 4.

1. **Development**

   Penggunaan mode environment ini ditujukan untuk pengembang. Menggunakan environment development ini memungkinkan pengembang melihat error output. Sebagai programmer tentu dalam membuat aplikasi kita membutuhkan error output untuk melihat dimana kesalahan penulisan script yang dibuat.

2. **Production**

   Mode production menampilkan aplikasi tanpa error output. Jika app telah selesai dan untuk digunakan oleh user sebaiknya menggunakan mode production.

3. **Testing**

   Mode ini digunakan untuk test.

Setelah memahami cara mengaktifkan dan hal apa saja yang dapat di setting di environment codeigniter 4 selanjutnya kita lakukan perubahan terhadap file .env untuk mengetahui fungsi dari perubahan yang dilakukan.

Mengubah jenis CI_ENVIRONMENT dari production ke development

Buka file .env cari tulisan berikut:

 # CI_ENVIRONMENT = production

Hilangkan tanda # dan ubah menjadi development

CI_ENVIRONMENT = development

```
#--------------------------------------------------------------------
# ENVIRONMENT
#--------------------------------------------------------------------

# CI_ENVIRONMENT = production
```

```
#--------------------------------------------------------------------
# ENVIRONMENT
#--------------------------------------------------------------------

CI_ENVIRONMENT = development
```

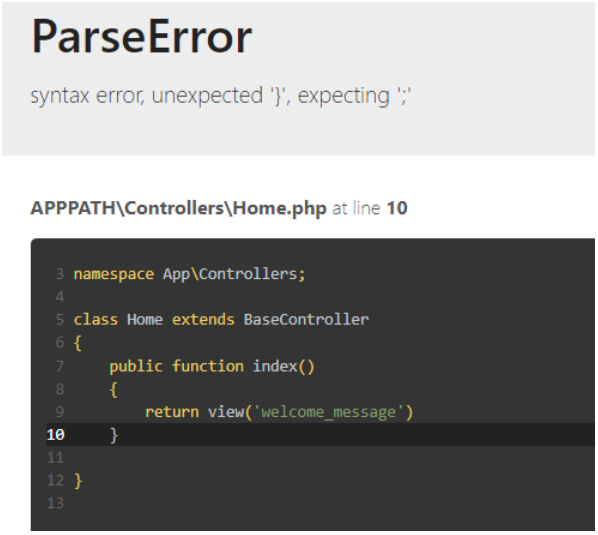
Kegunaan Mengubah Environment

Setelah mengubah mode environment dari production ke development kita dapat menampilkan error output codeigniter 4.

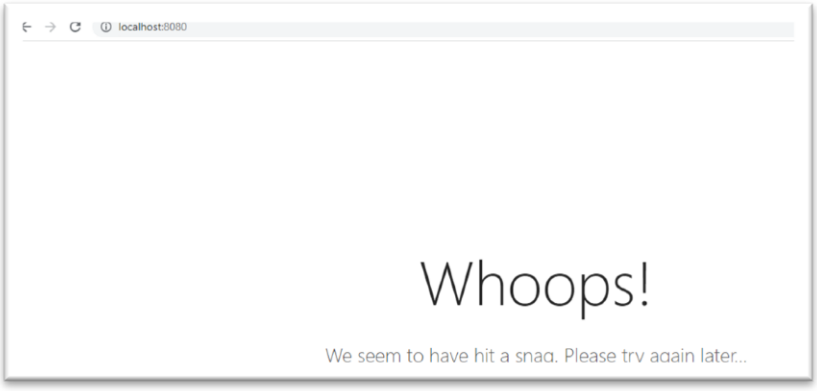
Untuk menguji error output hilangkan tanda ; pada script berikut.

```php
app > Controllers > Home.php
<?php

namespace App\Controllers;

class Home extends BaseController
{
    public function index()
    {
        return view('welcome_message');
    }

}

```

Berikut tampilan jika terdapat error pada script.



Jika environment terdapat pada mode production, bukan development maka tidak dapat menampilkan error code seperti pada tampilan di bawah ini.



APP

Di environment bagian app kita dapat setting app base url aplikasi yang dikembangkan. Sebelumnya untuk membuat setting base_url di set pada **application/config.** Saat ini juga dapat melakukan set

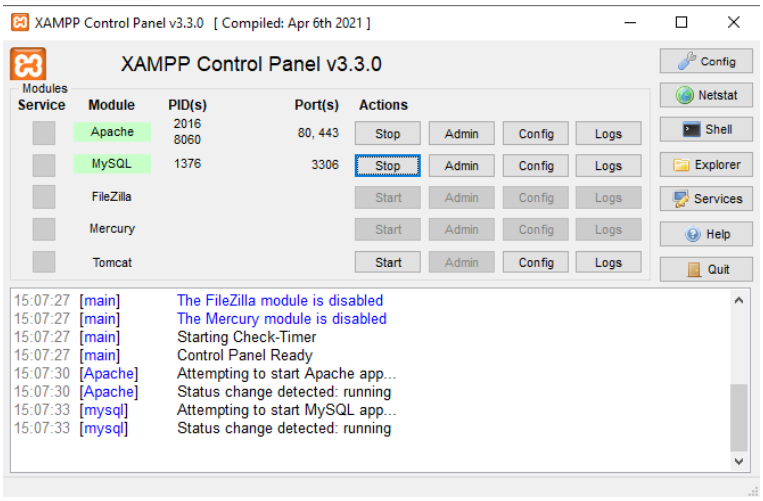
base_url di app/config namun jika base_url juga di set pada environment maka yang dipakai sebagai base_url adalah base url yang terdapat pada environment.

DATABASE

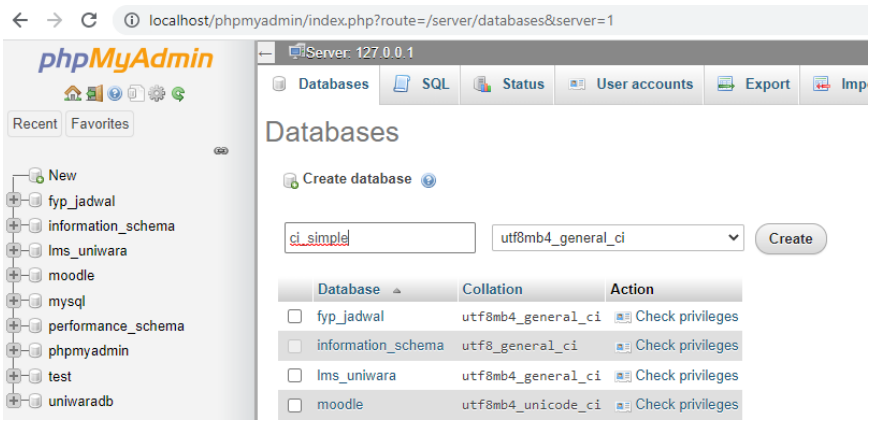
Menggunakan environment juga dapat melakukan set terhadap config database. Meskipun setting untuk database juga terdapat pada app/config. Set database di environment akan digunakan jika telah di set. Config database di environment dapat diisi dalam bentuk default dan test.

5. Mempersiapkan Database Project Baru

1. Sebelum memulai membuat database, pastikan database mySQL yang berada pada XAMPP terinstall dan aktif.



2. Masuk pada browser, ketikkan http://localhost/phpmyadmin/ untuk menuju pada database mySQL
3. Buat database yang dibutuhkan ex: **ci_simple** dan beberapa table yang diperlukan



4. Selanjutnya buat table. Ex: **tbl_mahasiswa** dengan beberapa atribut yang dibutuhkan seperti contoh berikut:

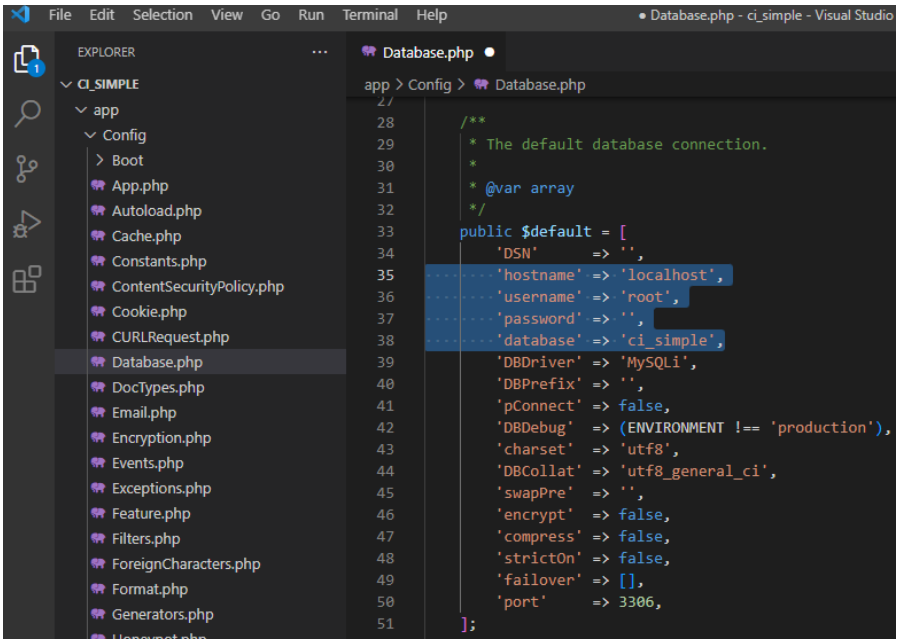
#	Name	Type
1	id_mhs	int(35)
2	nim	int(35)
3	nama	varchar(255)
4	jurusan	varchar(255)
5	alamat	varchar(255)

5. Koneksikan database pada project codeigniter kita, dengan cara masuk pada file **App – Config – Database.php** pada struktur direktori
6. Ubah sesuai dengan konfigurasi database kita.

   Ex: **username= root**

   password= ‘ ‘

   Database= ci_simple



Pada pertama kali tampil web codeigniter akan menampilkan welcome message. Tampilan merupakan hasil dari tampilan file **Views** **welcome_message.php** (terdapat pada folder **Views)** yang dipanggil oleh

Home.php

app > Controllers > Home.php

```php
<?php

namespace App\Controllers;

class Home extends BaseController
{
    public function index()
    {
        return view('welcome_message');
    }

}

```

Welcome_message.php

app > Views > welcome_message.php

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Welcome to CodeIgniter 4!</title>
    <meta name="description" content="The small framework with powerful features">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="shortcut icon" type="image/png" href="/favicon.ico"/>

    <!-- STYLES -->

    <style {csp-style-nonce}>
        * {
            transition: background-color 300ms ease, color 300ms ease;
        }
        *:focus {
            background-color: rgba(221, 72, 20, .2);
            outline: none;
        }
        html, body {
```

6. Membuat Project Baru Aplikasi Sederhana

Selanjutnya coba membuat aplikasi sederhana dengan beberapa form seperti form input, update delete mahasiswa dengan database yang telah di buat.

Berikut script untuk form **Input_mhs.php** pada **VIEWS.** Form ini nantinya akan diproses oleh **Controllers Home.php** dengan method **simpanmhs.**

Input_mhs.php

app > Views > Input_mhs.php

```
<center><h2>Form Data Mahasiswa</h2>
<form method="POST" action="<?php echo site_url('home/simpanmhs') ?> " >
 <input type="text" name="nim_mhs" placeholder="nim">
 <br>
 <input type="text" name="nama_mhs" placeholder="nama">
 <br>
 <select name="jurusan_mhs">
  <option value="Teknik Informatika">Teknik Informatika</option>
  <option value="Teknik Arsitektur">Teknik Arsitektur</option>
 </select>
 <br>
 <input type="text" name="alamat_mhs" placeholder="alamat">
 <br><br>
 <button type="submit">Simpan Data</button>
  
 <a href="<?php echo site_url('home/viewmhs') ?>">Lihat Data</a>
</form></center>

```

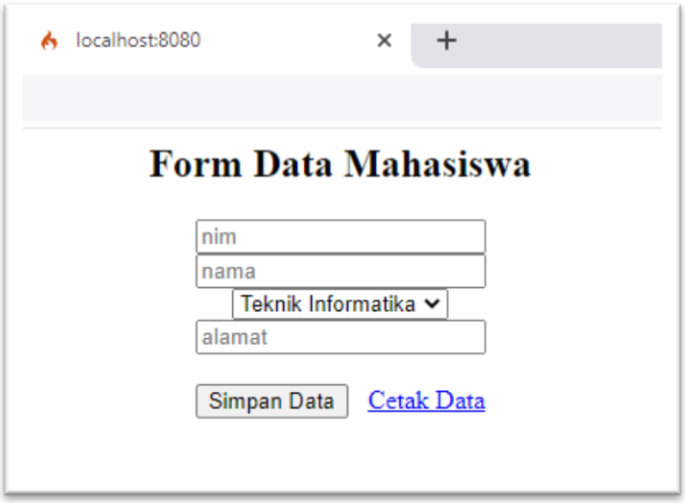
Kemudian **Views Input_mhs.php** (terdapat pada folder **Views)** panggil pada **Controllers Home.php.** Ubah pemanggilan welcome message sebelumnya.

Home.php

app > Controllers > Home.php

```
<?php

namespace App\Controllers;

class Home extends BaseController
{

    public function index()
    {
        return view('Input_mhs');
    }


}

```

Kemudian lihatlah perubahan pada welcome message akan menyesuaikan halaman input data mahasiswa yang telah dibuat pada **Views Input_mhs.php.**



A. CREATE DATA

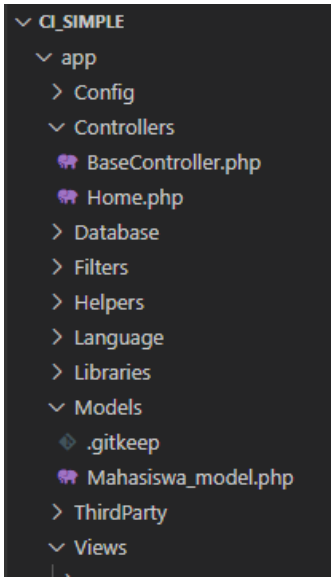
Setelah berhasil menampilkan halaman input yang memanggil dari struktur **Views.** Selanjutnya kita akan bekerja pada struktur **Controllers** **Home.php** dimana akan memproses inputan dari form. Tambahkan Method **simpanmhs()** seperti pada script berikut ini.

Home/simpanmhs

app > Controllers > Home.php

```
10      public function index()
11      {
12          return view('Input_mhs');
13      }
14
15      public function simpanmhs()
16      {
17          $nim = $this->request->getPost('nim_mhs');
18          $nama = $this->request->getPost('nama_mhs');
19          $jurusan = $this->request->getPost('jurusan_mhs');
20          $alamat = $this->request->getPost('alamat_mhs');
21
22          $data = ['nim'=> $nim,
23          'nama'=> $nama,
24          'jurusan'=> $jurusan,
25          'alamat'=> $alamat,];
26
27          $mhs = new Mahasiswa_model();
28          $tabelMhs="tbl_mahasiswa";
29          $simpan=$mhs->saveMhs($tabelMhs,$data);
30          return redirect()->to(site_url());
31
32      }
33  }
```

Method pada **Controllers** **Home/simpanmhs** digunakan untuk menampung data pada form inputan yang selanjutnya akan kita salurkan pada Struktur **Models**, untuk proses memasukkan ke dalam database. Buatlah terlebih dahulu file **Mahasiswa_model.php** pada **Models.**



Tuliskan script berikut pada file **Models** **mahasiswa_model.php.** Buatlah beberapa method seperti pada script di bawah ini. Kita akan menggunakan method **__construct** untuk melakukan koneksi dengan table yang telah dibuat sebelumnya (**tbl_mahasiswa**) dan method **saveMhs** untuk proses memasukkan data mahasiswa ke dalam database.

Mahasiswa_model/saveMhs

app > Models > Mahasiswa_model.php

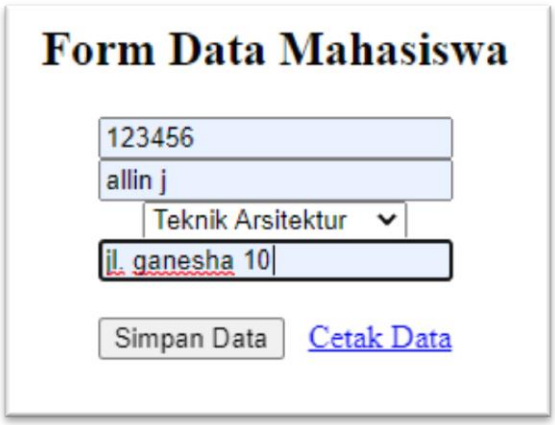
```php
<?php
namespace App\Models;
use CodeIgniter\Model;

class Mahasiswa_model extends Model
{
    protected $table = 'tbl_mahasiswa';

    function __construct()
    {
        $this->db = db_connect();
    }

    function saveMhs($tabel,$data)
    {
        $this->db->table($tabel)->insert($data);
        return true;
    }
}
```

Koneksikan **models Mahasiswa_model** dengan **controllers Home** dengan menambahkan script berikut. Dan letakkan sebelum penulisan Class.

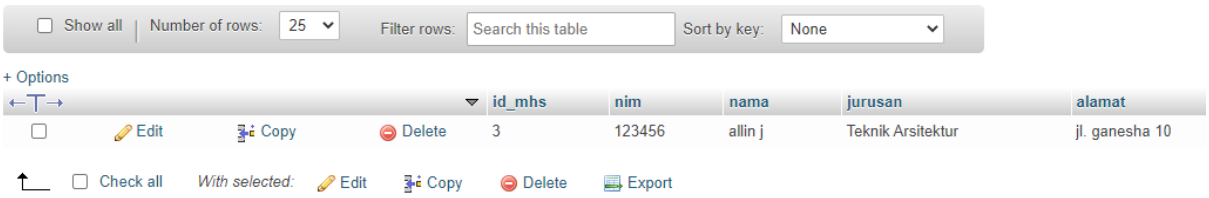
Home.php

```
app > Controllers > Home.php
1   <?php
2
3   namespace App\Controllers;
4   use App\Models\Mahasiswa_model;
5
6   class Home extends BaseController
7   {
8
```

Selanjutnya coba inputkan beberapa inputan dalam form untuk mengetahui keberhasilan koneksi database.



Lihat hasil inputan pada database seperti berikut.



B. READ DATA

Untuk melakukan proses View/Read Data yang telah dinput. Kita dapat menambahkan method **viewmhs** pada **Controllers Home.php** (Home/viewmhs)

Home/viewmhs

```
33      public function viewmhs()
34      {
35          $mhs = new Mahasiswa_model();
36          $datamhs = $mhs->tampildata();
37          $data = array('dataMhs'=> $datamhs,);
38           echo View ('Viewmhs',$data);
39
40      }
```

Controller ini akan menampilkan data dari database yang dilakukan oleh method **tampildata()** pada struktur **Models** **Mahasiswa_model.php** (Mahasiswa_model/tampildata) dan menampilkan hasil pada struktur **Views** yang terdapat pada **Viewmhs.php.**

Karena method dan file tersebut belum dibuat maka buatlah terlebih dahulu seperti pada script berikut.

Mahasiswa_model/tampildata

```
20      function tampildata()
21      {
22
23          $dataquery=$this->db->query("select * from tbl_mahasiswa");
24          return $dataquery->getResult();
25      }
```

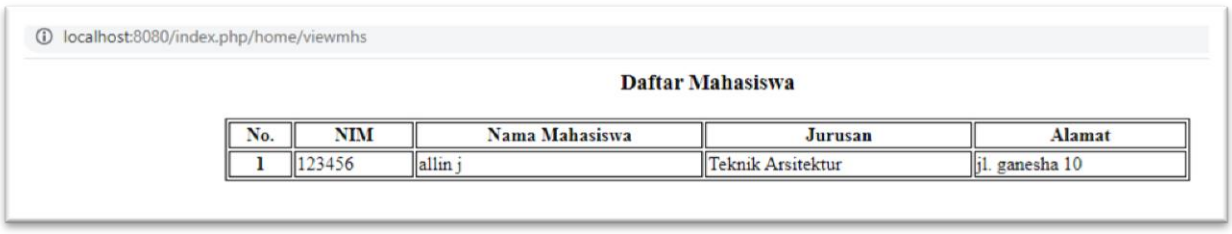
Buat file ViewMhs.php pada struktur Views untuk menampilkan data mahasiswa.

ViewMhs.php

app > Views > ViewMhs.php

```
<style>
table, th, td {
  border: 1px solid black;
}
</style>
<center>
    <h3>Daftar Mahasiswa</h3>
    <table style="width:60%">
    <thead>
        <tr>
            <th>No.</th>
            <th>NIM</th>
            <th>Nama Mahasiswa</th>
            <th>Jurusan</th>
            <th>Alamat</th>
        </tr>
    </thead>
    <tbody>
        <?php $no=0; foreach($dataMhs as $row):$no++;?>
        <tr>
            <th> <?= $no; ?></th>
            <td><?= $row->nim; ?></td>
            <td><?= $row->nama; ?></td>
            <td><?= $row->jurusan; ?></td>
            <td><?= $row->alamat; ?></td>
        </tr>
        <?php endforeach;?>
    </tbody>
</table>
</center>
```

Data selanjutnya dapat di lihat pada link **'Lihat Data'**



C. UPDATE DATA

Tambahkan link untuk proses update/edit data pada tampilan HTML ViewMhs.php seperti pada script berikut ini.

ViewMhs.php

```
        <tr>
            <th>No.</th>
            <th>NIM</th>
            <th>Nama Mahasiswa</th>
            <th>Jurusan</th>
            <th>Alamat</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody>
        <?php $no=0; foreach($dataMhs as $row):$no++;?>
        <tr>
            <th> <?= $no; ?></th>
            <td><?= $row->nim; ?></td>
            <td><?= $row->nama; ?></td>
            <td><?= $row->jurusan; ?></td>
            <td><?= $row->alamat; ?></td>
            <td>
                <a href="/home/formeditmhs/<?=$row->id_mhs;?>">Edit</a>
            </td>
        </tr>
        <?php endforeach;?>
    </tbody>
```

Proses edit data akan dilakukan method **formeditmhs()** yang ada pada **Controllers** **Home.php.** Karena method ini belum tersedia buatlah terlebih dahulu method tersebut seperti pada script di bawah ini.

Home/formeditmhs

```
    public function formeditmhs($id)
    {
        $mhs = new Mahasiswa_model();
        $datamhs = $mhs->getMhs($id);
        $data = array('dataMhs'=> $datamhs,);
        echo View ('Editmhs',$data);

    }
```

Tambahkan method **getMhs()** pada **Model** **Mahasiswa_modul** yang berfungsi untuk melakukan proses pemanggilan data sebelumnya berdasar dari **ID data** yang akan dilakukan pengeditan.

Mahasiswa_model/getMhs

```php
    public function getMhs($id)
    {
        $dataquery=$this->db->query("select * from tbl_mahasiswa where id_mhs=".$id);
        return $dataquery->getResult();
    }
```

Selanjutnya buat tampilan halaman proses pengeditan data **EditMhs.php** pada **Views** dengan script sebagai berikut.

EditMhs.php

app > Views > EditMhs.php

```php
<center><h2>Form Edit Data Mahasiswa</h2>
<?php
    $no=0;
    foreach($dataMhs as $row):
    $no++;
?>

<form action="/home/editmhs/<?=$row->id_mhs;?>" method="POST">
<input type="hidden" class="form-control" name="idMhs" value="<?= $row->id_mhs; ?>">
<label>NIM</label>
<input type="text" class="form-control" name="nimMhs" value="<?= $row->nim; ?>">
<br>
<label>Nama mahasiswa</label>
<input type="text" class="form-control" name="namaMhs" value="<?= $row->nama; ?>">
<br>
<label>Jurusan</label>
<select name="jurusanMhs">
    <option><?= $row->jurusan; ?></option>
    <option value="Teknik Informatika">Teknik Informatika</option>
    <option value="Teknik Arsitektur">Teknik Arsitektur</option>
</select>
<br>
<label>Alamat</label>
<input type="text" class="form-control" name="alamatMhs" value="<?= $row->alamat; ?>">
<br><br>
<button type="submit">Simpan Data</button>
</form>
<?php endforeach;?>
</center>
```

Form halaman edit selanjutnya akan diproses oleh method **editmhs()** pada **Controller Home.** Karena method ini belum tersedia buatlah method tersebut terlebih dahulu.

Home/editmhs

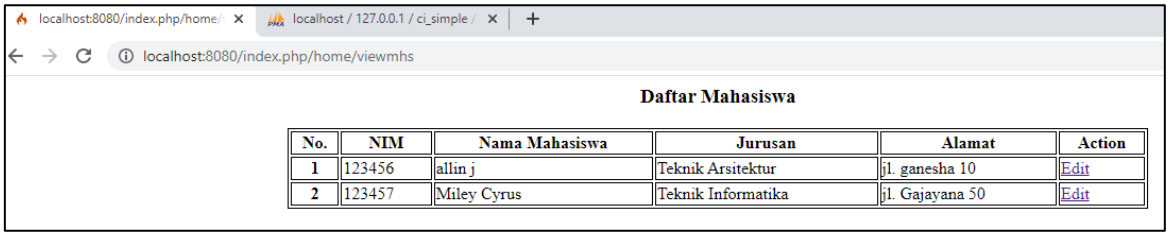
```php
public function editmhs($id)
{
    $nim = $this->request->getPost('nimMhs');
    $nama = $this->request->getPost('namaMhs');
    $jurusan = $this->request->getPost('jurusanMhs');
    $alamat = $this->request->getPost('alamatMhs');

    $data = ['nim'=> $nim,
    'nama'=> $nama,
    'jurusan'=> $jurusan,
    'alamat'=> $alamat,];

    $where=['id_mhs'=>$id];

    $mhs = new Mahasiswa_model();
    $tabelMhs="tbl_mahasiswa";
    $simpan=$mhs->prosesEditMhs($tabelMhs,$data,$where);
    return redirect()->to(site_url());
}
```

Selanjutnya proses akan diteruskan pada method **prosesEditMhs()** pada **Models Mahasiswa_model**, untuk pemrosesan memasukkan data ke dalam database.

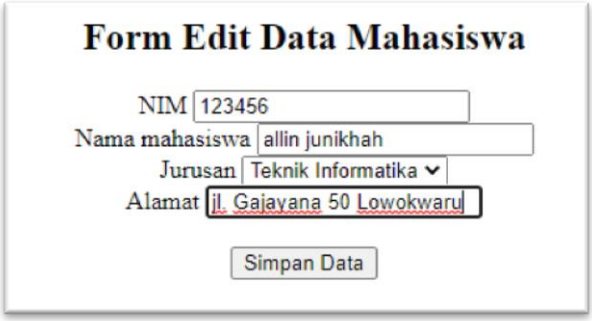
Mahasiswa_model/prosesEditMhs

```php
function prosesEditMhs($table,$data,$where)
{
    $this->db->table($table)->update($data,$where);
    return true;
}
```

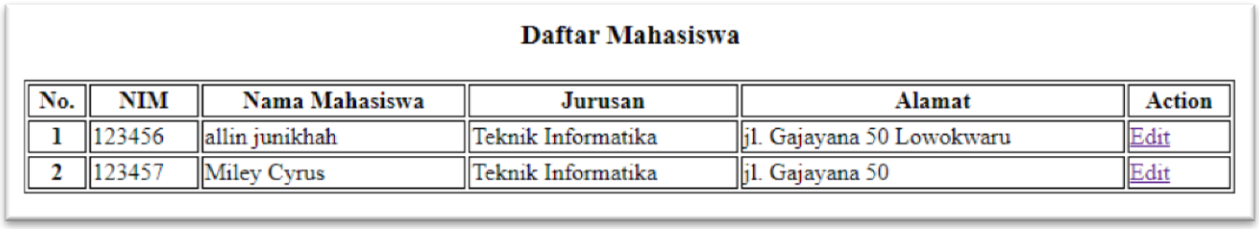
Struktur kerja proses edit telah dibangun. Selanjutnya kita lakukan proses pada form yang telah dibuat sebelumnya. Klik **action Edit** pada data yang akan dilakukan perubahan.



Lakukan proses perubahan data sebelumnya. Selanjutnya simpan Data.



Berikut ini hasil proses pengeditan pada form Edit.



Daftar Mahasiswa

No.	NIM	Nama Mahasiswa	Jurusan	Alamat	Action
1	123456	allin junikhah	Teknik Informatika	jl. Gajayana 50 Lowokwaru	Edit
2	123457	Miley Cyrus	Teknik Informatika	jl. Gajayana 50	Edit

D. DELETE DATA

Proses selanjutnya yaitu bagaimana menghapus data yang telah dibuat. Pertama kali buat terlebih dahulu action **Delete** pada samping masing-masing data. Berikut ini potongan script untuk menambahkan action delete.

ViewMhs.php

```
            <th>Jurusan</th>
            <th>Alamat</th>
            <th>Action</th>
        </tr>
    </thead>
    <tbody>
        <?php $no=0; foreach($dataMhs as $row):$no++;?>
        <tr>
            <th> <?= $no; ?></th>
            <td><?= $row->nim; ?></td>
            <td><?= $row->nama; ?></td>
            <td><?= $row->jurusan; ?></td>
            <td><?= $row->alamat; ?></td>
            <td>
                <a href="/home/formeditmhs/<?=$row->id_mhs;?>">Edit</a>
                <a href="/home/deletemhs/<?=$row->id_mhs;?>">Hapus</a>
            </td>
        </tr>
        <?php endforeach;?>
    </tbody>
</table>
```

Action delete merupakan link pada method **Home/deletemhs** yang berarti proses delete akan dikerjakan oleh method **deletemhs()** yang terdapat pada **Controllers Home.php.** Link ini juga membawa serta **ID data** dari data yang akan dihapus**.** Langkah pertama, karena method **deletemhs()** belum tersedia, tambahkan method terlebih dahulu dengan script sebagai berikut.

Home/deletemhs

```
public function deletemhs($id)
{
    $mhs = new Mahasiswa_model();
    $tabelMhs="tbl_mahasiswa";
    $hapus=$mhs->hapusmhs($id);
    return redirect()->to(site_url());
}
```

Pada script method diatas terlihat terdapat method **hapusmhs()** yang bekerja untuk menghapus data dengan ID yang telah ditentukan. Ini berarti bahwa kita akan membuat method **hapusmhs()** pada **Models Mahasiswa_model.** Berikut script method tersebut.

Mahasiswa_model/hapusMhs

```
function hapusMhs($id)
{
    $dataquery=$this->db->query("delete from tbl_mahasiswa where id_mhs=".$id);
    return true;
}
```

Selanjutnya kita akan mencoba menjalankan sistem penghapusan data yang telah dibuat sebelumnya. Tampilkan data terlebih dahulu secara keseluruhan.

Daftar Mahasiswa

No.	NIM	Nama Mahasiswa	Jurusan	Alamat	Action
1	123456	allin junikhah	Teknik Informatika	jl. Gajayana 50 Lowokwaru	Edit Hapus
2	123457	Miley Cyrus	Teknik Informatika	jl. Gajayana 50	Edit Hapus

Klik **Action Hapus** pada data yang akan dihapus. Berikutnya akan tampil data seperti di bawah ini, yang menandakan data berkurang dan proses penghapusan berhasil.

Daftar Mahasiswa

No.	NIM	Nama Mahasiswa	Jurusan	Alamat	Action
1	123456	allin junikhah	Teknik Informatika	jl. Gajayana 50 Lowokwaru	Edit Hapus

BAB IV
FRAMEWORK LARAVEL

Tujuan

1. Mampu mengetahui dan memahami implementasi framework selain Codeigniter
2. Mampu membangun aplikasi berbasis website menggunakan framework

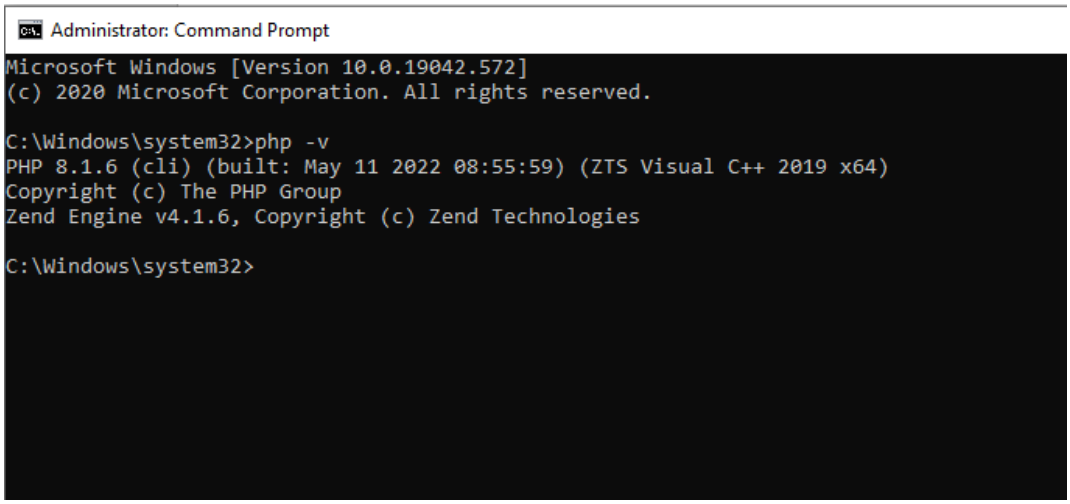
1. Installasi Laravel

Laravel yang digunakan pada modul ini merupakan Laravel versi 9 (terbaru) ketika modul *framework programming* ini disusun. Pada Laravel 9 dibutuhkan PHP dengan versi minimum 8.0. Berikut ini tabel rilis versi Laravel dengan kebutuhan versi PHP.

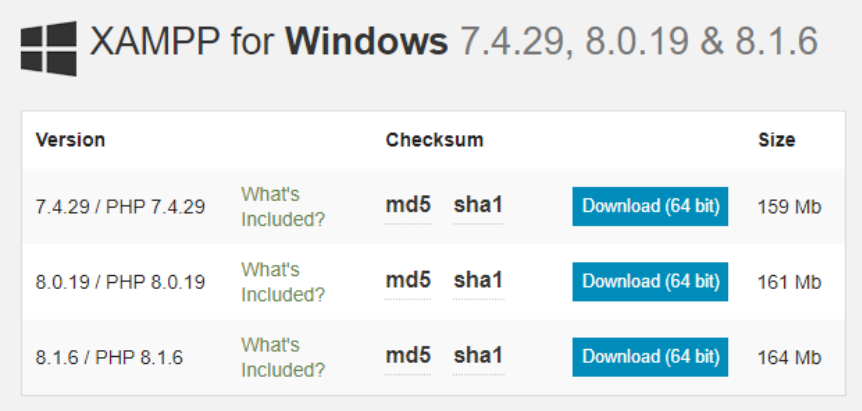
Version	PHP (*)	Release	Bug Fixes Until	Security Fixes Until
6 (LTS)	7.2 - 8.0	September 3rd, 2019	January 25th, 2022	September 6th, 2022
7	7.2 - 8.0	March 3rd, 2020	October 6th, 2020	March 3rd, 2021
8	7.3 - 8.1	September 8th, 2020	July 26th, 2022	January 24th, 2023
9	8.0 - 8.1	February 8th, 2022	August 8th, 2023	February 8th, 2024
10	8.1	February 7th, 2023	August 7th, 2024	February 7th, 2025

Sumber : https://laravel.com/docs/9.x/releases

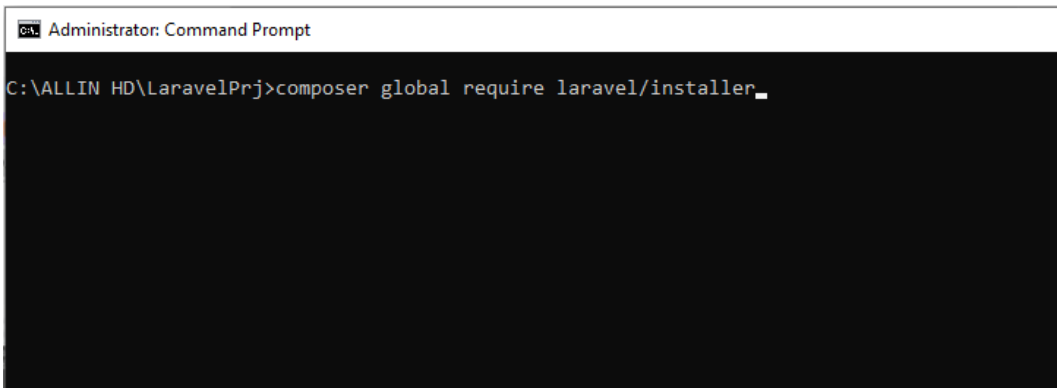
Untuk mengetahui versi PHP kita, masuk pada Terminal (CMD) kemudian pilih **Run as Adminstrator**. Cek versi PHP pada komputer dengan mengetik **PHP -v** pada terminal. Jika versi PHP belum memenuhi kriteria minimum pada Laravel 9, maka kita dapat menginstall kembali dengan versi yang lebih tinggi, atau menggunakan XAMPP versi terbaru yaitu XAMPP 8.1.6.



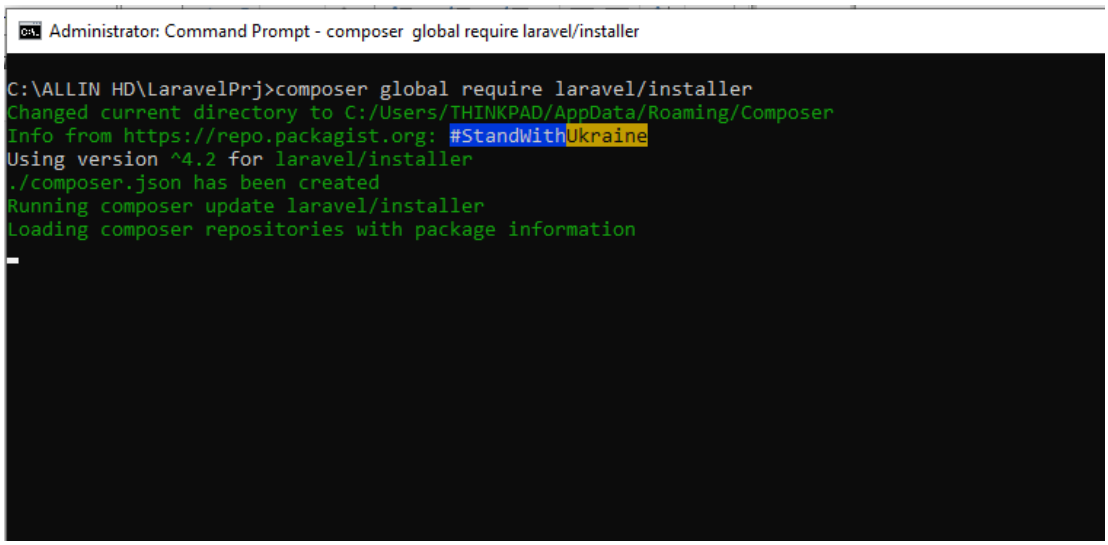
Jika dalam komputer yang sama telah terdapat PHP versi lebih kecil dan masih digunakan dalam aplikasi lain kita dapat melakukan setting pada **edit system environment** (*environment variables*) dan mengubah **Path** pada **System Variables** dengan meletakkan PHP versi terbaru pada posisi atas terhadap PHP versi sebelumnya.

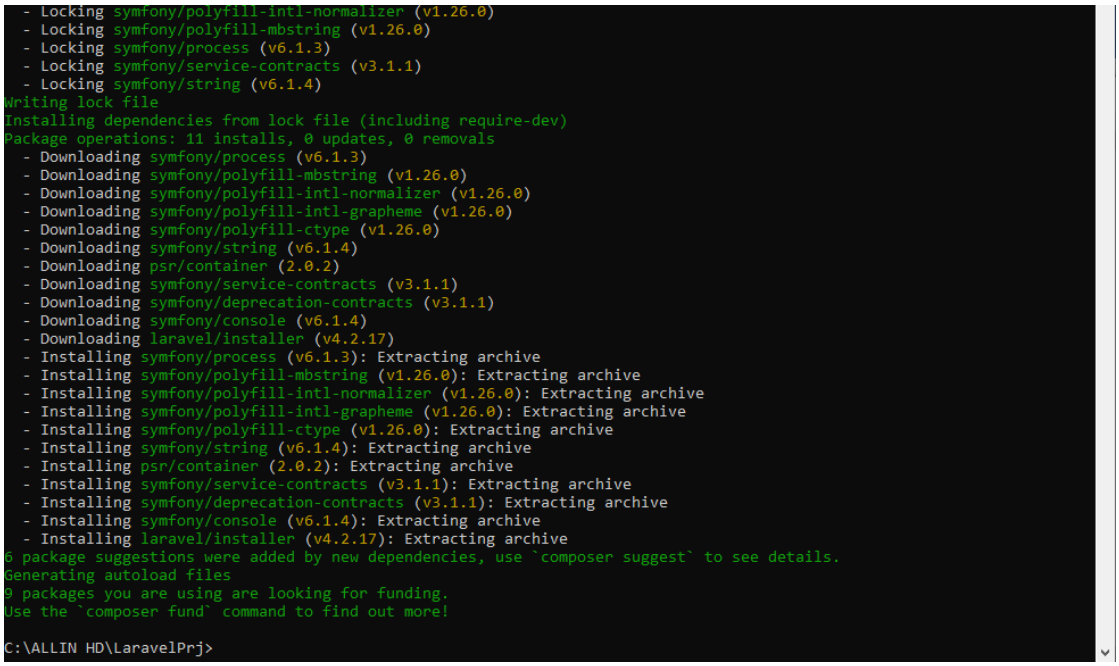


Setelah memenuhi kebutuhan sistem dari Laravel 9, selanjutnya kita dapat melakukan proses Installasi Laravel 9 melalui composer melalui Command Prompt Terminal (CMD) dengan mengetikkan perintah **composer global require Laravel/installer** pada direktori kita meletakkan file Laravel kita.



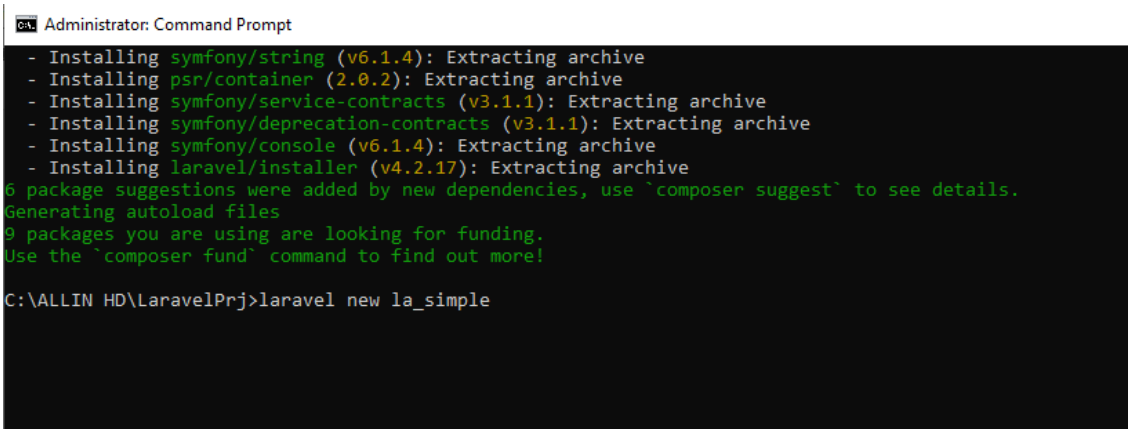
Berikut ini gambaran proses installasi Laravel.



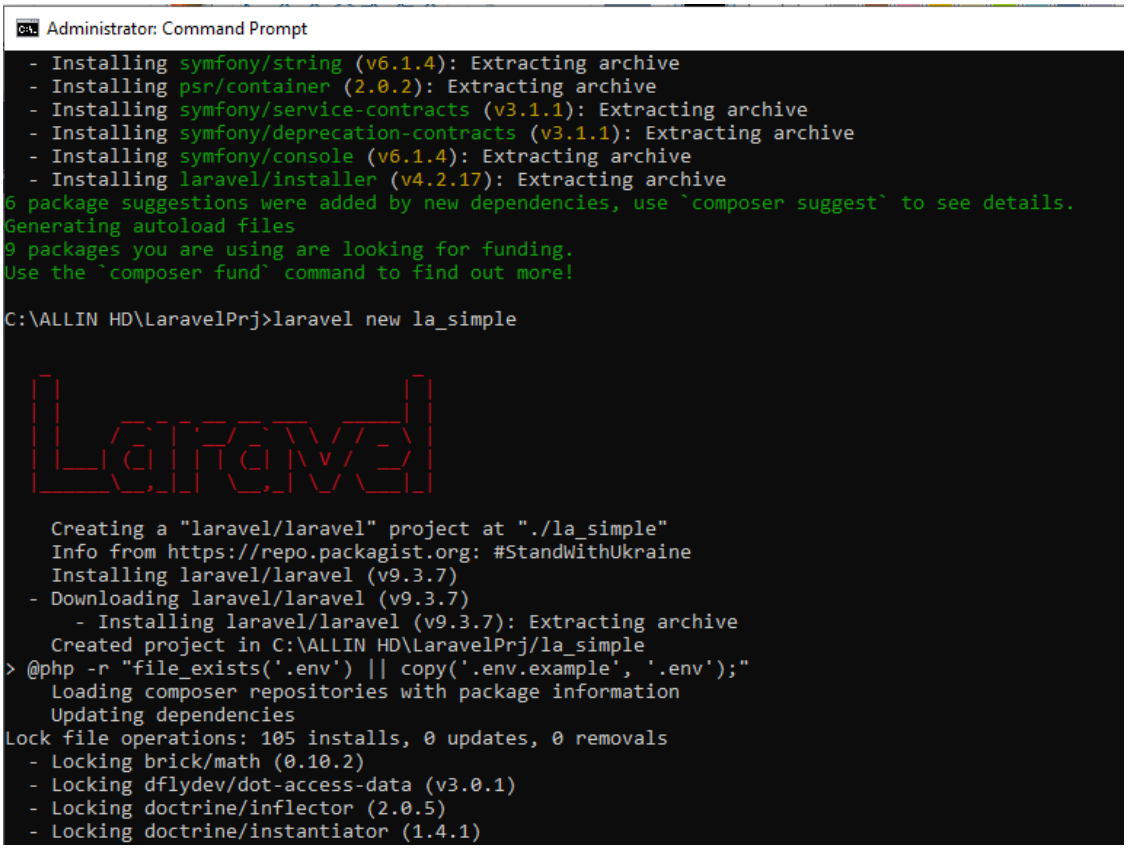


2. Membuat Folder Project Baru pada Laravel

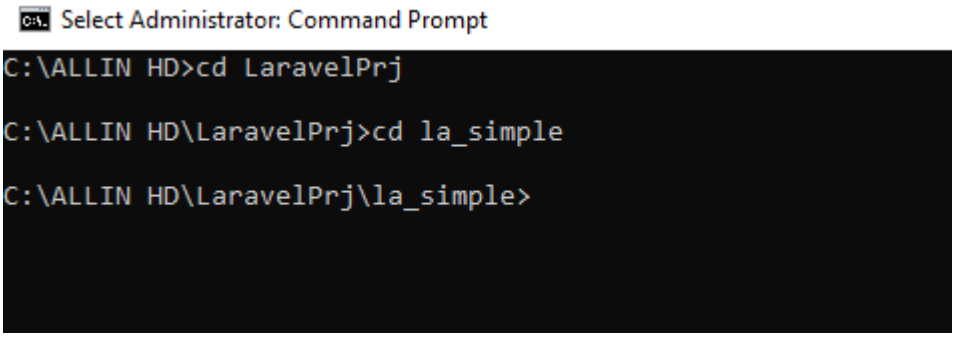
Untuk membuat folder project baru, kita lakukan dengan mengetikkan **Laravel new [nama folder]**



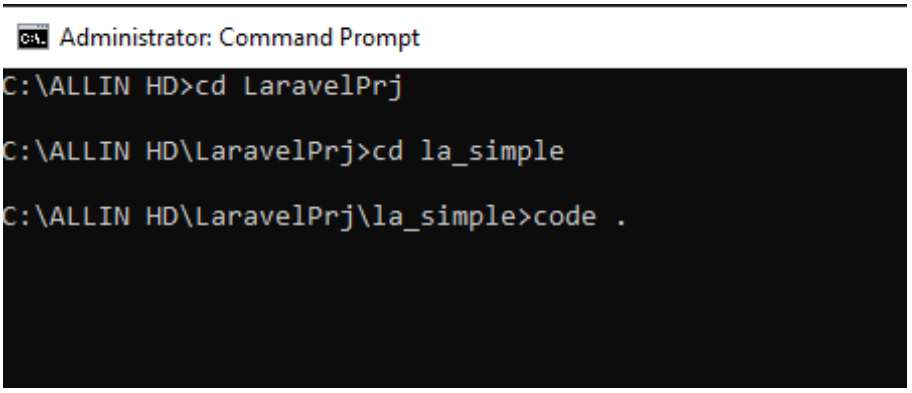
Berikut ini gambaran proses membuat folder project baru pada Laravel



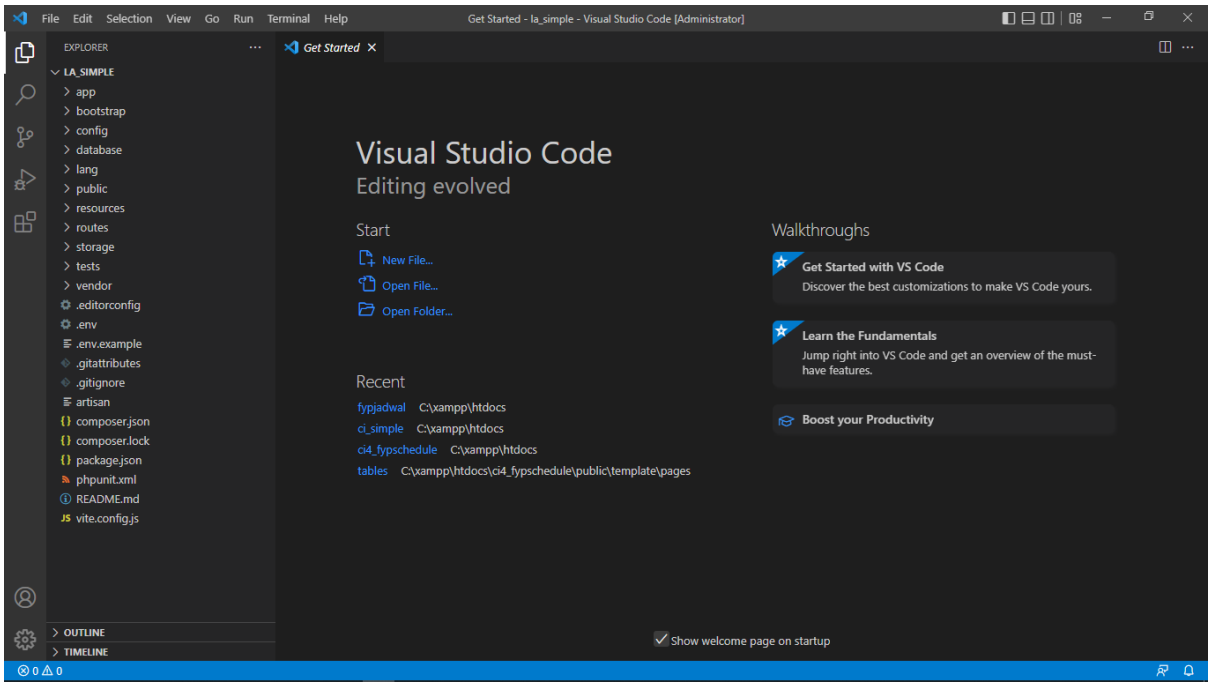
Kemudian masuk pada direktori folder project yang baru yang telah dibuat.



Selanjutnya kita masuk pada visual studio code dengan script di bawah ini.

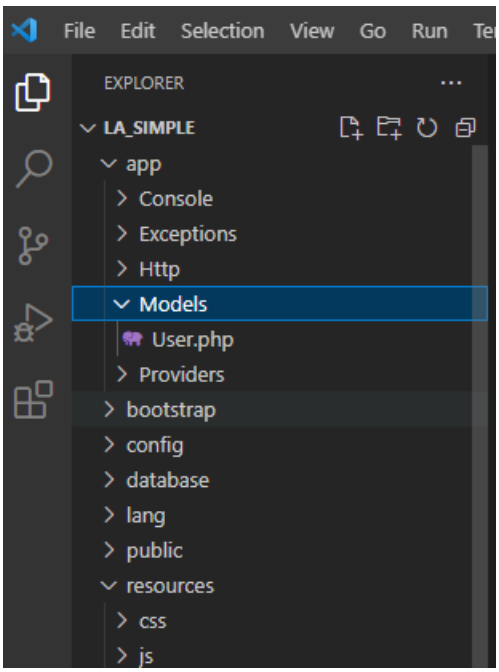


Hingga tampil halaman Visual Studio Code seperti di bawah ini

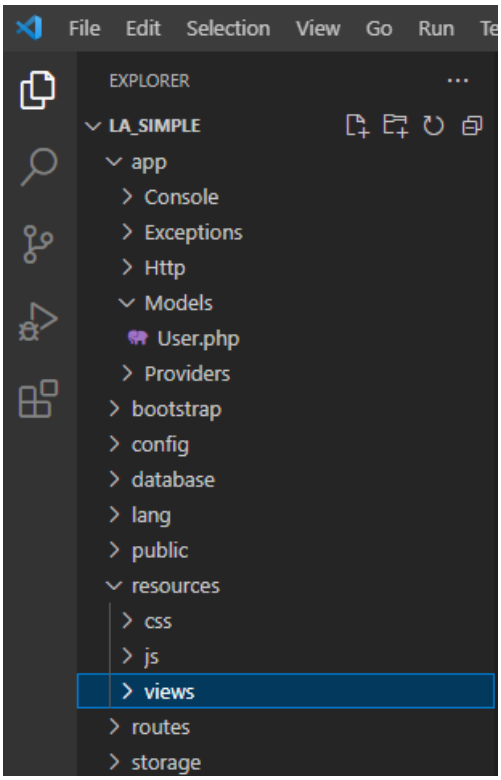


Laravel dengan konsep **Model View Controller** juga memiliki struktur yang hampir sama dengan Codeigniter, hanya saja letak pada penempatan folder **Models, Views, dan Controllers** memiliki penempatan yang berbeda. Berikut ini struktur Models Views dan Controllers pada Laravel.

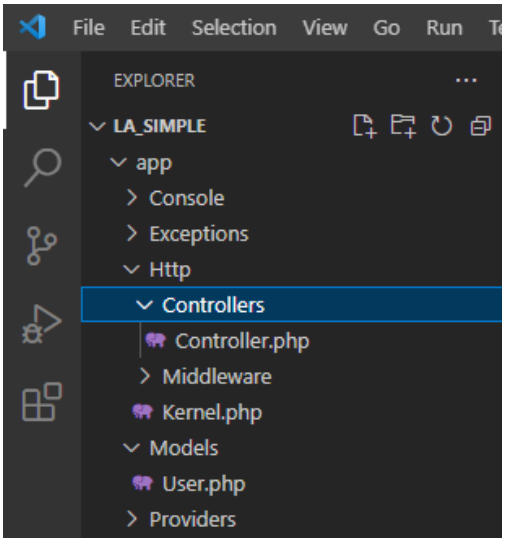
Berikut struktur **Models** pada Laravel.



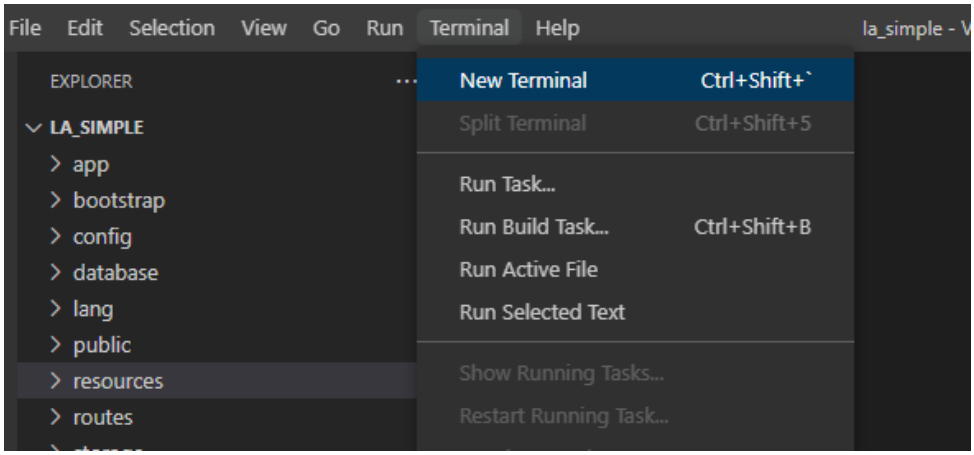
Berikut struktur **Views** pada Laravel.



Berikut struktur **Controllers** pada Laravel.

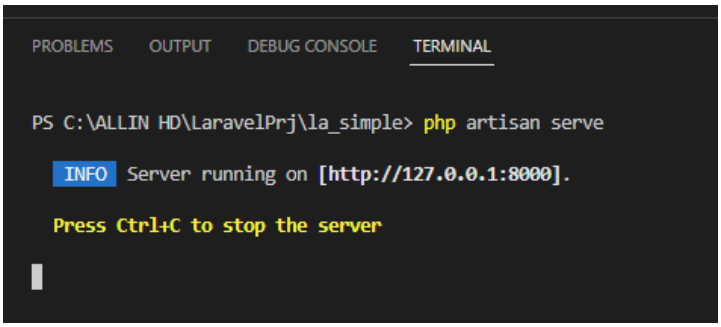


Seperti halnya pada codeigniter kita menjalankan Laravel melalui terminal. Seperti yang dilakukan pada contoh gambar berikut.

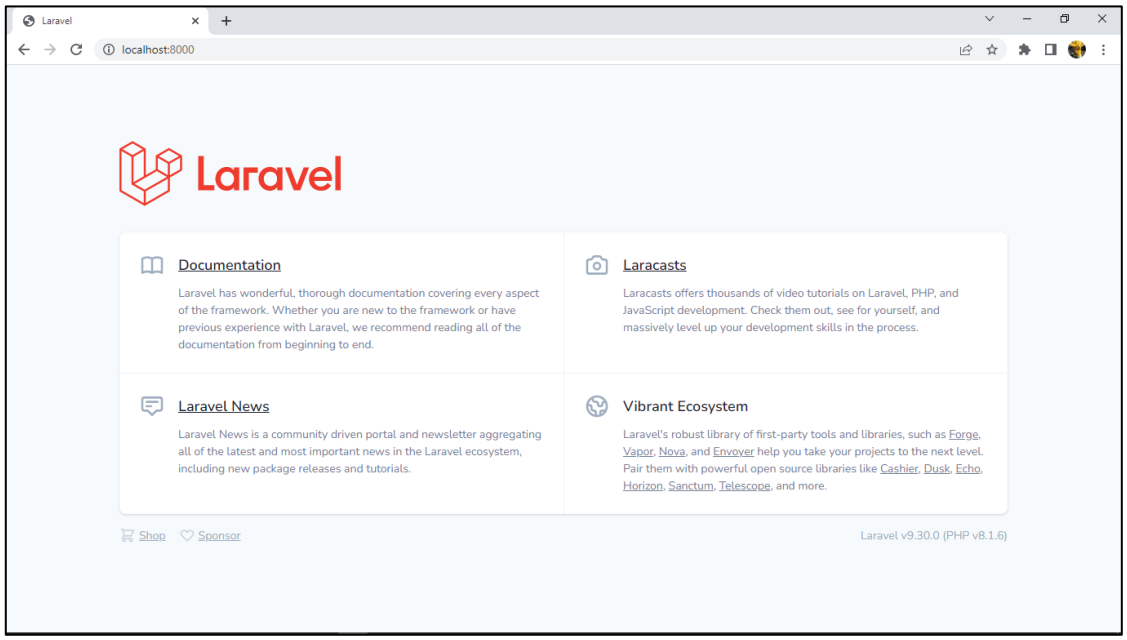


3. Menjalankan Laravel

Untuk menjalankan laravel yang telah kita install, tulis script berikut pada terminal : **php artisan serve**

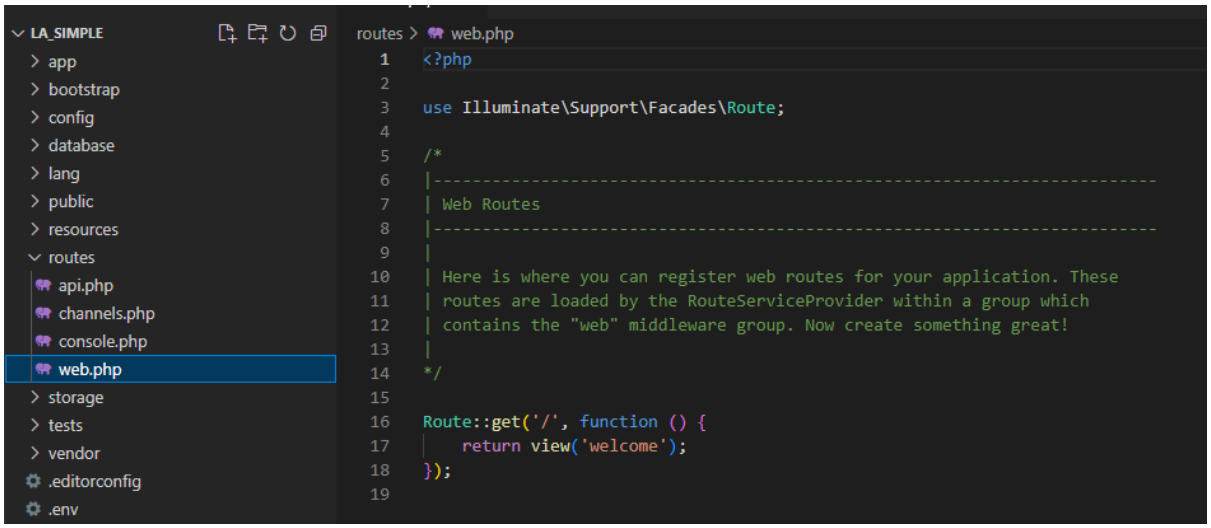


Kemudian buka browser ketikkan alamat base url http://localhost:8000/, aplikasi web Laravel telah dapat dijalankan.



4. ROUTES pada Laravel

Routes pada Laravel merupakan bagian yang mengatur lalu lintas kerja models views dan controllers. Untuk menuliskan routing pada Laravel dilakukan pada file **web.php**



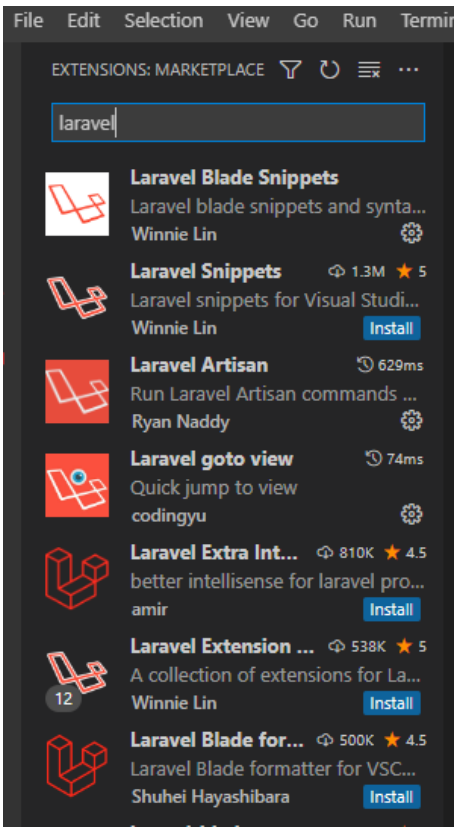
5. Menambahkan EXTENSION Laravel

Klik pada menu extension di sebelah kiri, ketik Laravel pada menu search. Tambahkan beberapa extension yang dibutuhkan:

Extension	Fungsi
Laravel Blade Snippet	Laravel Blade Snippets merupakan salah satu ekstensi wajib karena kita akan sangat terbantu ketika menulis kode pada file blade, ekstensi ini memberikan *autocomplete* untuk syntax - syntax blade.
Laravel Artisan	Ekstensi Laravel Artisan ini dibuat oleh Ryan Naddy yang berguna sebagai jalan pintas dari penggunaan perintah php artisan. Kita dapat menggunakannya dengan kombinasi tombol **cmd + shift + p** untuk pengguna mac OS atau **ctrl + shift + p** untuk pengguna Windows / Linux, lalu kemudian ketik **artisan**.
Laravel goto view	Fungsi dari extension ini adalah shortcut yang memudahkan kita langsung menuju file view dari controller/template yang include file lain hanya dengan menahan tombol **cmd / ctrl** kemudian diikuti dengan klik dari **mouse/trackpad**.
Laravel blade formatter	Extension yang memiliki fungsi melakukan formatting dalam menuliskan script file blade pada Laravel, seperti spasi, tab indent.

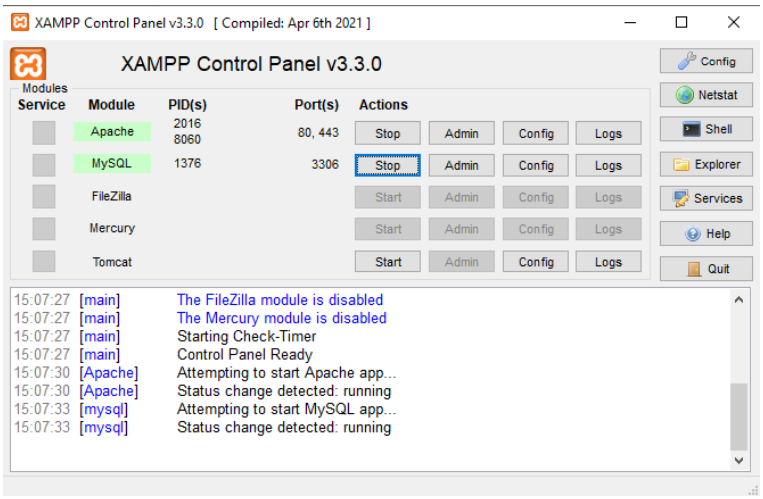
Laravel blade spacer	Laravel blade spacer adalah ekstensi yang menambahkan spasi dan penutup otomatis ketika kita mengetik **{{ atau {!!** dalam file blade
Laravel-goto-components	Extension ini dapat memberikan visual highlight pada beberapa link komponen file **.blade** serta dapat memberikan link shortcut pada komponen melalui **CTRL + click**
Laravel Docs	Ketika mengembangkan aplikasi Laravel tentu kita tidak lepas dengan membuka dokumentasi, karena kita akan menggunakan beberpa fungsi penting di Laravel, ekstensi ini akan memudahkan kita untuk menuju halaman dokumentasi Laravel, yaitu dengan menekan **cmd + shift + p** untuk mac OS atau **ctrl + shift + p** untuk Windows / Linux.

Beberapa extension diatas digunakan untuk mempermudah menuliskan perintah-perintah yang digunakan dalam Laravel.

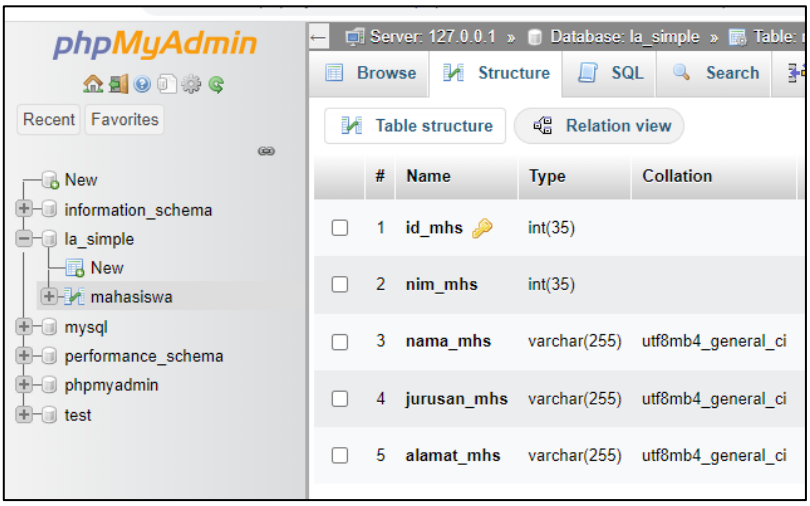


6. Membuat Database dan Konfigurasi Database

Sebelum memulai membuat database, pastikan database mySQL yang berada pada XAMPP terinstall dan aktif.



Masuk pada browser, ketikkan http://localhost/phpmyadmin/ untuk menuju pada database mySQL. Selanjutnya buat database yang dibutuhkan ex: **la_simple** dan beberapa table yang diperlukan



Selanjutnya buat table. Ex: **mahasiswa** dengan beberapa atribut yang dibutuhkan seperti contoh berikut:

#	Name	Type
1	id_mhs	int(35)
2	nim	int(35)
3	nama	varchar(255)
4	jurusan	varchar(255)
5	alamat	varchar(255)

Kemudian isi beberapa data dalam database

	id_mhs	nim_mhs	nama_mhs	jurusan_mhs	alamat_mhs
Edit Copy Delete	1	123456	allin j	Teknik Informatika	Jl. Gajayana 50 Lowokwaru
Edit Copy Delete	2	123457	miley cyrus	Teknik Arsitektur	Jl. Ganesha 10

Untuk melakukan konfigurasi database, pilih file **.env** kemudian ubah nama project dan beberapa settingan database lain sesuai dengan data kita.

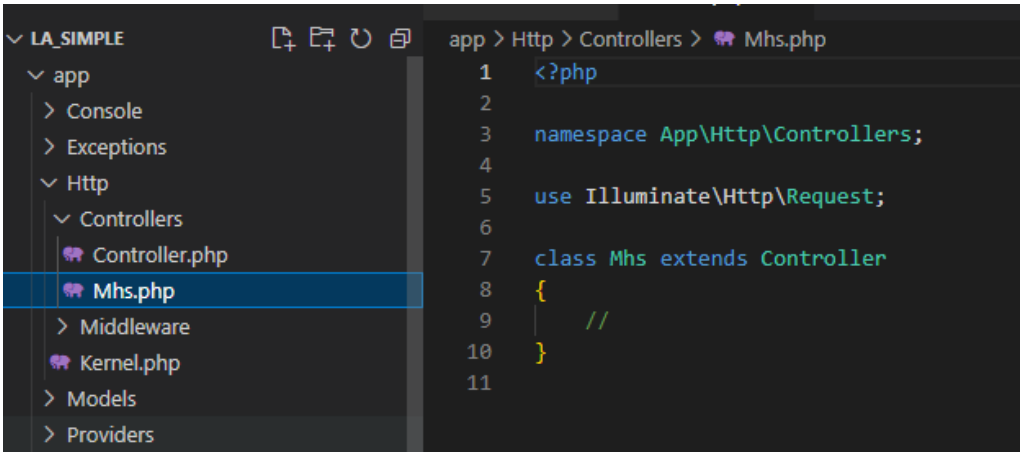
```
.env
.env
1   APP_NAME=la_simple
2   APP_ENV=local
3   APP_KEY=base64:G+1PN30TrdNjvGygmj3gWYtQHdWzjfI5FwX3QL/8q1A=
4   APP_DEBUG=true
5   APP_URL=http://localhost
6
7   LOG_CHANNEL=stack
8   LOG_DEPRECATIONS_CHANNEL=null
9   LOG_LEVEL=debug
10
11  DB_CONNECTION=mysql
12  DB_HOST=127.0.0.1
13  DB_PORT=3306
14  DB_DATABASE=la_simple
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
```

7. Membuat Project Baru Aplikasi Sederhana

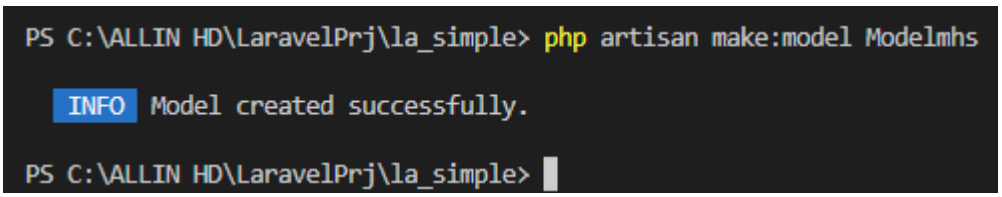
Langkah awal dalam membuat project baru ini yaitu membuat Controller **Mhs** melalui terminal dengan script sebagai berikut: **Php artisan make:controller Mhs**

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS C:\ALLIN HD\LaravelPrj\la_simple> php artisan make:controller Mhs

  INFO  Controller created successfully.

PS C:\ALLIN HD\LaravelPrj\la_simple>
```

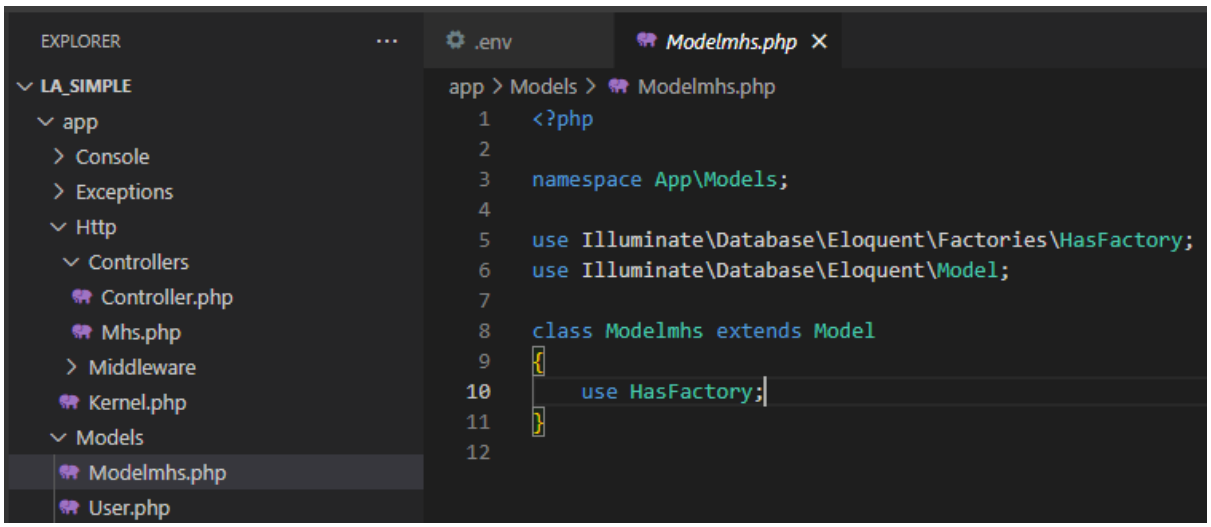
Selain melalui terminal, kita juga dapat membuat controller melalui folder secara langsung seperti gambar berikut ini.



Selanjutnya kita membuat model dengan script yang hampir sama dengan pembuatan controller, yaitu **Php artisan make:model Modelmhs,** hingga file ModelMhs.php terbentuk.



Kelebihan menggunakan terminal pada pembuatan baik controller dan model, secara otomatis kita akan dibuatkan potongan script yang siap digunakan seperti di bawah ini



Kemudian tambahkan beberapa script sesuaikan dengan database kita.

Modelmhs.php

app > Models > Modelmhs.php

```php
<?php

namespace App\Models;

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

class Modelmhs extends Model
{
    use HasFactory;

    protected $table = 'mahasiswa';
    protected $primaryKey = 'id_mhs';
    public $timestamps = false;
}
```

A. READ DATA

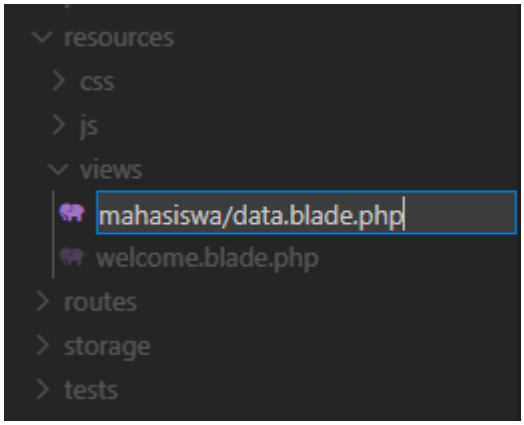
Pada proses Read/View data pada database yang telah kita isi sebelumnya, buat return view pada **Controllers Mhs/index** dimana akan menampilkan **mahasiswa.data** yang diload pertama kali pada aplikasi web, yang artinya mengarah pada file **Views mahasiswa/data.blade.php**.

Controllers Mhs.php

app > Http > Controllers > Mhs.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class Mhs extends Controller
{
    public function index(){
        return View('mahasiswa.data');
    }
}
```

Karena folder Views mahasiswa dan data.blade.php belum tersedia maka buat file yang dibutuhkan terlebih dahulu seperti gambar berikut ini.



Kemudian Isi file data.blade.php dengan script seperti berikut.

Views mahasiswa/data.blade.php

```
<style>
    table, th, td {
      border: 1px solid black;
    }
    </style>
    <center>
        <h3>Daftar Mahasiswa</h3>
        <table style="width:60%">
        <thead>
            <tr>
                <th>No.</th>
                <th>NIM</th>
                <th>Nama Mahasiswa</th>
                <th>Jurusan</th>
                <th>Alamat</th>
            </tr>
        </thead>
        <tbody>
            <?php $no=0; foreach($dataMhs as $row):$no++;?>
            <tr>
                <th> <?= $no; ?></th>
                <td><?= $row->nim_mhs; ?></td>
                <td><?= $row->nama_mhs; ?></td>
                <td><?= $row->jurusan_mhs; ?></td>
                <td><?= $row->alamat_mhs; ?></td>
            </tr>
            <?php endforeach;?>
        </tbody>
    </table>
    </center>
```

Selanjutnya akses aplikasi web seperti gambar berikut. Pastikan sebelumnya server telah berjalan, yaitu dengan mengetikkan **php artisan serve** pada Terminal.

Dapat kita lihat halaman dapat diakses namun masih kosong / tidak ditemukan, hal ini karena kita belum melakukan routing pada route yang digunakan (Mhs/index). Oleh karena itu, selanjutnya kita akan menambahkan routing pada **Routes web.php.**

Tambahkan **use App\Http\Controllers\Mhs;** terlebih dahulu.

Routes web.php

routes > web.php

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Mhs;

/*
|--------------------------------------------------------------------------
| 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('welcome');
});

Route::controller(Mhs::class)->group(function () {
    Route::get('/Mhs/index', 'index');
});
```

Akses halaman http://localhost:8000/Mhs/index sekali lagi. Dapat kita lihat data telah berhasil ditampilkan.

Daftar Mahasiswa

No.	NIM	Nama Mahasiswa	Jurusan	Alamat
1	123456	allin j	Teknik Informatika	Jl. Gajayana 50 Lowokwaru
2	123457	miley cyrus	Teknik Arsitektur	Jl. Ganesha 10

B. CREATE DATA

Untuk create data (menambahkan data), terlebih dahulu kita menambahkan tombol/button form **Tambah Data Mahasiswa** yang terletak pada **Views mahasiswa/data.blade.php,** dimana tombol ini akan mengantarkan kita pada form isian data baru mahasiswa. Buat button seperti pada gambar berikut ini.

Views mahasiswa/data.blade.php

```
<style>
    table, th, td {
      border: 1px solid black;
    }
    </style>
    <center>
        <p>
            <button type="button" onclick="window.location='/Mhs/tambah'">Tambah Data Mahasiswa
            </button>
        </p>
        <h3>Daftar Mahasiswa</h3>
        <table style="width:60%">
        <thead>
            <tr>
                <th>No.</th>
                <th>NIM</th>
                <th>Nama Mahasiswa</th>
                <th>Jurusan</th>
                <th>Alamat</th>
            </tr>
        </thead>
        <tbody>
            <?php $no=0; foreach($dataMhs as $row):$no++;?>
            <tr>
                <th> <?= $no; ?></th>
                <td><?= $row->nim_mhs; ?></td>
                <td><?= $row->nama_mhs; ?></td>
```

Selanjutnya kita menambahkan routing untuk menuju method **add()**. **Mhs/tambah** merupakan route yang diakses oleh user. Method **add()** merupakan method yang dipanggil oleh route tersebut.

Routes web.php

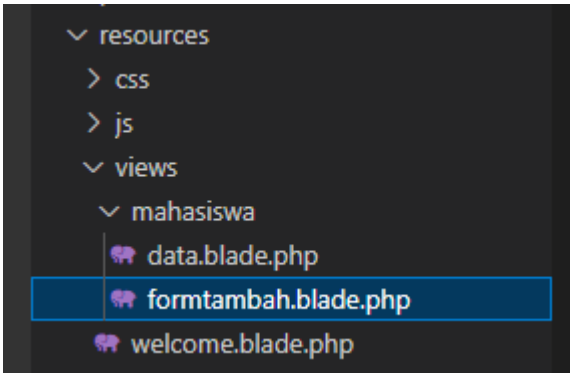
```
Route::get('/', function () {
    return view('welcome');
});

Route::controller(Mhs::class)->group(function () {
    Route::get('/Mhs/index', 'index');
    Route::get('/Mhs/tambah', 'add');
});
```

Karena method ini belum tersedia, buat terlebih dahulu methode **add()** pada **Controllers Mhs**.

Controllers Mhs.php

```
    }
    public function add(){
        return View('mahasiswa.formtambah');
    }
```

Method Add() pada **Controllers** **Mhs/Add** akan menampilkan **mahasiswa.formtambah** yang artinya akan memanggil folder mahasiswa file **formtambah.blade.php.** Karena file ini belum tersedia maka kita akan membuatnya terlebih dahulu pada struktur **Views**.

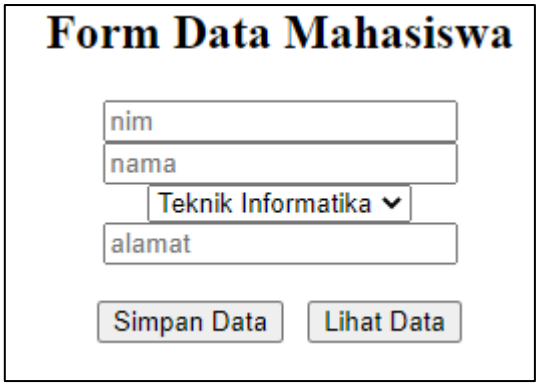


Selanjutnya isi file formtambah.blade.php dengan script seperti berikut.

Views mahasiswa/Formtambah.blade.php

```
<center><h2>Form Data Mahasiswa</h2>
    <form method="POST" action="{{ url('/Mhs/simpan')}} " >
        @csrf
    <input type="text" name="nim_mhs" placeholder="nim">
    <br>
    <input type="text" name="nama_mhs" placeholder="nama">
    <br>
    <select name="jurusan_mhs">
     <option value="Teknik Informatika">Teknik Informatika</option>
     <option value="Teknik Arsitektur">Teknik Arsitektur</option>
    </select>
    <br>
    <input type="text" name="alamat_mhs" placeholder="alamat">
    <br><br>
    <button type="submit">Simpan Data</button>
     
    <button type="button" onclick="window.location='/Mhs/index'">Lihat Data
    </button>
    </form></center>
```

Berikut tampilan Form Data Mahasiswa pada browser.



Untuk melakukan proses penyimpanan kita akan melakukan routing action Url **/Mhs/simpan** yang dikerjakan oleh method **simpanMhs()** pada **Controllers** **Mhs.php**. Sebelum kita membuat method tersebut, lakukan routing terlebih dahulu.

Routes web.php

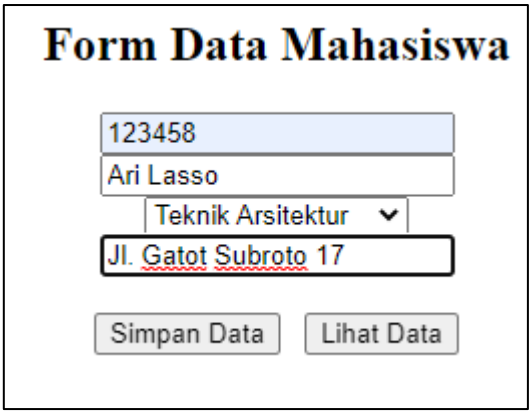
```
Route::controller(Mhs::class)->group(function () {
    Route::get('/Mhs/index', 'index');
    Route::get('/Mhs/tambah', 'add');
    Route::post('/Mhs/simpan', 'simpanMhs');
});
```

Buat method **simpanMhs()** pada **controllers** **Mhs.php** seperti berikut.

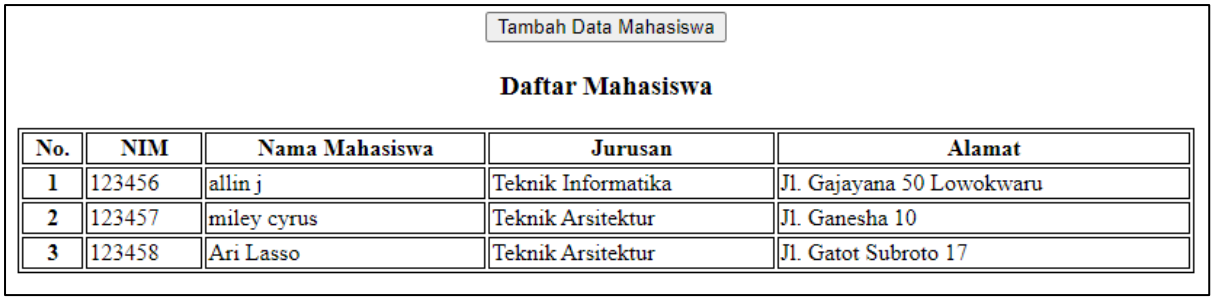
Controllers Mhs.php

```
public function simpanMhs(Request $r){
    $nim=$r->nim_mhs;
    $nama=$r->nama_mhs;
    $jurusan=$r->jurusan_mhs;
    $alamat=$r->alamat_mhs;


    $mhs= new Modelmhs;
    $mhs->nim_mhs =$nim;
    $mhs->nama_mhs =$nama;
    $mhs->jurusan_mhs =$jurusan;
    $mhs->alamat_mhs =$alamat;
    $mhs->save();

    return View('mahasiswa.formtambah');

}
```

Method dan routing sudah dilakukan, selanjutnya akses form tambah data dan isi form dengan data yang diinginkan kemudian lakukan simpan data.



Data yang telah ditambahkan telah masuk dalam daftar mahasiswa, yang menunjukkan data telah berhasil bertambah.



No.	NIM	Nama Mahasiswa	Jurusan	Alamat
1	123456	allin j	Teknik Informatika	Jl. Gajayana 50 Lowokwaru
2	123457	miley cyrus	Teknik Arsitektur	Jl. Ganesha 10
3	123458	Ari Lasso	Teknik Arsitektur	Jl. Gatot Subroto 17

C. UPDATE DATA

Untuk mengubah data yang telah ada, sebelumnya kita akan membuat button aksi update/edit data pada file Views mahasiswa/data.blade.php seperti berikut ini.

Views mahasiswa/data.blade.php

```
<tbody>
    <?php $no=0; foreach($dataMhs as $row):$no++;?>
    <tr>
        <th> {{ $no; }}</th>
        <td>{{ $row->nim_mhs; }}</td>
        <td>{{ $row->nama_mhs; }}</td>
        <td>{{ $row->jurusan_mhs; }}</td>
        <td>{{ $row->alamat_mhs; }}</td>
        <td><button type="button" onclick="window.location='/Mhs/formedit/{{ $row->id_mhs }}'">Edit</bu
    </tr>
    <?php endforeach;?>
</tbody>
```

Setiap kali kita membuat link route secara bersamaan kita diharapkan untuk dapat melakukan routing sebagai tindak lanjut action route yang telah kita buat pada button tersebut. Sebagai contoh button aksi edit akan mengarah pada form edit dengan route **/Mhs/formedit** dimana akan menjalankan method **formedit()** pada **Controllers Mhs.php.**

Routes web.php

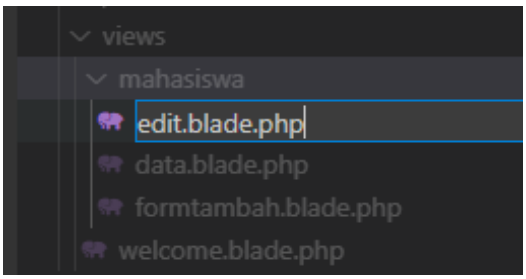
```
Route::controller(Mhs::class)->group(function () {
    Route::get('/Mhs/index', 'index');
    Route::get('/Mhs/tambah', 'add');
    Route::post('/Mhs/simpan', 'simpanMhs');
    Route::get('/Mhs/formedit/{idmhs}', 'formedit');

});
```

Setelah melakukan proses routing pada button yang kita buat, selanjutnya membuat method **formedit()** yang telah kita sebut sebelumnya.

Controllers Mhs.php

```
public function formedit($idMhs){
    $mhs = Modelmhs::find($idMhs);
    $data=['id'=>$mhs->id_mhs,
    'nim'=>$mhs->nim_mhs,
    'nama'=>$mhs->nama_mhs,
    'jurusan'=>$mhs->jurusan_mhs,
    'alamat'=>$mhs->alamat_mhs];
    return View('mahasiswa.edit',$data);
}
```

Method formedit() ini selanjutnya akan memanggil data dari Modelmhs yang ada untuk ditampilkan pada form edit pada **Views mahasiswa/edit.blade.php.** Karena file ini belum tersedia kita dapat membuatnya terlebih dahulu.



Kemudian isi dengan script seperti pada gambar berikut.

Views mahasiswa/edit.blade.php

resources > views > mahasiswa > edit.blade.php > center

```
<center><h2>Form Edit Data Mahasiswa</h2>
    <form method="POST" action="{{ url('/Mhs/edit')}} " >
        @csrf
        @method('PUT')
        <input type="hidden" class="form-control" name="idMhs" value="<?= $id; ?>">
        <label>NIM</label>
        <input type="text" class="form-control" name="nimMhs" value="<?= $nim; ?>">
        <br>
        <label>Nama mahasiswa</label>
        <input type="text" class="form-control" name="namaMhs" value="<?= $nama; ?>">
        <br>
        <label>Jurusan</label>
        <select name="jurusanMhs">
            <option><?= $jurusan; ?></option>
            <option value="Teknik Informatika">Teknik Informatika</option>
            <option value="Teknik Arsitektur">Teknik Arsitektur</option>
        </select>
        <br>
        <label>Alamat</label>
        <input type="text" class="form-control" name="alamatMhs" value="<?= $alamat; ?>">
        <br><br>
        <button type="submit">Simpan Data</button>
        <button type="button" onclick="window.location='/Mhs/index'">Lihat Data
        </button>
    </form>
</center>
```

Jika sebelumnya kita melakukan routing yang menuju pada method **formedit()** untuk load halaman form untuk proses edit, pada proses selanjutnya kita melakukan routing kembali yang mengarah pada method **edit()** dimana method ini memproses perubahan data dari form yang ada.

Routes web.php

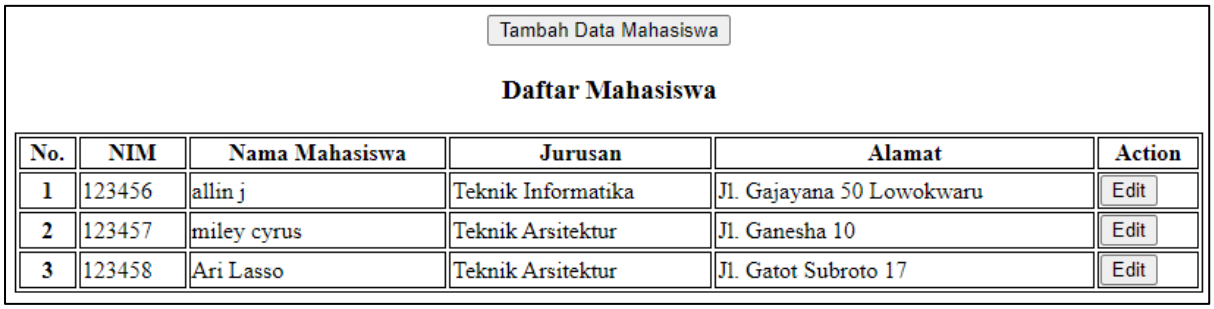
```
Route::controller(Mhs::class)->group(function () {
    Route::get('/Mhs/index', 'index');
    Route::get('/Mhs/tambah', 'add');
    Route::post('/Mhs/simpan', 'simpanMhs');
    Route::get('/Mhs/formedit/{idmhs}', 'formedit');
    Route::put('/Mhs/edit', 'edit');
});
```

Karena method **edit()** belum tersedia, maka kita akan membuatnya pada **Controllers Mhs.php** seperti berikut ini.

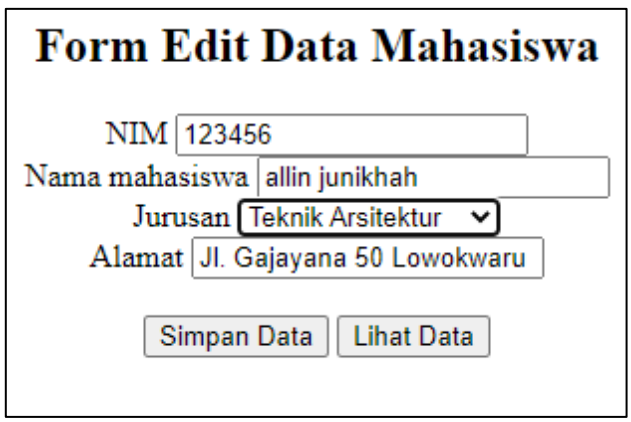
Controllers Mhs.php

```php
public function edit(Request $r){
    $id=$r->idMhs;
    $nim=$r->nimMhs;
    $nama=$r->namaMhs;
    $jurusan=$r->jurusanMhs;
    $alamat=$r->alamatMhs;

    $mhs= Modelmhs::find($id);
    $mhs->nim_mhs =$nim;
    $mhs->nama_mhs =$nama;
    $mhs->jurusan_mhs =$jurusan;
    $mhs->alamat_mhs =$alamat;
    $mhs->save();
    return redirect('/Mhs/index');
}
```

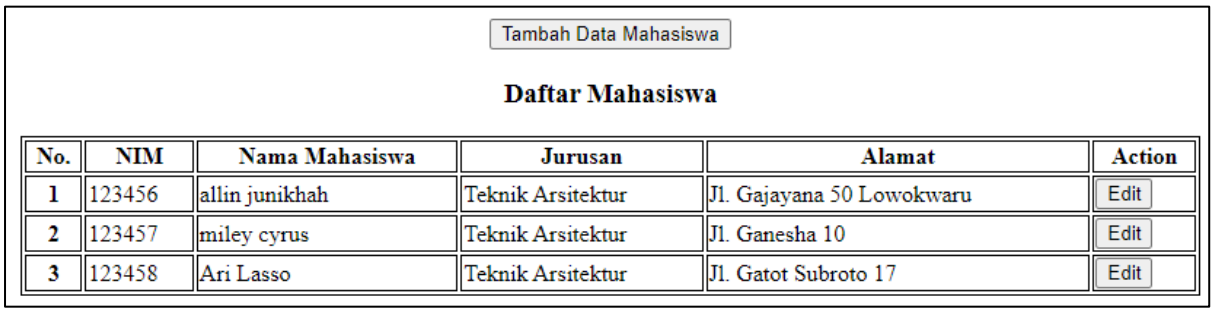
Button aksi, routing dan method telah tersedia. Maka proses editpun telah dapat dilakukan. Kita dapat mencobanya dengan mengedit sebuah data yang ada pada daftar Mahasiswa.



Selanjutnya akan muncul form untuk proses pengubahan data, isi sesuai dengan data yang ingin diganti, lalu klik **Simpan Data** untuk menyimpan.



Proses update/edit berhasil dapat kita lihat dengan perubahan data yang ada pada daftar Mahasiswa berikut ini.



D. DELETE DATA

Pada proses delete data, seperti sebelumnya kita akan menambahkan button aksi hapus yang akan mengantarkan pada proses delete. Button akan kita letakkan pada setiap data yang ditampilkan oleh **Views mahasiswa/data.blade.php.** Berikut ini potongan script untuk menambahkan button aksi HAPUS.

Views mahasiswa/data.blade.php

```
<tr>
    <th> {{ $no; }}</th>
    <td>{{ $row->nim_mhs; }}</td>
    <td>{{ $row->nama_mhs; }}</td>
    <td>{{ $row->jurusan_mhs; }}</td>
    <td>{{ $row->alamat_mhs; }}</td>
    <td><button type="button" onclick="window.location='/Mhs/formedit/{{ $row->id_mhs }}'">Edit</
    <form method="POST" action="/Mhs/hapus/{{ $row->id_mhs }}" style="display: inline"
        onsubmit="return KonfirmHapus()">
    @csrf
    @method('DELETE')
    <button type="submit">HAPUS</button>
    </form>
    </td>
</tr>
```

Tambahkan serta script sederhana seperti berikut untuk melakukan konfirmasi penghapusan.

Views mahasiswa/data.blade.php

```
            <?php endforeach;?>
        </tbody>
    </table>
    </center>
    <script>
        function KonfirmHapus(){
            msg=confirm('Yakin data di hapus?');
            if(msg)
            return true;
            else
            return false;
        }
    </script>
```

Button yang telah kita buat memiliki route **/Mhs/hapus/{idmhs}** yang perlu dilakukan routing terhadap method yang bekerja padanya. Seperti sebelumnya lakukan routing pada **Routes web.php**.

Routes web.php

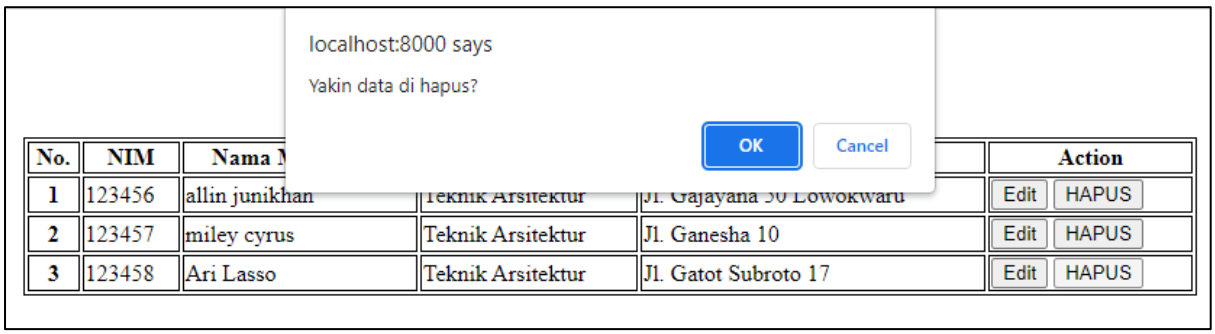
```
Route::controller(Mhs::class)->group(function () {
    Route::get('/Mhs/index', 'index');
    Route::get('/Mhs/tambah', 'add');
    Route::post('/Mhs/simpan', 'simpanMhs');
    Route::get('/Mhs/formedit/{idmhs}', 'formedit');
    Route::put('/Mhs/edit', 'edit');
    Route::delete('/Mhs/hapus/{idmhs}', 'hapus');
});
```

Setelah dilakukan routing, buat method yang bekerja pada route tersebut yaitu method **hapus()**.

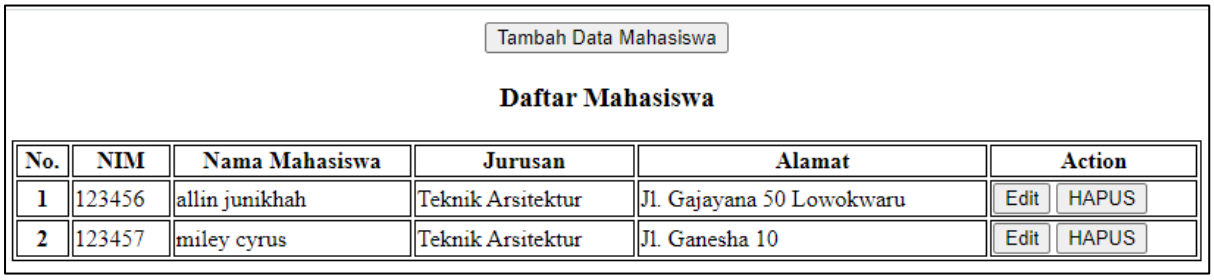
Controllers Mhs.php

```
public function hapus($idMhs){
    Modelmhs::find($idMhs)->delete();
    return redirect()->back();
}
```

Button aksi hapus, routing, dan method telah tersedia. Selanjutnya proses delete data telah dapat dilakukan. Sebagai contoh lakukan penghapusan pada salah satu data pada daftar mahasiswa.



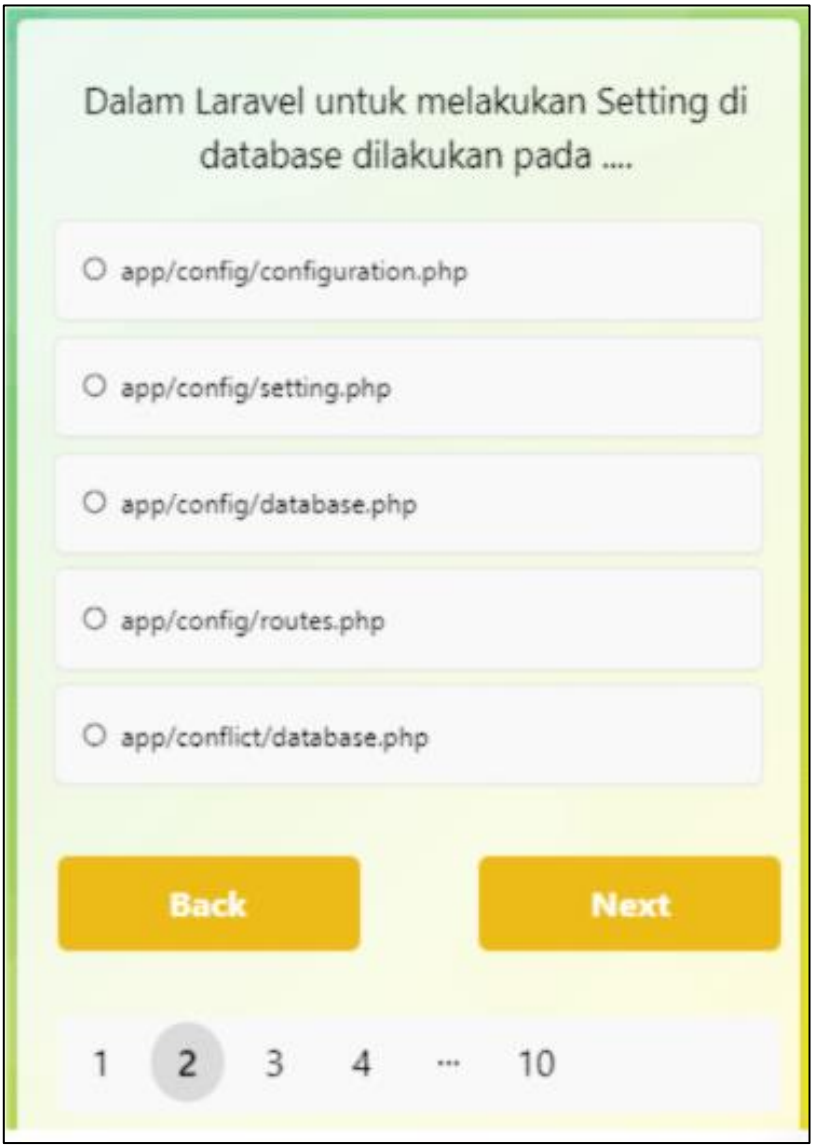
Proses delete data telah berhasil, dapat dilihat dari daftar mahasiswa yang telah berkurang.



QUIZ FRAMEWORK PROGRAMMING

Dapat diakses melalui e-module dengan link :

https://heyzine.com/flip-book/5706f7b478.html#page/59



DAFTAR REFERENSI

Accidental Teacher. (2021, September 26). *Membuat Buku Digital di Canva | Flipbook - YouTube*. Youtube. https://www.youtube.com/watch?v=3NKMKCnPU4s

Basuki, A. P. (2019). *Konsep dan Implementasi Pemrograman Laravel 5*. Lokomedia.

Berbasis Framework, P., & Santoso, H. (n.d.). *Framework CodeIgniter*. http://www.atmaluhur.ac.id

Dewi, E. R. (2022, April 3). *Membuat Flipbook dengan Canva dan Heyzine - YouTube*. Youtube. https://www.youtube.com/watch?v=tgc94JLGmUo

Installation — CodeIgniter 4.2.6 documentation. (n.d.). Retrieved September 12, 2022, from https://codeigniter.com/user_guide/installation/index.html

Kurniawan, A. (2020, May 25). *Tutorial Codeigniter 4 #3: Codeigniter Environment - Indonetsource*. https://www.indonetsource.com/codeigniter-environment/

Nisa, L. (2021, March 21). *Tutorial Desain dan Publish E Modul dengan Flip PDF Corporate|Part 4 - YouTube*. Youtube. https://www.youtube.com/watch?v=GlMNjijn5Rk

Pojok Belajar. (2021, July 23). *Membuat e-Modul Interaktif Lengkap Ada Teks, Image, Audio, Video, Survey, Kuis, dan QR code - YouTube*. Youtube. https://www.youtube.com/watch?v=6yDMcXQUObE&t=692s

Rol Asmi, A., E-Modul Berbasis Flip Book Maker, P., & Novemy Dhita Surbakti, A. B. (2018). Pengembangan E-Modul Berbasis Flip Book Maker Materi Pendidikan Karakter untuk Pembelajaran Mata Kuliah Pancasila MPK Universitas Sriwijaya. *JPIS Jurnal Pendidikan Ilmu Sosial*, *27*(1). http://ejournal.upi.edu/index.php/jpis

Sa'diyah, K. (2021). Pengembagan E-Modul Berbasis Digital Flipbook Untuk Mempermudah Pembelajaran Jarak Jauh Di SMA. *EDUKATIF : JURNAL ILMU PENDIDIKAN*, *3*(4), 1298–1308. https://doi.org/10.31004/edukatif.v3i4.561

Subagia, A. (2017). *Membangun Aplikasi dengan Codeigniter dan Database SQL Server*. PT Elex Media Komputindo.

Syauqi, A., Teguh WA, A., Mujtahidah, A., Yulia K, F., & Heparyanti S, A. (2020). *Menguasai MariaDB Menggunakan Laragon* (Tim Laboratotium Database, Ed.). UIN Maliki Press.

Widiana, F. H., & Rosy, B. (2021). Pengembangan E-Modul Berbasis Flipbook Maker pada Mata Pelajaran Teknologi Perkantoran. *EDUKATIF : JURNAL ILMU PENDIDIKAN*, *3*(6), 3728–3739. https://doi.org/10.31004/edukatif.v3i6.1265