

JOBSHEET 10 MEMBUAT CRUD MENGGUNAKAN LARAVEL PEMROGRAMAN WEB LANJUT



Oleh :
Inayati Machsus Izza Addin **1841720202**

**Program Studi D-IV Teknik Informatika
Jurusan Teknologi Informasi
Politeknik Negeri Malang
2020**

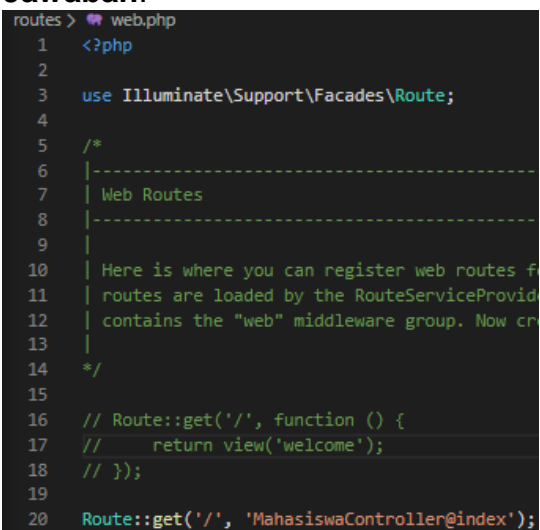
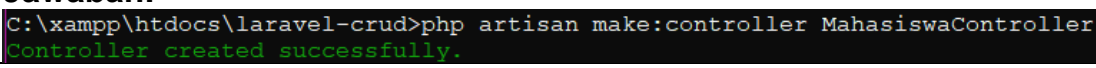
Praktikum – Bagian 1: Membuat CRUD di Laravel menggunakan Query Builder

a. Konfigurasi Database dan Pembuatan Tabel di MySQL

Langka h	Keterangan
1	<p>Buatlah project Laravel baru dengan nama laravel-crud. Buka command prompt, tuliskan perintah berikut.</p> <pre>cd C:\xampp\htdocs laravel new laravel-crud</pre> <p>Jawaban:</p> <pre>C:\xampp\htdocs>laravel new laravel-crud Crafting application... Loading composer repositories with package information Installing dependencies (including require-dev) from lock file Package operations: 92 installs, 0 updates, 0 removals - Installing doctrine/inflector (1.3.1): Loading from cache - Installing doctrine/lexer (1.2.0): Loading from cache - Installing dragonmantank/cron-expression (v2.3.0): Loading fr om cache 15 packages you are using are looking for funding. Use the composer fund command to find out more! > @php -r "file_exists('.env') copy('.env.example', '.env');" > > @php artisan key:generate --ansi Application key set successfully. > illuminate\Foundation\ComposerScripts::postAutoloadDump > @php artisan package:discover --ansi Discovered Package: facade/ignition Discovered Package: fideloper/proxy Discovered Package: fruitcake/laravel-cors Discovered Package: laravel/tinker Discovered Package: nesbot/carbon Discovered Package: nunomaduro/collision Package manifest generated successfully. Application ready! Build something amazing. C:\xampp\htdocs></pre>
2	<p>Selanjutnya kita lakukan konfigurasi database di Laravel. Untuk melakukan konfigurasi database, bukalah file .env pada project laravel-crud. Ubah seperti di bawah ini.</p> <p>Keterangan:</p> <ul style="list-style-type: none">- Nama database yang akan digunakan adalah latihan_laravel dengan username root. <p>Jawaban:</p> <pre>.env 1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:AJwkYP+sMDNF7rwISut9BLKyK+jncupi0S4Mgq1lHbs= 4 APP_DEBUG=true 5 APP_URL=http://localhost 6 7 LOG_CHANNEL=stack 8 9 DB_CONNECTION=mysql 10 DB_HOST=localhost 11 DB_PORT=3306 12 DB_DATABASE=latihan_laravel 13 DB_USERNAME=root 14 DB_PASSWORD=</pre>
3	<p>Jalankan xampp, selanjutnya buat tabel dengan nama mahasiswa di mysql pada database latihan_laravel.</p> <p>Jawaban:</p> 

4	<p>Isilah beberapa data pada tabel mahasiswa tersebut.</p> <p>Jawaban:</p> 
---	---

b. Menampilkan Data dari Database

Langkah	Keterangan
1	<p>Setelah kita memiliki beberapa data pada tabel mahasiswa, kita akan mencoba untuk menampilkan data tersebut ketika project dijalankan.</p> <p>Pertama, buatlah <i>route</i> pada routes/web.php sehingga ketika pertama kali project dijalankan akan terbuka halaman yang menampilkan data.</p> <p>Keterangan:</p> <p>Ketika route ('/') diakses, akan dijalankan method index pada controller bernama MahasiswaController.</p> <p>Jawaban:</p> 
2	<p>Buat controller baru menggunakan command prompt yaitu MahasiswaController menggunakan php artisan</p> <p>cd laravel-crud</p> <p>php artisan make:controller MahasiswaController</p> <p>Jawaban:</p> 
3	<p>Buat method index pada MahasiswaController.php pada folder app/Http/Controllers</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Tambahkan 'use Illuminate\Support\Facades\DB;' (line 6) agar <i>query builder</i> dapat digunakan - Line 13 untuk mengambil data dari tabel mahasiswa menggunakan <i>query builder</i> laravel dan akan disimpan di variabel \$mahasiswa - Line 16 : data akan dikirim ke blade view bernama index <p>Jawaban:</p>

```

app > Http > Controllers > MahasiswaController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class MahasiswaController extends Controller
9  {
10     public function index()
11     {
12         //mengambil data dari tabel mahasiswa
13         $mahasiswa = DB::table('mahasiswa')->get();
14
15         //mengirim data mahasiswa ke view index
16         return view('index',['mahasiswa' => $mahasiswa]);
17     }
18 }

```

4

Selanjutnya kita akan membuat view untuk menampilkan data mahasiswa dengan nama index.blade.php. Tetapi agar pembuatan view selanjutnya menjadi lebih mudah, terlebih dahulu kita akan membuat **template blade** (seperti file *header-footer* pada pembahasan CI) dengan nama **master.blade.php** pada folder **resources/views**.

Keterangan:

- Fungsi @yield pada line 6(title), 15(judul_halaman), dan 19(konten) berfungsi sebagai penanda bagian-bagian pada master blade. Nantinya bagian @yield ini akan diisi sesuai dengan halaman view yang menerapkan master.blade.php

Jawaban:

```

resources > views > master.blade.php
1  <html>
2  <head>
3      <meta charset="UTF-8">
4      <meta name="viewport" content="width=device-width, initial-scale=1.0">
5      <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
6      <title> @yield('title') </title>
7  </head>
8  <body>
9      <div class="container">
10         <div class="row mt-3">
11             <div class="col-md-6">
12                 <div class="card">
13                     <div class="card-header text-center">
14                         <!-- bagian judul halaman -->
15                         <h2> @yield('judul_halaman') </h2>
16                     </div>
17                     <div class="card-body">
18                         <!-- bagian konten blog -->
19                         @yield('konten')
20                     </div>
21                 </div>
22             <!-- card -->
23         </div>
24     </div>
25 </div>
26 <!-- container -->
27 </body>
28 </html>

```

5

Sekarang buat file **index.blade.php** yang menerapkan *template* master.blade.php dan akan digunakan untuk menampilkan data.

Keterangan:

- Line 1 : @extends menunjukkan bahwa file index.blade.php menerapkan blade lain yaitu master.blade.php
- Line 4 : @section('title', 'Home') berarti bahwa file index mengisi @yield('title') pada master dengan 'Home'
- Line 7 : @section('judul_halaman', 'Data Mahasiswa) berarti bahwa file index mengisi @yield('judul_halaman) pada master dengan 'Home'
- Line 10-30 : mengisi yield(@konten), karena terdapat banyak baris pada diawali dengan @section('konten') dan diakhiri dengan @endsection
- Line 24-25 menampilkan data mahasiswa dengan kolom nama dan nim

Jawaban:

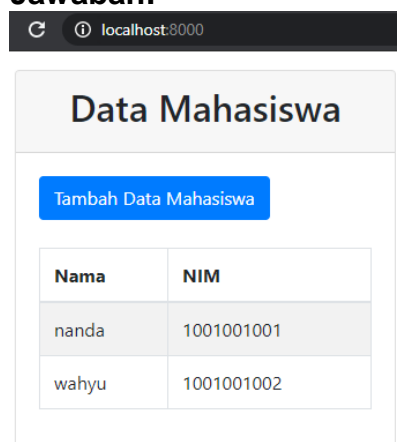
```
resources > views > index.blade.php
1  @extends('master')
2
3  <!-- isi title -->
4  @section('title', 'Home')
5
6  <!-- isi bagian judul halaman -->
7  @section('judul_halaman', 'Data Mahasiswa')
8
9  <!-- isi bagian konten -->
10 @section('konten')
11     <a href="/mahasiswa/tambah" class="btn btn-primary">Tambah Data Mahasiswa</a>
12     <br>
13     <br>
14     <table class="table table-bordered table-hover table-striped">
15         <thead>
16             <tr>
17                 <th>Nama</th>
18                 <th>NIM</th>
19             </tr>
20         </thead>
21         <tbody>
22             @foreach($mahasiswa as $mhs)
23                 <tr>
24                     <td>{{ $mhs->nama }}</td>
25                     <td>{{ $mhs->nim }}</td>
26                 </tr>
27             @endforeach
28         </tbody>
29     </table>
30 @endsection
```

6

Jalankan command prompt, tuliskan perintah untuk menjalankan project laravel-crud **php artisan serve**

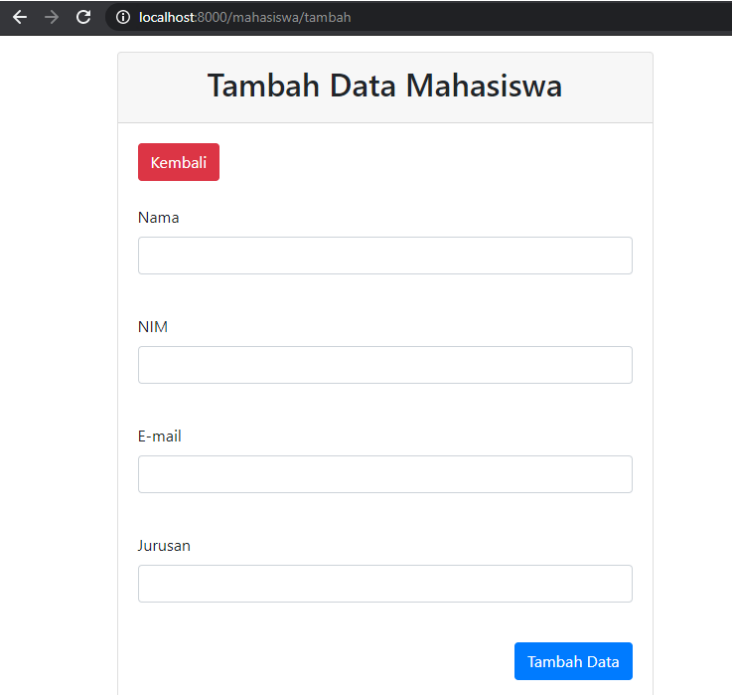
Buka browser dan ketikkan localhost:8000, maka akan tampil sebagai berikut

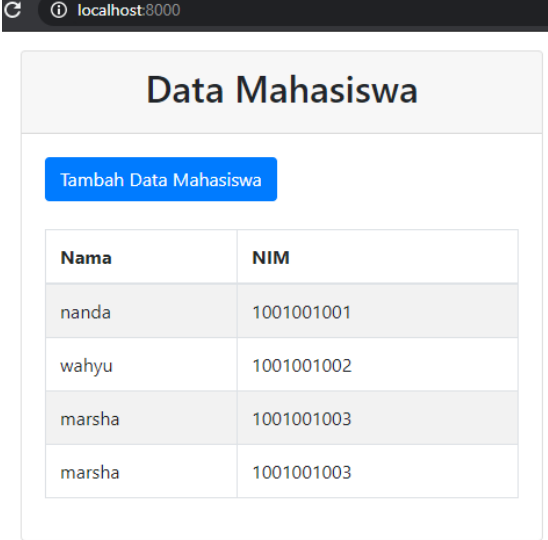
Jawaban:



c. Memasukkan Data (Create) ke Database

Langka h	Keterangan
1	<p>Buatlah <i>route</i> baru pada routes/web.php dengan nama /mahasiswa/tambah yang akan menjalankan fungsi tambah pada MahasiswaController</p> <p>Jawaban:</p> <pre> 20 Route::get('/', 'MahasiswaController@index'); 21 22 Route::get('/mahasiswa/tambah', 'MahasiswaController@tambah');</pre>
2	<p>Buat method tambah pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan view tambah.</p> <p>Jawaban:</p> <pre> 15 //mengirim data mahasiswa 16 return view('index', ['mahasiswa' => \$mahasiswa]); 17 } 18 19 public function tambah() 20 { 21 //memanggil view tambah 22 return view('tambah'); 23 }</pre>
3	<p>Kemudian buatlah view tambah.blade.php yang berisi form untuk memasukkan data baru pada folder resources/views. View tambah juga mengaplikasikan master.blade.php.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Line 13-32 merupakan form untuk memasukkan data mahasiswa berupa nama, nim, email, dan jurusan - Line 13 terdapat action="/mahasiswa/simpan" yang menunjukkan routes /mahasiswa/simpan dimana data pada form tersebut akan dikirimkan ke fungsi simpan pada controller MahasiswaController - Line 14 terdapat {{ csrf_field() }} yang digunakan untuk menerapkan fitur laravel yaitu csrf protection. csrf protection adalah fitur keamanan untuk mencegah penginputan dari luar aplikasi, csrf akan men-generate kode token otomatis yang dibuat dalam bentuk <i>form hidden</i>. <p>Jawaban:</p> <pre> resources > views > tambah.blade.php 1 @extends('master') 2 3 <!-- isi title --> 4 @section('title', 'Tambah Data') 5 6 <!-- isi bagian judul halaman --> 7 @section('judul_halaman', 'Tambah Data Mahasiswa') 8 9 <!-- isi bagian konten --> 10 @section('konten') 11 Kembali 12
 13
 14 <form action="/mahasiswa/simpan" method="post"> 15 {{ csrf_field() }} 16 <div class="form-group"> 17 <label for="namamhs">Nama</label> 18 <input type="text" class="form-control" required="required" name="namamhs">
 19 </div> 20 <div class="form-group"> 21 <label for="nimsh">NIM</label> 22 <input type="text" class="form-control" required="required" name="nimsh">
 23 </div> 24 <div class="form-group"> 25 <label for="emailmhs">E-mail</label> 26 <input type="email" class="form-control" required="required" name="emailmhs">
 27 </div> 28 <div class="form-group"> 29 <label for="jurusanmhs">Jurusan</label> 30 <input type="text" class="form-control" required="required" name="jurusanmhs">
 31 </div> 32 <button type="submit" name="tambah" class="btn btn-primary float-right">Tambah Data</button> 33 </form> 34 @endsection</pre>

4	<p>Ketika tombol simpan ditekan, akan dipanggil routes /mahasiswa/simpan. Oleh karena itu, kita buat terlebih dahulu route tersebut.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Pada route ini menggunakan metode post karena data mahasiswa dari form akan dikirim ke method simpan di MahasiswaController.php <p>Jawaban:</p> <pre>22 Route::get('/mahasiswa/tambah','MahasiswaController@tambah'); 23 24 Route::post('/mahasiswa/simpan','MahasiswaController@simpan');</pre>
5	<p>Buat method simpan pada MahasiswaController untuk menyimpan data ke database.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Variabel \$request untuk menerima data yang akan ditambahkan ke database - Line 28-33 merupakan query builder untuk insert data ke tabel mahasiswa <p>Jawaban:</p> <pre>25 public function simpan(Request \$request) 26 { 27 // insert data ke table mahasiswa 28 DB::table('mahasiswa')->insert([29 'nama' => \$request->namamsh, 30 'nim' => \$request->nimmhs, 31 'email' => \$request->emailmhs, 32 'jurusan' => \$request->jurusanmhs 33]); 34 return redirect('/mahasiswa'); 35 }</pre>
6	<p>Kembali coba jalankan localhost:8000 dan pilih tombol 'Tambah Data Mahasiswa', maka akan ditampilkan halaman tambah yang berisi form untuk memasukkan data baru. Kita coba isikan data pada form tersebut.</p> <p>Jawaban:</p>  <p>Setelah kita klik tombol tambah data, maka data baru akan ditampilkan pada halaman index.</p> <p>Jawaban:</p>

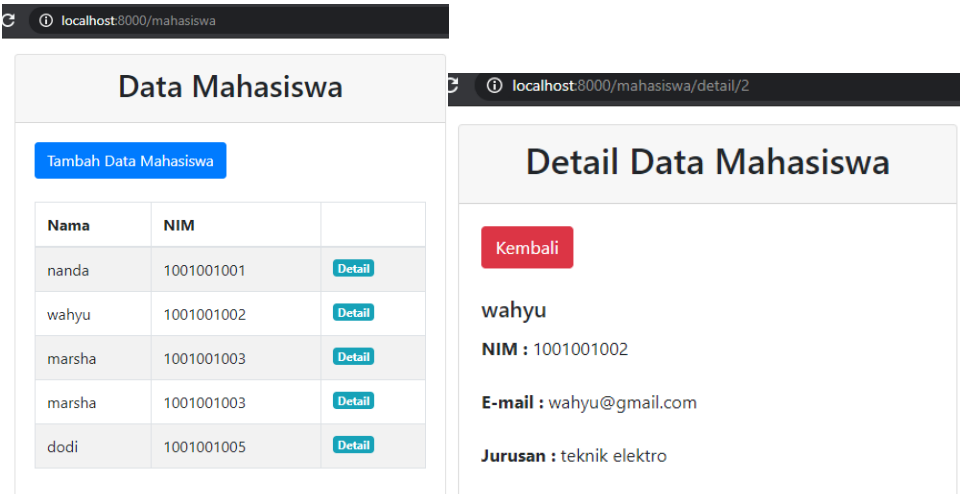


Kita dapat mencoba menambahkan beberapa data lagi agar jumlah data lebih banyak.

d. Melihat Detail Data dari Database

Langka h	Keterangan
1	<p>Buatlah tombol untuk melihat detail data (Line 28) pada file index.blade.php di folder resources/views</p> <p>Jawaban:</p> <pre> <thead> <tr> <th>Nama</th> <th>NIM</th> </tr> </thead> <tbody> @foreach(\$mahasiswa as \$mhs) <tr> <td>{{ \$mhs->nama }}</td> <td>{{ \$mhs->nim }}</td> <td> id }}" class="badge badge-info"> </td> </tr> @endforeach </tbody> </pre>
2	<p>Buatlah <i>route</i> baru pada routes/web.php dengan nama /mahasiswa/detail yang akan menjalankan fungsi detail pada MahasiswaController</p> <p>Jawaban:</p> <pre> 25 26 Route::get('/mahasiswa/detail/{id}', 'MahasiswaController@detail'); </pre>
3	<p>Buat method detail pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan data mahasiswa pada view detail.blade.php.</p> <p>Jawaban:</p> <pre> 37 public function detail(\$id) 38 { 39 //mengambil data mahasiswa berdasarkan id yang dipilih 40 \$mahasiswa = DB::table('mahasiswa')->where('id',\$id)->get(); 41 //kirim data mahasiswa yang diambil ke view edit.blade.php 42 return view('detail',['mahasiswa' => \$mahasiswa]); 43 } </pre>
4	<p>Kemudian buatlah view detail.blade.php yang menampilkan detail data mahasiswa pada folder resources/views. View detail juga mengaplikasikan master.blade.php.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Line 16-27 digunakan untuk menampilkan data mahasiswa berupa nama,

	<p>nim, email, dan jurusan</p> <p>Jawaban:</p> <pre>resources > views > detail.blade.php 1 @extends('master') 2 3 <!-- isi title --> 4 @section('title', 'Detail Mahasiswa') 5 6 <!-- isi bagian judul halaman --> 7 @section('judul_halaman', 'Detail Data Mahasiswa') 8 9 <!-- isi bagian konten --> 10 @section('konten') 11 Kembali 12
 13
 14 @foreach(\$mahasiswa as \$mhs) 15 <h5 class="card-title"> {{ \$mhs->nama }} </h5> 16 <p class="card-text"> 17 <label for=""> NIM : </label> 18 {{ \$mhs->nim }} 19 </p> 20 <p class="card-text"> 21 <label for=""> E-mail : </label> 22 {{ \$mhs->email }} 23 </p> 24 <p class="card-text"> 25 <label for=""> Jurusan : </label> 26 {{ \$mhs->jurusan }} 27 </p> 28 @endforeach 29 @endsection</pre>
--	---

5	<p>Jalankan localhost:8000 dan pilih tombol ‘Detail’ di suatu data yang ingin kita lihat detailnya.</p> <p>Jawaban:</p> 
---	--

e. Mengubah Data (Update) dari Database

Langkah	Keterangan
1	<p>Buatlah tombol untuk mengubah data (Line 29) pada file index.blade.php di folder resource/views</p> <p>Jawaban:</p> <pre>24 <tr> 25 <td>{{ \$mhs->nama }}</td> 26 <td>{{ \$mhs-> nim }}</td> 27 <td> 28 id }}" class="badge badge-info">Detail 29 id }}" class="badge badge-warning">Edit 30 </td> 31 </tr></pre>
2	<p>Buatlah <i>route</i> baru pada routes/web.php dengan nama /mahasiswa/edit yang akan</p>

	<p>menjalankan fungsi edit pada MahasiswaController</p> <p>Jawaban:</p> <pre>27 28 Route::get('/mahasiswa/edit/{id}', 'MahasiswaController@edit');</pre>
3	<p>Buat method edit pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan view edit.</p> <p>Keterangan :</p> <ul style="list-style-type: none"> - Line 49 : query builder untuk mengambil data mahasiswa berdasarkan id yang dipilih <p>Jawaban:</p> <pre>45 public function edit(\$id) 46 { 47 //mengambil data mahasiswa berdasarkan id yang dipilih 48 \$mahasiswa = DB::table('mahasiswa')->where('id',\$id)->get(); 49 //kirim data mahasiswa yang diambil ke view edit.blade.php 50 return view('edit',['mahasiswa' => \$mahasiswa]); 51 }</pre>
4	<p>Kemudian buatlah view edit.blade.php yang berisi form untuk mengubah data pada folder resources/views. View edit juga mengaplikasikan master.blade.php.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Line 14-36 merupakan form untuk memasukkan data mahasiswa berupa nama, nim, email, dan jurusan - Line 15 terdapat action="/mahasiswa/update" yang menunjukkan routes /mahasiswa/update dimana data pada form tersebut akan dikirimkan ke fungsi update pada controller MahasiswaController <p>Jawaban:</p> <pre>resources > views > edit.blade.php 1 @extends('master') 2 3 <!-- isi title --> 4 @section('title', 'Edit Data') 5 6 <!-- isi bagian judul halaman --> 7 @section('judul_halaman', 'Edit Data Mahasiswa') 8 9 <!-- isi bagian konten --> 10 @section('konten') 11 Kembali 12
 13
 14 @foreach(\$mahasiswa as \$mhs) 15 <form action="/mahasiswa/update" method="post"> 16 {{ csrf_field() }} 17 <input type="hidden" name="id" value="{{ \$mhs->id }}">
 18 <div class="form-group"> 19 <label for="namamhs">Nama</label> 20 <input type="text" class="form-control" required="required" name="namamhs" value="{{ \$mhs->nama }}">
 21 </div> 22 <div class="form-group"> 23 <label for="nimhs">NIM</label> 24 <input type="text" class="form-control" required="required" name="nimhs" value="{{ \$mhs->nim }}">
 25 </div> 26 <div class="form-group"> 27 <label for="emailhs">E-mail</label> 28 <input type="email" class="form-control" required="required" name="emailhs" value="{{ \$mhs->email }}">
 29 </div> 30 <div class="form-group"> 31 <label for="jurusanhs">Jurusan</label> 32 <input type="text" class="form-control" required="required" name="jurusanhs" value="{{ \$mhs->jurusan }}">
 33 </div> 34 </form> 35 @endforeach 36 @endsection</pre>
5	<p>Ketika tombol simpan ditekan, akan dipanggil routes /mahasiswa/update. Oleh karena itu, kita buat terlebih dahulu route tersebut.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Pada route ini menggunakan metode post karena data mahasiswa dari form akan dikirim ke method update di MahasiswaController.php <p>Jawaban:</p> <pre>29 30 Route::post('/mahasiswa/update', 'MahasiswaController@update');</pre>
6	<p>Buat method update pada MahasiswaController untuk menyimpan data yang diubah ke database.</p>

Keterangan:

- Variabel \$request untuk menerima data yang akan ditambahkan ke database
- Line 58-63 merupakan query builder untuk update data ke tabel mahasiswa

Jawaban:

```
53 public function update(Request $request)
54 {
55     //update data mahasiswa
56     DB::table('mahasiswa')->where('id',$request->id)->update([
57         'nama' => $request->namamhs,
58         'nim' => $request->nimmhs,
59         'email' => $request->emailmhs,
60         'jurusan' => $request->jurumasmhs
61     ]);
62     return redirect('/mahasiswa');
63 }
```

7

Jalankan localhost:8000 dan pilih tombol 'Edit', maka akan ditampilkan halaman edit yang berisi form untuk mengubah data baru.

Jawaban:

localhost:8000/mahasiswa

Data Mahasiswa

Tambah Data Mahasiswa

Nama	NIM	
nanda	1001001001	Detail Edit
wahyu	1001001002	Detail Edit
marsha	1001001003	Detail Edit
marsha	1001001003	Detail Edit
dodi	1001001005	Detail Edit

Klik edit di data keempat, kita akan mengubah namanya.

Jawaban:

localhost:8000/mahasiswa/edit/4

Edit Data Mahasiswa

Kembali

Nama

vendi

NIM

1001001004

E-mail

vendi@gmail.com

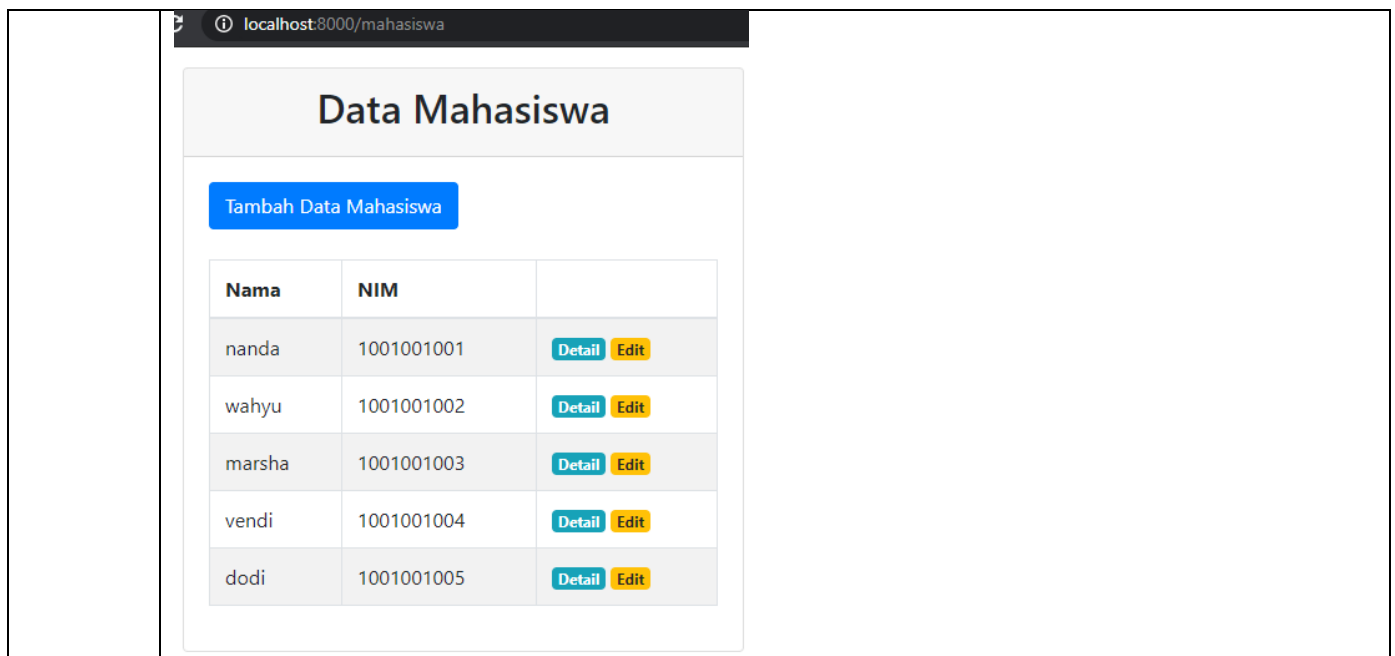
Jurusan

Teknik Mesin

Simpan Data

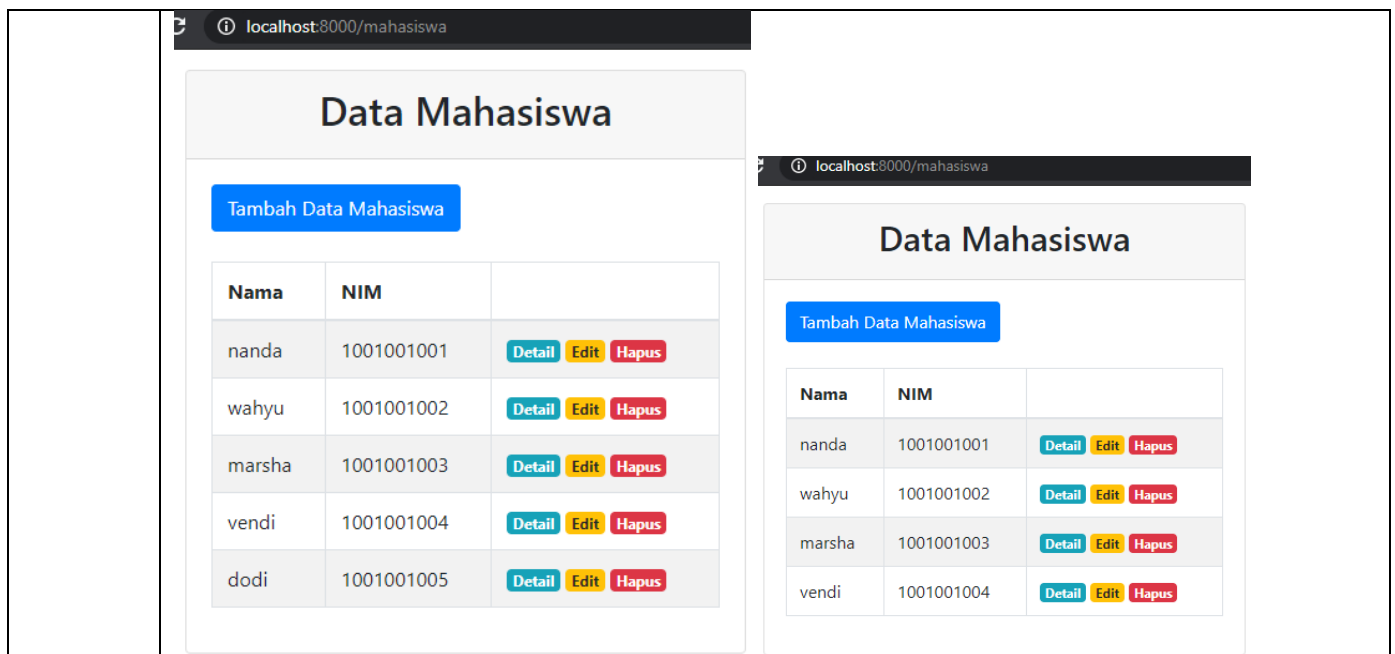
Setelah kita klik simpan Data, maka data pertama akan berubah.

Jawaban:



f. Menghapus Data (Delete) dari Database

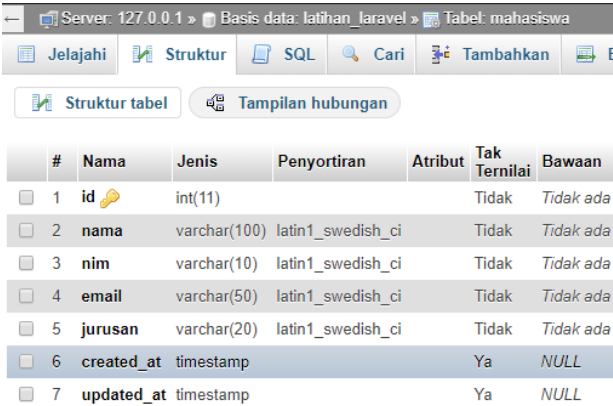
Langka h	Keterangan
1	<p>Buatlah tombol untuk menghapus data (Line 30) pada file index.blade.php di folder resources/views</p> <p>Jawaban:</p> <pre> 24 <tr> 25 <td>{{ \$mhs->nama }}</td> 26 <td>{{ \$mhs->nim }}</td> 27 <td> 28 id }}" class="badge badge-info">Detail 29 id }}" class="badge badge-warning">Edit 30 id }}" class="badge badge-danger">Hapus 31 </td> 32 </tr> </pre>
2	<p>Buatlah <i>route</i> baru pada routes/web.php dengan nama /mahasiswa/hapus yang akan menjalankan fungsi hapus pada MahasiswaController</p> <p>Jawaban:</p> <pre> 31 32 Route::get('/mahasiswa/hapus/{id}', 'MahasiswaController@hapus'); </pre>
3	<p>Buat method hapus pada MahasiswaController.php di folder app/Http/Controllers yang akan menjalankan fungsi hapus.</p> <p>Keterangan :</p> <ul style="list-style-type: none"> - Line 67 : query builder untuk menghapus data mahasiswa berdasarkan id yang dipilih <p>Jawaban:</p> <pre> 65 public function hapus(\$id) 66 { 67 //menghapus data mahasiswa berdasarkan id yang dipilih 68 DB::table('mahasiswa')->where('id',\$id)->delete(); 69 70 return redirect('/mahasiswa'); 71 } </pre>
4	<p>Jalankan localhost:8000 dan pilih tombol 'Hapus', maka data yang terpilih akan dihapus. Contoh: menghapus data terakhir.</p> <p>Jawaban:</p>



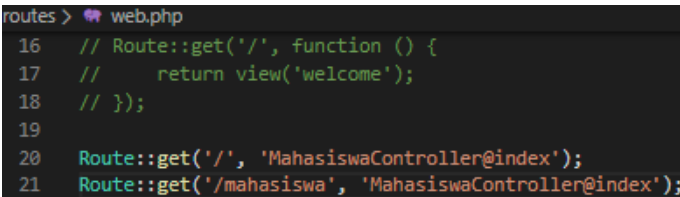
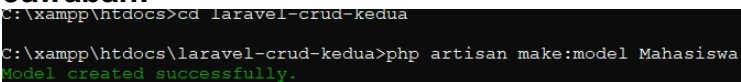
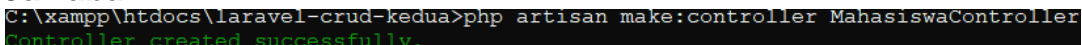
Praktikum – Bagian 2: Membuat CRUD di Laravel menggunakan Eloquent

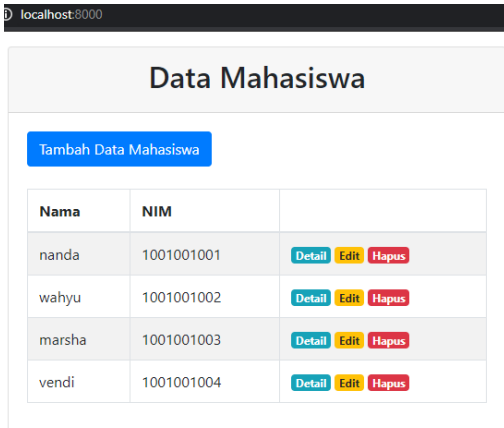
a. Konfigurasi Database

Langka h	Keterangan
1	<p>Buatlah project Laravel baru dengan nama laravel-crud-kedua. Buka command prompt, tuliskan perintah berikut.</p> <pre>cd C:\xampp\htdocs laravel new laravel-crud-kedua</pre> <p>Jawaban:</p> <pre>C:\xampp\htdocs>cd C:\xampp\htdocs C:\xampp\htdocs>laravel new laravel-crud-kedua Crafting application... Loading composer repositories with package information Installing dependencies (including require-dev) from lock file Package operations: 92 installs, 0 updates, 0 removals - Installing doctrine/inflector (1.3.1): Loading from cache - Installing doctrine/lexer (1.2.0): Loading from cache - Installing dragonmantank/cron-expression (v2.3.0): Loading from cache - Installing voku/portable-ascii (1.4.10): Loading from cache - Installing symfony/polyfill-ctype (v1.15.0): Loading from cache 15 packages you are using are looking for funding. Use the composer fund command to find out more! > @php -r "file_exists('.env') copy('.env.example', '.env');" > @php artisan key:generate --ansi Application key set successfully. > Illuminate\Foundation\ComposerScripts::postAutoloadDump > @php artisan package:discover --ansi Discovered Package: facade/ignition Discovered Package: fideller/proxy Discovered Package: fruitcake/laravel-core Discovered Package: laravel/tinker Discovered Package: nesbot/carbon Discovered Package: nunomaduro/collision Package manifest generated successfully. Application ready! Build something amazing. C:\xampp\htdocs></pre>
2	<p>Selanjutnya kita lakukan konfigurasi database di Laravel. Untuk melakukan konfigurasi database, bukalah file .env pada project laravel-crud. Ubah seperti di bawah ini.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Nama database yang akan digunakan adalah latihan_laravel dengan username root. <p>Jawaban:</p> <pre>.env 1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:U9IsBCFr6+N42ESoMJmUY02AxUoR7ECIsd4Bn0bHzik= 4 APP_DEBUG=true 5 APP_URL=http://localhost 6 7 LOG_CHANNEL=stack 8 9 DB_CONNECTION=mysql 10 DB_HOST=localhost 11 DB_PORT=3306 12 DB_DATABASE=latihan_laravel 13 DB_USERNAME=root 14 DB_PASSWORD=</pre>

3	<p>Pada project ini kita gunakan database dan tabel yang sebelumnya digunakan pada Praktikum Bagian 1. Tambahkan kolom <code>created_at</code> dan <code>updated_at</code> yang bertipe <code>TIMESTAMP</code> dan default nilainya <code>NULL</code></p> <p>Jawaban:</p> 
---	--

b. Menampilkan Data dari Database

Langkah	Keterangan
1	<p>Setelah kita memiliki beberapa data pada tabel mahasiswa, kita akan mencoba untuk menampilkan data tersebut ketika project dijalankan. Pertama, buatlah <i>route</i> pada routes/web.php sehingga ketika pertama kali project dijalankan akan terbuka halaman yang menampilkan data.</p> <p>Jawaban:</p> 
2	<p>Buat model menggunakan command prompt dengan nama Mahasiswa menggunakan php artisan</p> <p>cd laravel-crud-kedua php artisan make:model Mahasiswa</p> <p>Jawaban:</p> 
3	<p>Ubah model Mahasiswa.php pada folder App menjadi seperti berikut.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Model Mahasiswa akan menangani tabel mahasiswa <p>Jawaban:</p> 
4	<p>Buat controller baru yaitu MahasiswaController menggunakan php artisan</p> <p>php artisan make:controller MahasiswaController</p> <p>Jawaban:</p> 

5	<p>Buat method index pada MahasiswaController.php pada folder app/Http/Controllers</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Tambahkan 'use App\Mahasiswa' (line 6) untuk menggunakan model Mahasiswa - Line 12 untuk mengambil semua data dari model/tabel Mahasiswa dan akan disimpan di variabel \$mahasiswa - Line 16 : data akan dikirim ke blade view bernama index <p>Jawaban:</p> <pre> app > Http > Controllers > MahasiswaController.php 1 <?php 2 3 namespace App\Http\Controllers; 4 5 use Illuminate\Http\Request; 6 use App\Mahasiswa; 7 8 class MahasiswaController extends Controller 9 { 10 public function index() 11 { 12 \$mahasiswa = Mahasiswa::all(); 13 return view('index', ['mahasiswa' => \$mahasiswa]); 14 } 15 } </pre>
6	<p>Selanjutnya kita akan membuat view untuk menampilkan data mahasiswa dengan nama index.blade.php. Tetapi agar pembuatan view selanjutnya menjadi lebih mudah, terlebih dahulu kita akan membuat template blade (seperti pada Bagian 1). Pada bagian view kita buat seperti bagian 1 (copy-paste dari bagian 1)</p>
7	<p>Jalankan command prompt, tuliskan perintah untuk menjalankan project laravel-crud php artisan serve</p> <p>Buka browser dan ketikkan localhost:8000, maka akan tampil sebagai berikut</p> <p>Jawaban:</p> 

c. Memasukkan Data (Create) ke Database

Langka h	Keterangan
1	<p>Buatlah <i>route</i> baru pada routes/web.php dengan nama /mahasiswa/tambah yang akan menjalankan fungsi tambah pada MahasiswaController ketika tombol Tambah Data Mahasiswa ditekan.</p> <p>Jawaban:</p> <pre> 21 Route::get('/mahasiswa', 'MahasiswaController@index'); 22 23 Route::get('/mahasiswa/tambah', 'MahasiswaController@tambah'); </pre>
2	<p>Buat method tambah pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan view tambah.</p> <p>Jawaban:</p> <pre> 16 public function tambah() 17 { 18 return view('tambah'); 19 } </pre>

3	Kemudian buatlah view tambah.blade.php yang berisi form untuk memasukkan data baru pada folder resources/views . Kita lakukan copy paste dari tambah.blade.php di Bagian 1
4	<p>Ketika tombol simpan ditekan, akan dipanggil routes /mahasiswa/simpan. Oleh karena itu, kita buat terlebih dahulu route tersebut.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Pada route ini menggunakan metode post karena data mahasiswa dari form akan dikirim ke method simpan di MahasiswaController.php <p>Jawaban:</p> <pre> 24 25 Route::post('/mahasiswa/simpan', 'MahasiswaController@simpan');</pre>
5	<p>Buat method simpan pada MahasiswaController untuk menyimpan data ke database.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Variabel \$request untuk menerima data yang akan ditambahkan ke database - Line 23-28 merupakan fitur eloquent menggunakan fungsi create() untuk insert data ke tabel mahasiswa <p>Jawaban:</p> <pre> 21 public function simpan(Request \$request) 22 { 23 Mahasiswa::create([24 'nama' => \$request->namamhs, 25 'nim' => \$request->nimmhs, 26 'email' => \$request->emailmhs, 27 'jurusan' => \$request->jurusanmhs 28]); 29 30 return redirect('/mahasiswa'); 31 }</pre>
6	<p>Karena kita menggunakan fungsi create pada Controller, maka butuh ditambahkan code pada Line 10 yang disebut Mass Assignment pada model Mahasiswa.php. Mass Assignment digunakan untuk memfilter kolom mana yang boleh dan tidak boleh diinput.</p> <p>Jawaban:</p> <pre> 7 class Mahasiswa extends Model 8 { 9 protected \$table = "mahasiswa"; 10 protected \$fillable = ['nama','nim','email','jurusan']; 11 }</pre>
7	<p>Kembali coba jalankan localhost:8000 dan pilih tombol 'Tambah Data Mahasiswa', maka akan ditampilkan halaman tambah yang berisi form untuk memasukkan data baru. Kita coba isikan data pada form tersebut.</p> <p>Jawaban:</p>

Tambah Data Mahasiswa

[Kembali](#)

Nama

NIM

E-mail

Jurusan

[Tambah Data](#)

Setelah kita klik tombol tambah data, maka hasilnya sebagai berikut.

Jawaban:

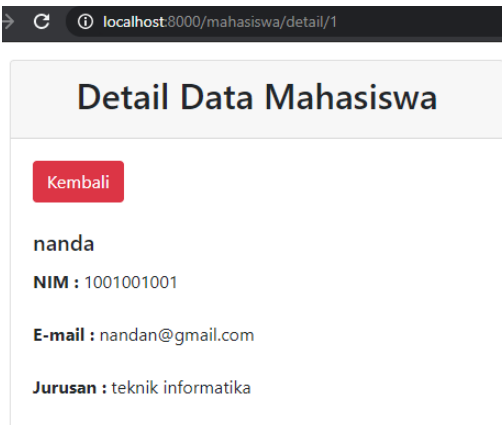
Data Mahasiswa

[Tambah Data Mahasiswa](#)

Nama	NIM	
nanda	1001001001	Detail Edit Hapus
wahyu	1001001002	Detail Edit Hapus
marsha	1001001003	Detail Edit Hapus
vendi	1001001004	Detail Edit Hapus
ijah	1001001006	Detail Edit Hapus

d. Melihat Detail Data dari Database

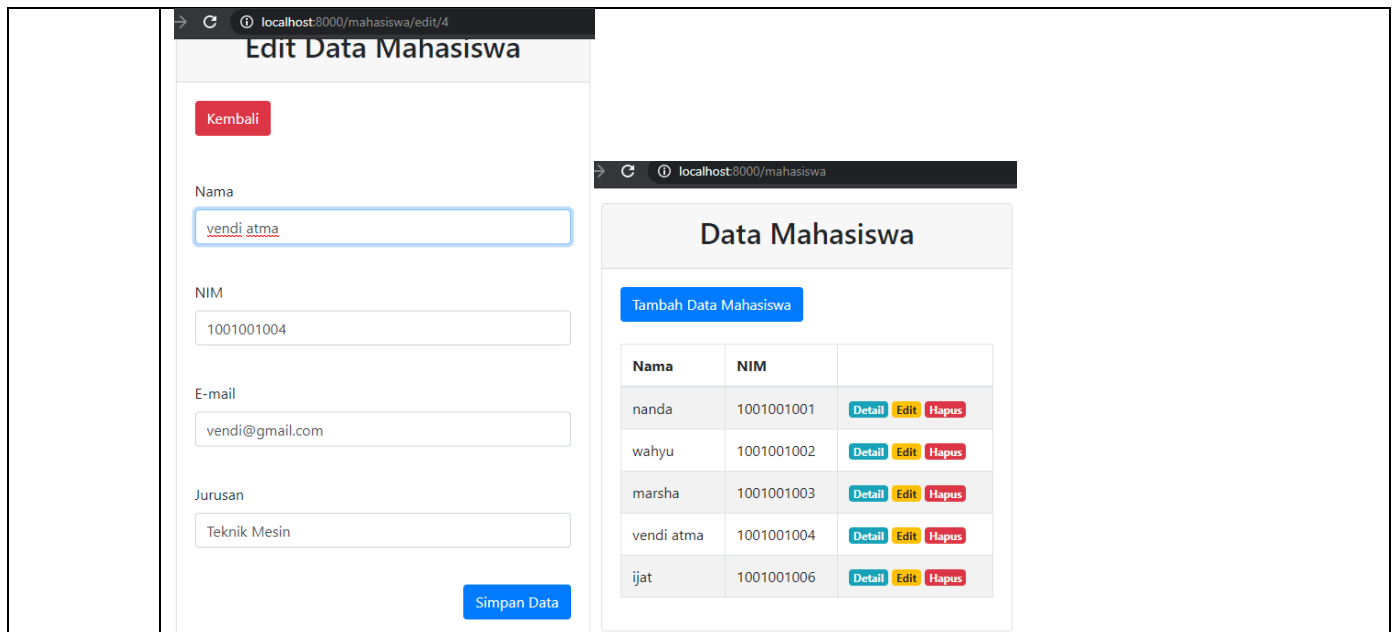
Langkah	Keterangan
1	<p>Buatlah <i>route</i> baru pada routes/web.php dengan nama /mahasiswa/detail yang akan menjalankan fungsi detail pada MahasiswaController</p> <p>Jawaban:</p> <pre>26 27 Route::get('/mahasiswa/detail/{id}', 'MahasiswaController@detail/{id}');</pre>
3	<p>Buat method detail pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan data mahasiswa pada view detail.blade.php.</p> <p>Jawaban:</p> <pre>33 public function detail(\$id) 34 { 35 \$mahasiswa = Mahasiswa::find(\$id); 36 return view('detail', ['mahasiswa' => \$mahasiswa]); 37 }</pre>

4	<p>Kemudian buatlah view detail.blade.php yang menampilkan detail data mahasiswa pada folder resources/views. View detail juga mengaplikasikan master.blade.php.</p> <p>Jawaban:</p> <pre>resources > views > detail.blade.php 1 @extends('master') 2 3 <!-- isi title --> 4 @section('title', 'Detail Mahasiswa') 5 6 <!-- isi bagian judul halaman --> 7 @section('judul_halaman', 'Detail Data Mahasiswa') 8 9 <!-- isi bagian konten --> 10 @section('konten') 11 Kembali 12 13
 14
 15 16 <h5 class="card-title">{{ \$mahasiswa->nama }} </h5> 17 <p class="card-text"> 18 <label for=""> NIM : </label> 19 {{ \$mahasiswa->nim }} 20 </p> 21 <p class="card-text"> 22 <label for=""> E-mail : </label> 23 {{ \$mahasiswa->email }} 24 </p> 25 <p class="card-text"> 26 <label for=""> Jurusan : </label> 27 {{ \$mahasiswa->jurusan }} 28 </p> 29 @endsection</pre>
5	<p>Jalankan localhost:8000 dan pilih tombol ‘Detail’ di suatu data yang ingin kita lihat detailnya.</p> <p>Jawaban:</p> 

e. Mengubah Data (Update) dari Database

Langka h	Keterangan
1	<p>Buatlah <i>route</i> baru pada routes/web.php dengan nama /mahasiswa/edit yang akan menjalankan fungsi edit pada MahasiswaController</p> <p>Jawaban:</p> <pre>28 29 Route::get('/mahasiswa/edit/{id}', 'MahasiswaController@edit');</pre>
2	<p>Buat method edit pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan view edit.</p> <p>Keterangan :Line 41 : fungsi eloquent untuk mengambil data mahasiswa berdasarkan id yang dipilih</p> <p>Jawaban:</p> <pre>39 public function edit(\$id) 40 { 41 \$mahasiswa = Mahasiswa::find(\$id); 42 return view('edit', ['mahasiswa' => \$mahasiswa]); 43 }</pre>

3	<p>Kemudian buatlah view edit.blade.php yang berisi form untuk mengubah data pada folder resources/views. View edit juga mengaplikasikan master.blade.php.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Line 11-31 merupakan form untuk memasukkan data mahasiswa berupa nama, nim, email, dan jurusan - Line 11 terdapat action="/mahasiswa/update/{{ \$mahasiswa->id }}" yang menunjukkan routes /mahasiswa/update/{id} dimana data pada form tersebut akan dikirimkan ke fungsi update pada controller MahasiswaController <p>Jawaban:</p> <pre> resources > views > edit.blade.php 1 @extends('master') 2 3 <!-- isi title --> 4 @section('title', 'Edit Data') 5 6 <!-- isi bagian judul halaman --> 7 @section('judul_halaman', 'Edit Data Mahasiswa') 8 9 <!-- isi bagian konten --> 10 @section('konten') 11 Kembali 12
 13
 14 <form action="/mahasiswa/update/{{ \$mahasiswa->id }}" method="post"> 15 {{ csrf_field() }} 16 <input type="hidden" name="id" value="{{ \$mahasiswa->id }}">
 17 <div class="form-group"> 18 <label for="nama" >Nama</label> 19 <input type="text" class="form-control" required="required" name="nama" value="{{ \$mahasiswa->nama }}">
 20 </div> 21 <div class="form-group"> 22 <label for="nim" >NIM</label> 23 <input type="text" class="form-control" required="required" name="nim" value="{{ \$mahasiswa->nim }}">
 24 </div> 25 <div class="form-group"> 26 <label for="email" >E-mail</label> 27 <input type="text" class="form-control" required="required" name="email" value="{{ \$mahasiswa->email }}">
 28 </div> 29 <div class="form-group"> 30 <label for="jurusan" >Jurusan</label> 31 <input type="text" class="form-control" required="required" name="jurusan" value="{{ \$mahasiswa->jurusan }}">
 32 </div> 33 <button type="submit" name="edit" class="btn btn-primary float-right">Simpan Data</button> 34 </form> 35 @endsection </pre>
4	<p>Ketika tombol simpan ditekan, akan dipanggil routes /mahasiswa/update/{id}. Oleh karena itu, kita buat terlebih dahulu route tersebut.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Pada route ini menggunakan metode post karena data mahasiswa dari form akan dikirim ke method update di MahasiswaController.php <p>Jawaban:</p> <pre> 30 31 Route::post('/mahasiswa/update/{id}', 'MahasiswaController@update'); </pre>
5	<p>Buat method update pada MahasiswaController untuk menyimpan data yang diubah ke database.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> - Variabel \$request untuk menerima data yang akan ditambahkan ke database - Line 48-52 merupakan fungsi eloquent untuk update data ke tabel mahasiswa <p>Jawaban:</p> <pre> 45 public function update(\$id, Request \$request) 46 { 47 \$mahasiswa = Mahasiswa::find(\$id); 48 \$mahasiswa->nama = \$request->nama; 49 \$mahasiswa->nim = \$request->nim; 50 \$mahasiswa->email = \$request->email; 51 \$mahasiswa->jurusan = \$request->jurusan; 52 \$mahasiswa->save(); 53 return redirect('/mahasiswa'); 54 } </pre>
6	<p>Jalankan localhost:8000 dan pilih tombol 'Edit', maka akan ditampilkan halaman edit yang berisi form untuk mengubah data baru. Cobalah untuk mengedit data.</p> <p>Jawaban:</p>



f. Menghapus Data (Delete) dari Database

Langka h	Keterangan
1	<p>Buatlah <i>route</i> baru pada routes/web.php dengan nama /mahasiswa/hapus yang akan menjalankan fungsi hapus pada MahasiswaController</p> <p>Jawaban:</p> <pre>Route::get('/mahasiswa/hapus/{id}', 'MahasiswaController@hapus');</pre>
2	<p>Buat method hapus pada MahasiswaController.php di folder app/Http/Controllers yang akan menjalankan fungsi hapus.</p> <p>Keterangan :</p> <ul style="list-style-type: none"> - Line 59 :fungsi eloquent untuk menghapus data mahasiswa berdasarkan id yang dipilih <p>Jawaban:</p> <pre>public function hapus(\$id) { \$mahasiswa = Mahasiswa::find(\$id); \$mahasiswa->delete(); return redirect('/mahasiswa'); }</pre>
3	<p>Jalankan localhost:8000 dan pilih tombol 'Hapus', maka data yang terpilih akan dihapus.</p> <p>Jawaban:</p>