

LAPORAN PEMROGRAMAN WEB LANJUT

JOBSHEET 10

MEMBUAT CRUD MENGGUNAKAN LARAVEL

Oleh:
KINANTI PERMATA PUTRI NIM. 1841720022
TI-2A



PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

APRIL 2020

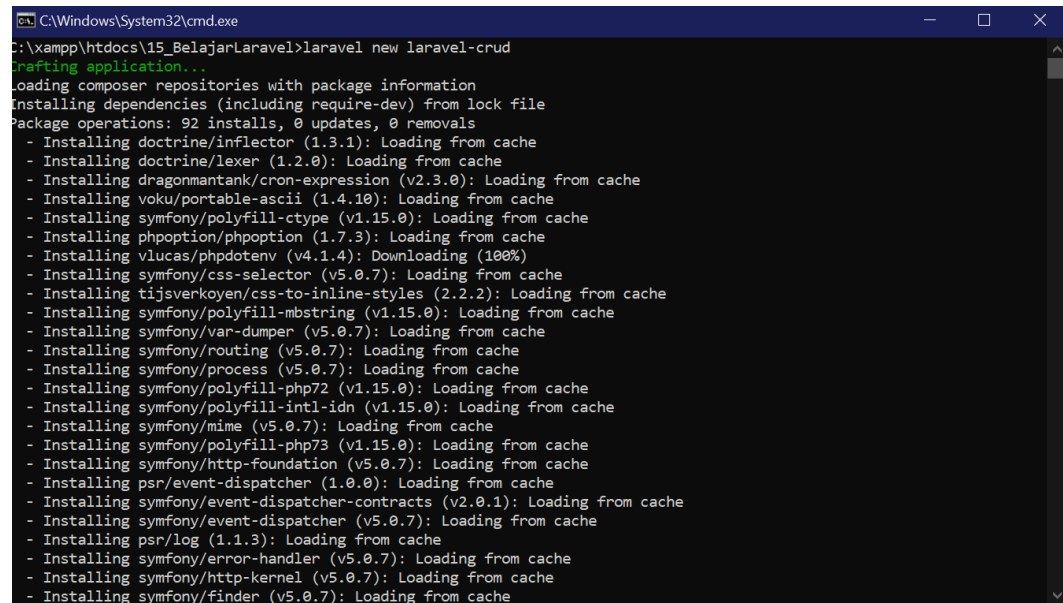
1. Praktikum – Bagian 1: Membuat CRUD di Laravel menggunakan Query Builder

a. Konfigurasi Database dan Pembuatan Tabel di MySQL

- 1) Buatlah project Laravel baru dengan nama laravel-crud. Buka command prompt, tuliskan perintah berikut.

```
cd C:\xampp\htdocs
```

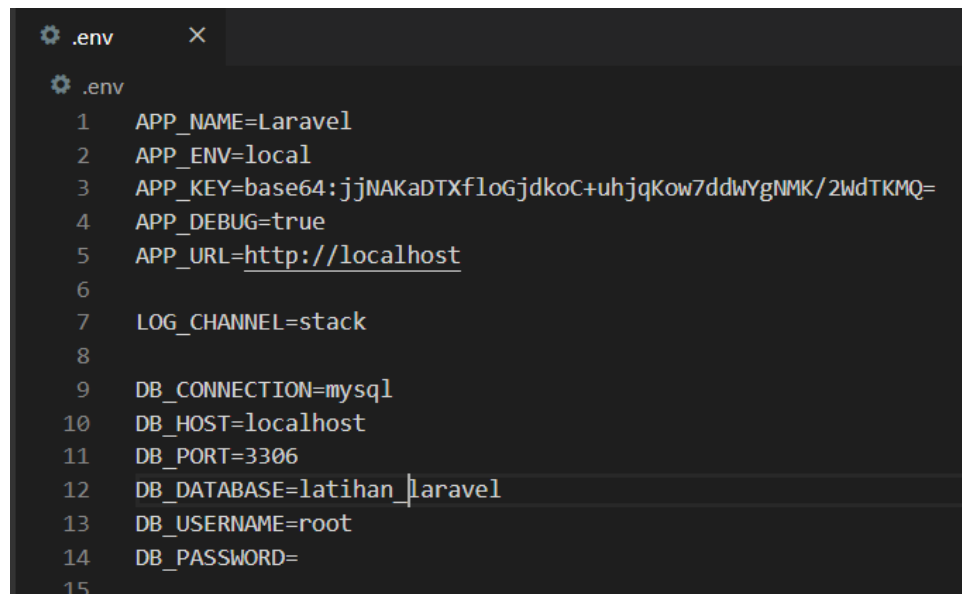
```
laravel new laravel-crud
```



```
C:\Windows\System32\cmd.exe
C:\xampp\htdocs\15_BelajarLaravel>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 from cache
- Installing voku/portable-ascii (1.4.10): Loading from cache
- Installing symfony/polyfill-ctype (v1.15.0): Loading from cache
- Installing phoption/phoption (1.7.3): Loading from cache
- Installing vlucas/phpdotenv (v4.1.4): Downloading (100%)
- Installing symfony/css-selector (v5.0.7): Loading from cache
- Installing tijsverkoyen/css-to-inline-styles (2.2.2): Loading from cache
- Installing symfony/polyfill-mbstring (v1.15.0): Loading from cache
- Installing symfony/var-dumper (v5.0.7): Loading from cache
- Installing symfony/routing (v5.0.7): Loading from cache
- Installing symfony/process (v5.0.7): Loading from cache
- Installing symfony/polyfill-php72 (v1.15.0): Loading from cache
- Installing symfony/polyfill-intl-idn (v1.15.0): Loading from cache
- Installing symfony/mime (v5.0.7): Loading from cache
- Installing symfony/polyfill-php73 (v1.15.0): Loading from cache
- Installing symfony/http-foundation (v5.0.7): Loading from cache
- Installing psr/event-dispatcher (1.0.0): Loading from cache
- Installing symfony/event-dispatcher-contracts (v2.0.1): Loading from cache
- Installing symfony/event-dispatcher (v5.0.7): Loading from cache
- Installing psr/log (1.1.3): Loading from cache
- Installing symfony/error-handler (v5.0.7): Loading from cache
- Installing symfony/http-kernel (v5.0.7): Loading from cache
- Installing symfony/finder (v5.0.7): Loading from cache
```

- 2) Selanjutnya kita lakukan konfigurasi database di Laravel. Untuk melakukan konfigurasi database, bukalah file .env pada project laravel-crud. Ubah seperti di bawah ini.

Nama database yang akan digunakan adalah latihan_laravel dengan username root.



```
.env
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:jjNAKaDTXflogjdkoC+uhjqKow7ddwYgNMK/2WdTKMQ=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=localhost
DB_PORT=3306
DB_DATABASE=latihan_laravel
DB_USERNAME=root
DB_PASSWORD=
```

- 3) Jalankan xampp, selanjutnya buat tabel dengan nama mahasiswa di mysql pada database latihan_laravel.

Server: 127.0.0.1 » Database: latihan_laravel » Table: mahasiswa

Table structure

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	nama	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
3	nim	varchar(10)	latin1_swedish_ci		No	None			Change Drop More
4	email	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
5	jurusan	varchar(20)	latin1_swedish_ci		No	None			Change Drop More

- 4) Isilah beberapa data pada tabel mahasiswa tersebut.

	id	nama	nim	email	jurusan
<input type="checkbox"/> Edit <input type="image"/> Copy <input type="image"/> Delete	1	nanda	1001001001	nandan@gmail.com	Teknik Informatika
<input type="checkbox"/> Edit <input type="image"/> Copy <input type="image"/> Delete	2	wahyu	1001001002	wahyu@gmail.com	Teknik Elektro

b. Menampilkan Data dari Database

- 1) Setelah kita memiliki beberapa data pada tabel mahasiswa, kita akan mencoba untuk menampilkan data tersebut ketika project dijalankan.

Pertama, buatlah route pada routes/web.php sehingga ketika pertama kali project dijalankan akan terbuka halaman yang menampilkan data.

```

routes > web.php
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4
5  /*
6  |-----
7  | Web Routes
8  |-----
9  |
10 | Here is where you can register web routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | contains the "web" middleware group. Now create something great!
13 |
14 */
15
16 // Route::get('/', function () {
17 //     return view('welcome');
18 // });
19
20 Route::get('/', 'MahasiswaController@index');
```

Keterangan:

- Ketika route ('/') diakses, akan dijalankan method index pada controller bernama MahasiswaController.

- 2) Buat controller baru menggunakan command prompt yaitu MahasiswaController menggunakan php artisan

cd laravel-crud

php artisan make:controller MahasiswaController

```
C:\xampp\htdocs\15_BelajarLaravel\laravel-crud>php artisan make:controller MahasiswaController
Controller created successfully.
C:\xampp\htdocs\15_BelajarLaravel\laravel-crud>_
```

- 3) Buat method index pada MahasiswaController.php pada folder app/Http/Controllers

```
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 }
19
```

Keterangan:

- Tambahkan 'use Illuminate\Support\Facades\DB;' (line 6) agar query builder dapat digunakan
 - Line 13 untuk mengambil data dari tabel mahasiswa menggunakan query builder laravel dan akan disimpan di variabel \$mahasiswa
 - Line 16 : data akan dikirim ke blade view bernama index
- 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.

```

resources > views > master.blade.php > html > head
1 <html>
2 <head>
3   <meta charset="utf-8">
4   <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
5   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.2.1/css/bootstrap.min.css" rel="stylesheet">
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       </div>
23     </div>
24   </div>
25 </body>
26 </html>

```

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

```

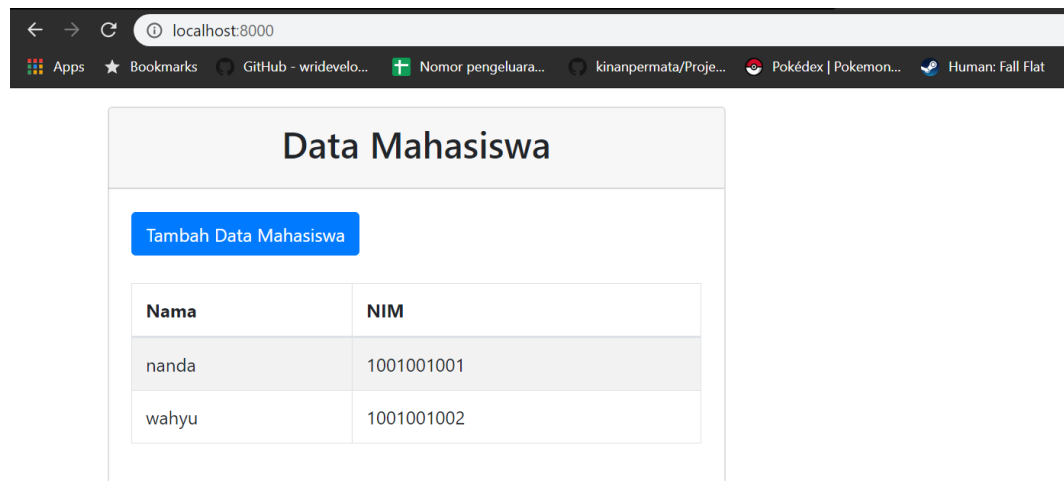
resources > views > index.blade.php > a.btn.btn-primary
1 @extends('master')
2
3 <!-- Isi title -->
4 @section('title', 'Home')
5
6 <!-- Isi bagian judul halaman-->
7 @section('judul_halaman', 'Data Mahasiswa')
8
9 <!-- Isi bagian konten-->
10 @section('konten')
11   <a href="/mahasiswa/tambah" class="btn 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

```

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
- 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



c. Memasukkan Data (Create) ke Database

- 1) Buatlah route baru pada `routes/web.php` dengan nama `/mahasiswa/tambah` yang akan menjalankan fungsi tambah pada `MahasiswaController`

```
21  
22 Route::get('/mahasiswa/tambah', 'MahasiswaController@tambah');
```

- 2) Buat method tambah pada `MahasiswaController.php` di folder `app/Http/Controllers` yang akan menampilkan view tambah.

```

19     public function tambah()
20     {
21         // Memanggil view tambah
22         return view('tambah');
23     }

```

- 3) Kemudian buatlah view tambah.blade.php yang berisi form untuk memasukkan data baru pada folder resources/views. View tambah juga mengaplikasikan master.blade.php.

```

resources > views > tambah.blade.php > form > div.form-group > input.form-control
1  @extends('master')
2  <!-- Isi title -->
3  @section('title', 'Tambah Data')
4
5  <!-- Isi bagian judul halaman-->
6  @section('judul_halaman', 'Tambah Data Mahasiswa')
7
8  <!-- Isi bagian konten-->
9  @section('konten')
10     <a href="/mahasiswa" class="btn btn-primary">Kembali</a>
11     <br/>
12     <br/>
13     <form action="/mahasiswa/simpan" method="post">
14         {{ csrf_field() }}
15         <div class="form-group">
16             <label for="namamhs">Nama</label>
17             <input type="text" class="form-control" required="required" name="namamhs"> <br/>
18         </div>
19         <div class="form-group">
20             <label for="nimhs">NIM</label>
21             <input type="number" class="form-control" required="required" name="nimhs"> <br/>
22         </div>
23         <div class="form-group">
24             <label for="emailmhs">E-mail</label>
25             <input type="email" class="form-control" required="required" name="emailmhs"> <br/>
26         </div>
27         <div class="form-group">
28             <label for="jurusanmhs">Jurusan</label>
29             <input type="text" class="form-control" required="required" name="jurusanmhs"> <br/>
30         </div>
31         <button type="submit" name="tambah" class="btn btn-primary float-right">Tambah Data</button>
32     </form>
33 @endsection

```

Keterangan:

- 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 form hidden.

- 4) Ketika tombol simpan ditekan, akan dipanggil routes /mahasiswa/simpan. Oleh karena itu, kita buat terlebih dahulu route tersebut.

```
23
24 Route::post('/mahasiswa/simpan', 'MahasiswaController@simpan');
```

Keterangan:

- Pada route ini menggunakan metode post karena data mahasiswa dari form akan dikirim ke method simpan di MahasiswaController.php

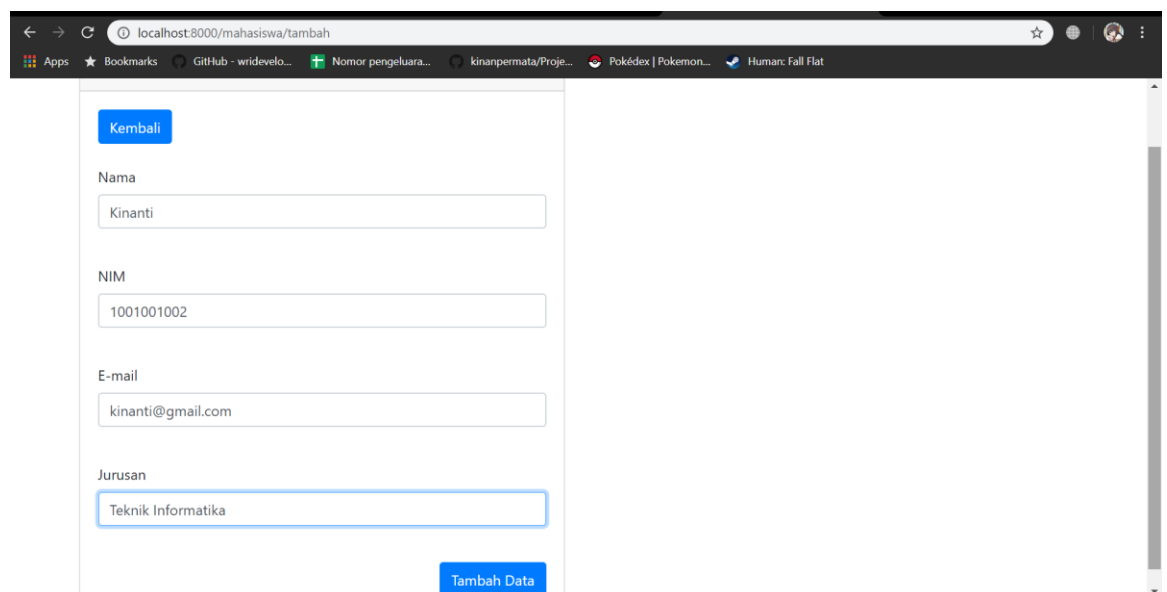
- 5) Buat method simpan pada MahasiswaController untuk menyimpan data ke database.

```
24
25 public function simpan(Request $request)
26 {
27     // Insert data ke tabel mahasiswa
28     DB::table('mahasiswa')->insert([
29         'nama' => $request->namamhs,
30         'nim' => $request->nimmhs,
31         'email' => $request->emailmhs,
32         'jurusan' => $request->jurusanmhs,
33     ]);
34     return redirect('/');
35 }
```

Keterangan:

- Variabel \$request untuk menerima data yang akan ditambahkan ke database
- Line 28-33 merupakan query builder untuk insert data ke tabel mahasiswa

- 6) 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.

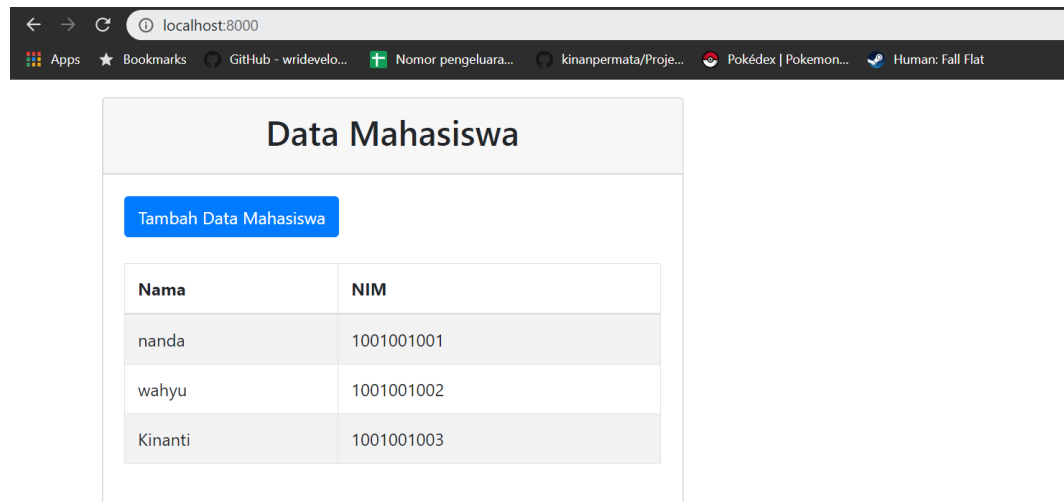


The screenshot shows a web browser window with the address bar displaying 'localhost:8000/mahasiswa/tambah'. The page contains a form with the following fields and values:

- Nama:** Kinanti
- NIM:** 1001001002
- E-mail:** kinanti@gmail.com
- Jurusan:** Teknik Informatika

There are two buttons: 'Kembali' (Back) at the top left and 'Tambah Data' (Add Data) at the bottom right.

Setelah kita klik tombol tambah data, maka data baru akan ditampilkan pada halaman index.



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

d. Melihat Detail Data dari Database

- 1) Buatlah tombol untuk melihat detail data (Line 28) pada file index.blade.php di folder resources/views

```
resources > views > index.blade.php > table.table-bordered.table-hover.table-striped > thead > tr > th
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                 <th> </th>
20             </tr>
21         </thead>
22         <tbody>
23             @foreach($mahasiswa as $mhs)
24                 <tr>
25                     <td>{{ $mhs->nama }}</td>
26                     <td>{{ $mhs->nim }}</td>
27                     <td>
28                         <a href="/mahasiswa/detail/{{ $mhs->id }}" class="badge badge-info">Detail</a>
29                     </td>
30                 </tr>
31             @endforeach
32         </tbody>
33     </table>
34 @endsection
```

- 2) Buatlah route baru pada routes/web.php dengan nama /mahasiswa/detail yang akan menjalankan fungsi detail pada MahasiswaController

```
25
26 Route::get('/mahasiswa/detail/{id}', 'MahasiswaController@detail');
```

- 3) Buat method detail pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan data mahasiswa pada view detail.blade.php.

```
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 }
```

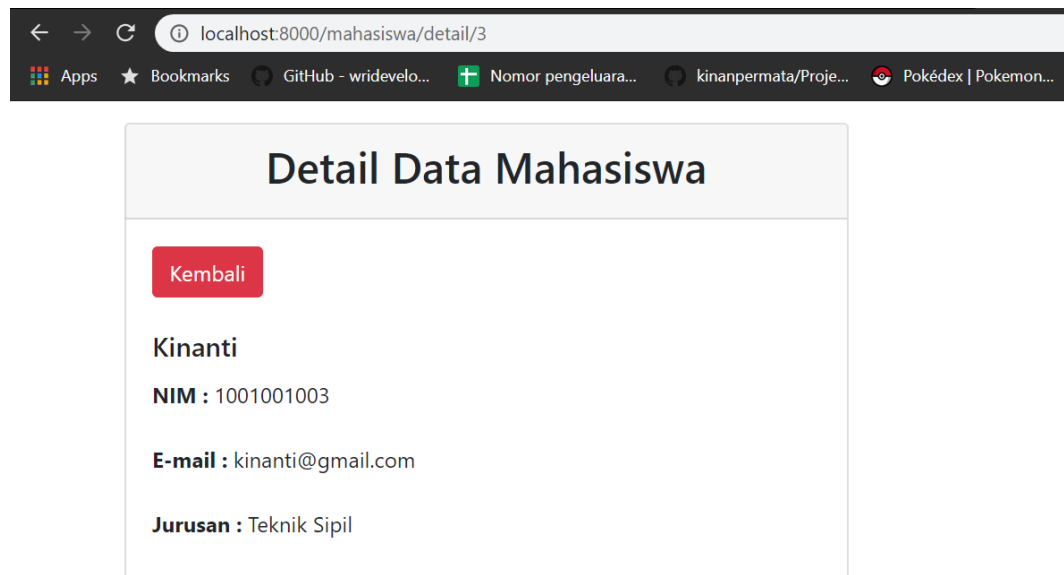
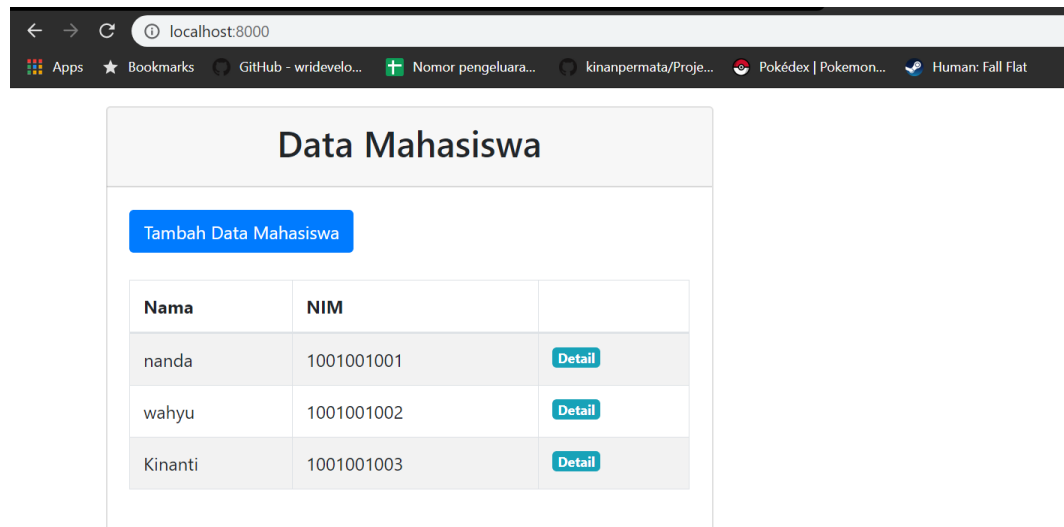
- 4) Kemudian buatlah view detail.blade.php yang menampilkan detail data mahasiswa pada folder resources/views. View detail juga mengaplikasikan master.blade.php.

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

Keterangan:

- Line 16-27 digunakan untuk menampilkan data mahasiswa berupa nama, nim, email, dan jurusan

- 5) Jalankan localhost:8000 dan pilih tombol 'Detail' di suatu data yang ingin kita lihat detailnya.



e. Mengubah Data (Update) dari Database

- 1) Buatlah tombol untuk mengubah data (Line 29) pada file index.blade.php di folder resources/views

```
22 <tbody>
23   @foreach($mahasiswa as $mhs)
24     <tr>
25       <td>{{ $mhs->nama }}</td>
26       <td>{{ $mhs->nim }}</td>
27       <td>
28         <a href="/mahasiswa/detail/{{ $mhs->id }}" class="badge badge-info">Detail</a>
29         <a href="/mahasiswa/edit/{{ $mhs->id }}" class="badge badge-warning">Edit</a>
30       </td>
31     </tr>
32   @endforeach
33 </tbody>
```

- 2) Buatlah route baru pada routes/web.php dengan nama /mahasiswa/edit yang akan menjalankan fungsi edit pada MahasiswaController

```
27
28 Route::get('/mahasiswa/edit/{id}', 'MahasiswaController@edit');
```

- 3) Buat method edit pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan view edit.

```
46 public function edit($id)
47 {
48     // Mengambil data mahasiswa berdasarkan id yang dipilih
49     $mahasiswa = DB::table('mahasiswa')->where('id', $id)->get();
50     // Kirim data mahasiswa yang diambil ke view edit.blade.php
51     return view('edit',['mahasiswa' => $mahasiswa]);
52 }
```

Keterangan :

- Line 49 : query builder untuk mengambil data mahasiswa berdasarkan id yang dipilih
- 4) Kemudian buatlah view edit.blade.php yang berisi form untuk mengubah data pada folder resources/views. View edit juga mengaplikasikan master.blade.php.

```
resources > views > edit.blade.php > form > button.btn.btn-primary.float-right
1 @extends('master')
2 <!-- Isi title -->
3 @section('title', 'Edit Data')
4
5 <!-- Isi bagian judul halaman-->
6 @section('judul_halaman', 'Edit Data Mahasiswa')
7
8 <!-- Isi bagian konten-->
9 @section('konten')
10 <a href="/" class="btn btn-danger">Kembali</a>
11 <br/>
12 <br/>
13 @foreach($mahasiswa as $mhs)
14 <form action="/mahasiswa/update" method="post">
15     {{ csrf_field() }}
16     <input type="hidden" name="id" value="{{ $mhs->id }}"> <br/>
17     <div class="form-group">
18         <label for="namamhs">Nama</label>
19         <input type="text" class="form-control" required="required" name="namamhs" value="{{ $mhs->nama }}"> <br/>
20     </div>
21     <div class="form-group">
22         <label for="nimhs">NIM</label>
23         <input type="number" class="form-control" required="required" name="nimhs" value="{{ $mhs->nim }}"> <br/>
24     </div>
25     <div class="form-group">
26         <label for="emailmhs">E-mail</label>
27         <input type="email" class="form-control" required="required" name="emailmhs" value="{{ $mhs->email }}"> <br/>
28     </div>
29     <div class="form-group">
30         <label for="jurusanmhs">Jurusan</label>
31         <input type="text" class="form-control" required="required" name="jurusanmhs" value="{{ $mhs->jurusan }}"> <br/>
32     </div>
33     <button type="submit" name="edit" class="btn btn-primary float-right">Edit Data</button>
34 </form>
35 @endforeach
36 @endsection
```

Keterangan:

- 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

5) Ketika tombol simpan ditekan, akan dipanggil routes /mahasiswa/update. Oleh karena itu, kita buat terlebih dahulu route tersebut.

```
29
30 Route::post('/mahasiswa/update', 'MahasiswaController@update');
```

Keterangan:

- Pada route ini menggunakan metode post karena data mahasiswa dari form akan dikirim ke method update di MahasiswaController.php

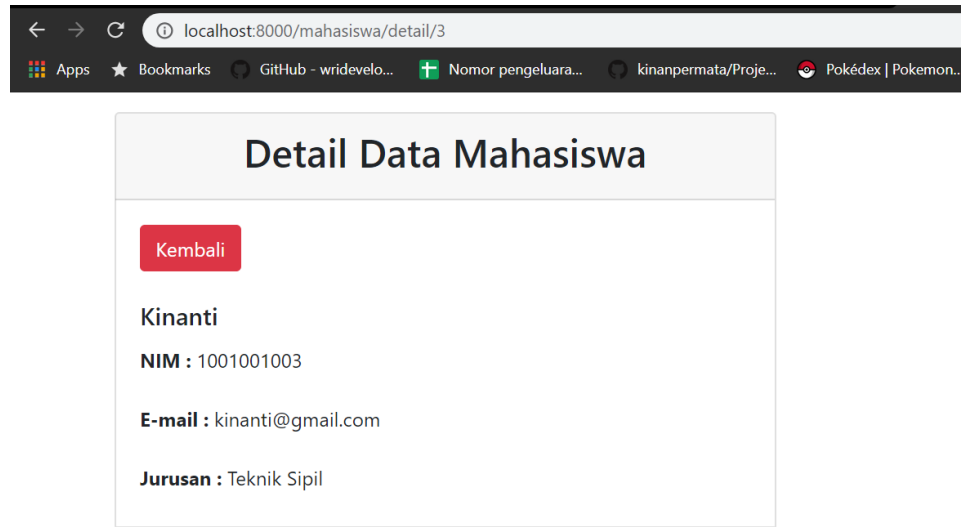
6) Buat method update pada MahasiswaController untuk menyimpan data yang diubah ke database.

```
55 public function update(Request $request)
56 {
57     // update data ke tabel mahasiswa
58     DB::table('mahasiswa')->where('id', $request->id)->update([
59         'nama' => $request->namamhs,
60         'nim' => $request->nimmhs,
61         'email' => $request->emailmhs,
62         'jurusan' => $request->jurusanmhs,
63     ]);
64     return redirect('/');
65 }
```

Keterangan:

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

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



← → ↻ localhost:8000/mahasiswa/detail/3

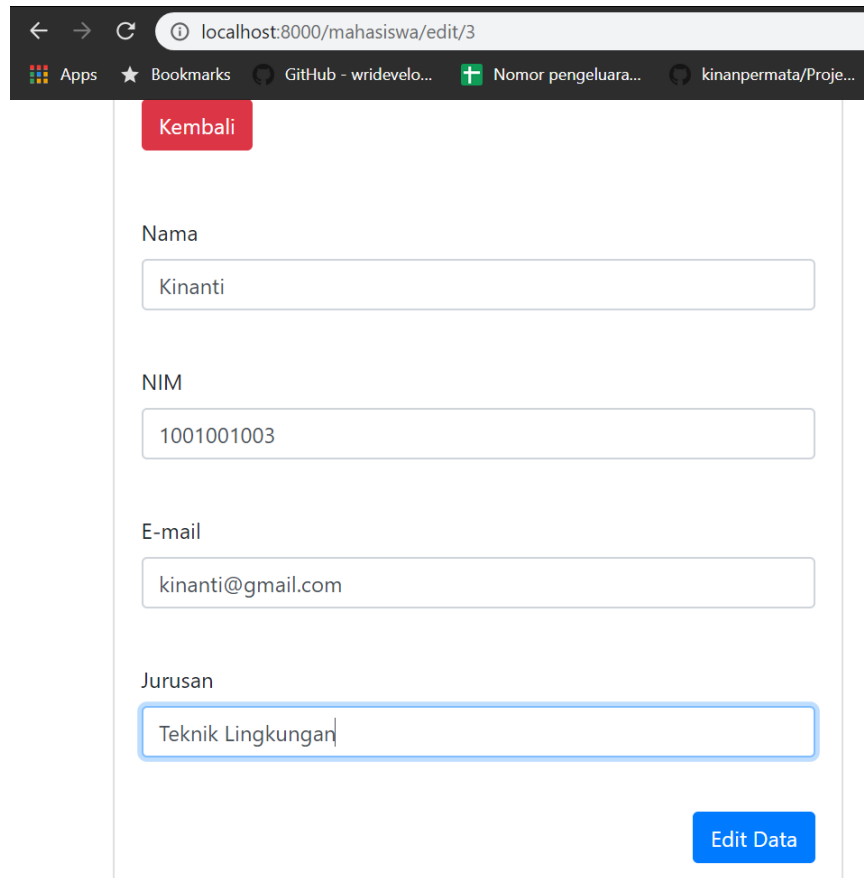
Apps ★ Bookmarks GitHub - wridevelo... Nomor pengeluaran... kinanpermata/Proje... Pokédex | Pokemon...

Detail Data Mahasiswa

[Kembali](#)

Kinanti
NIM : 1001001003
E-mail : kinanti@gmail.com
Jurusan : Teknik Sipil

Klik edit di data ketiga, kita akan mengubah namanya.



← → ↻ localhost:8000/mahasiswa/edit/3

Apps ★ Bookmarks GitHub - wridevelo... Nomor pengeluaran... kinanpermata/Proje...

[Kembali](#)

Nama

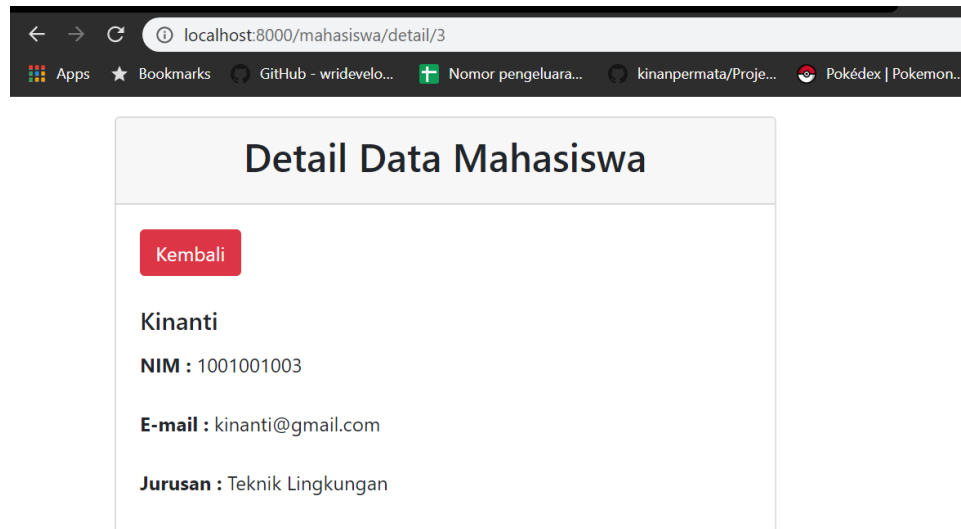
NIM

E-mail

Jurusan

[Edit Data](#)

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



f. Menghapus Data (Delete) dari Database

- 1) Buatlah tombol untuk menghapus data (Line 30) pada file index.blade.php di folder resources/views

```
22 <tbody>
23     @foreach($mahasiswa as $mhs)
24         <tr>
25             <td>{{ $mhs->nama }}</td>
26             <td>{{ $mhs->nim }}</td>
27             <td>
28                 <a href="/mahasiswa/detail/{{ $mhs->id }}" class="badge badge-info">Detail</a>
29                 <a href="/mahasiswa/edit/{{ $mhs->id }}" class="badge badge-warning">Edit</a>
30                 <a href="/mahasiswa/hapus/{{ $mhs->id }}" class="badge badge-danger">Hapus</a>
31             </td>
32         </tr>
33     @endforeach
34 </tbody>
```

- 2) Buatlah route baru pada routes/web.php dengan nama /mahasiswa/hapus yang akan menjalankan fungsi hapus pada MahasiswaController

```
31
32 Route::get('/mahasiswa/hapus/{id}', 'MahasiswaController@hapus');
```

- 3) Buat method hapus pada MahasiswaController.php di folder app/Http/Controllers yang akan menjalankan fungsi hapus.

```
67 public function hapus($id)
68 {
69     // Menghapus data mahasiswa berdasarkan id yang dipilih
70     $mahasiswa = DB::table('mahasiswa')->where('id', $id)->delete();
71
72     return redirect('/');
73 }
```

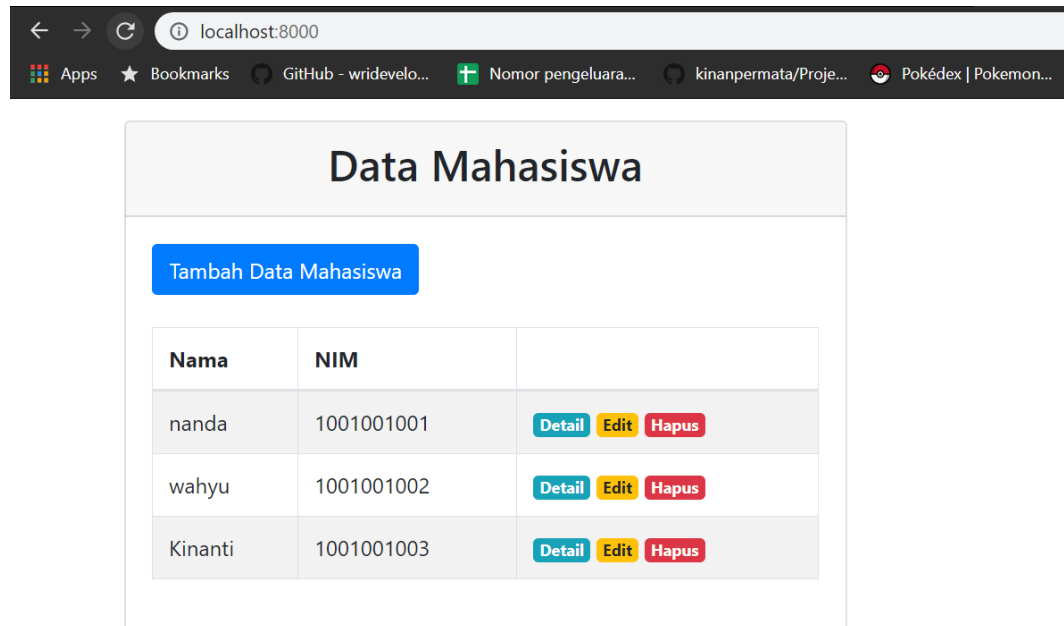
Keterangan :

- Line 67 : query builder untuk menghapus data mahasiswa berdasarkan id yang dipilih

4) Jalankan localhost:8000 dan pilih tombol 'Hapus', maka data yang terpilih akan dihapus.

Contoh: menghapus data terakhir.

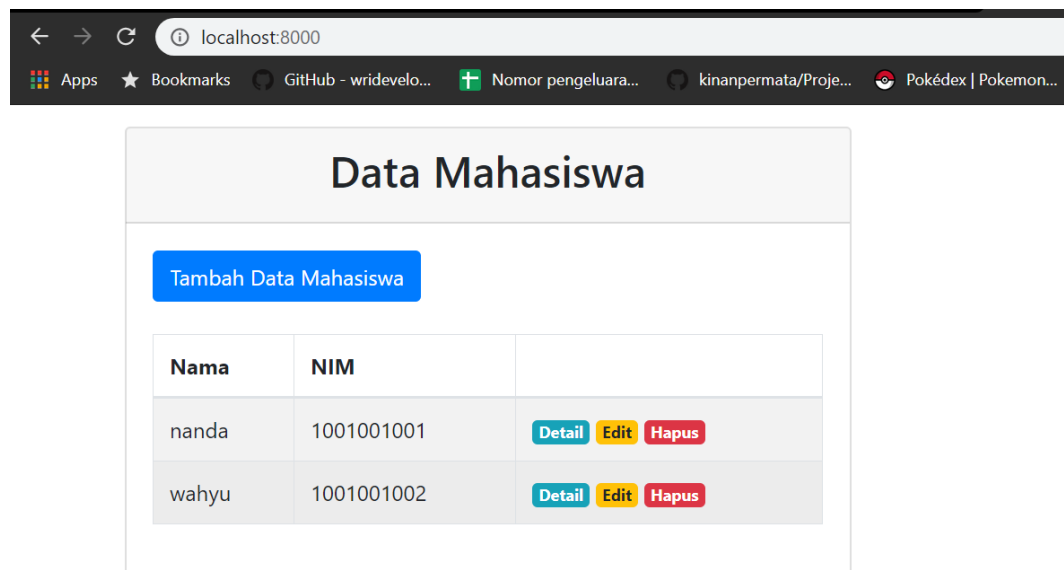
Sebelum:



The screenshot shows a web browser at localhost:8000 displaying a page titled "Data Mahasiswa". Below the title is a blue button labeled "Tambah Data Mahasiswa". Underneath is a table with three columns: "Nama", "NIM", and an empty column for actions. The table contains three rows of student data. Each row has "Detail", "Edit", and "Hapus" buttons in the action column.

Nama	NIM	
nanda	1001001001	Detail Edit Hapus
wahyu	1001001002	Detail Edit Hapus
Kinanti	1001001003	Detail Edit Hapus

Sesudah:



The screenshot shows the same web application after the deletion of the last student. The table now only contains two rows of student data, as the entry for "Kinanti" has been removed.

Nama	NIM	
nanda	1001001001	Detail Edit Hapus
wahyu	1001001002	Detail Edit Hapus

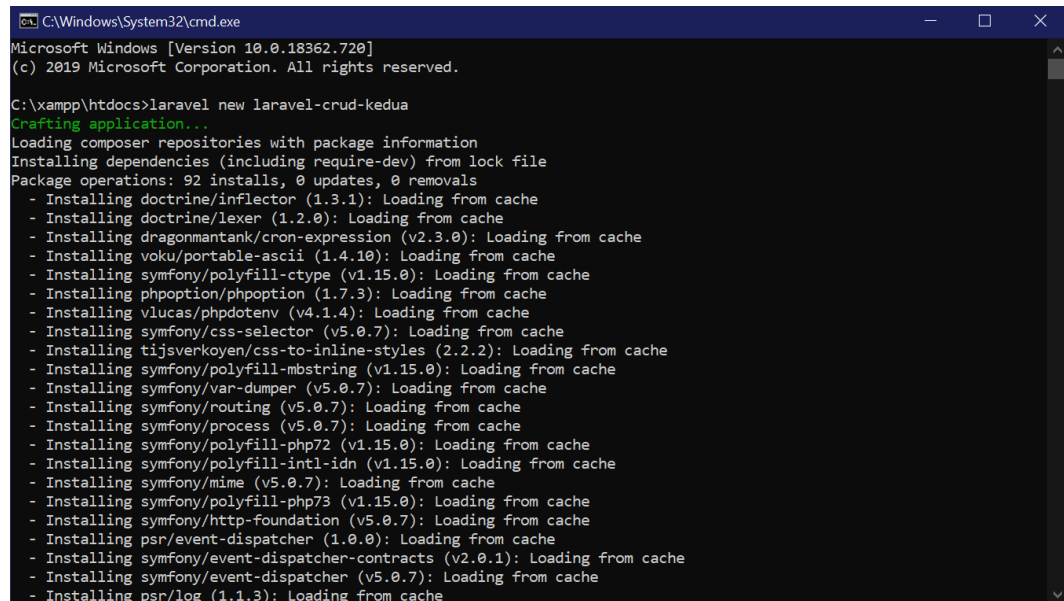
2. Praktikum – Bagian 2: Membuat CRUD di Laravel menggunakan Eloquent

a. Konfigurasi Database

- 1) Buatlah project Laravel baru dengan nama laravel-crud-kedua. Buka command prompt, tuliskan perintah berikut.

```
cd C:\xampp\htdocs
```

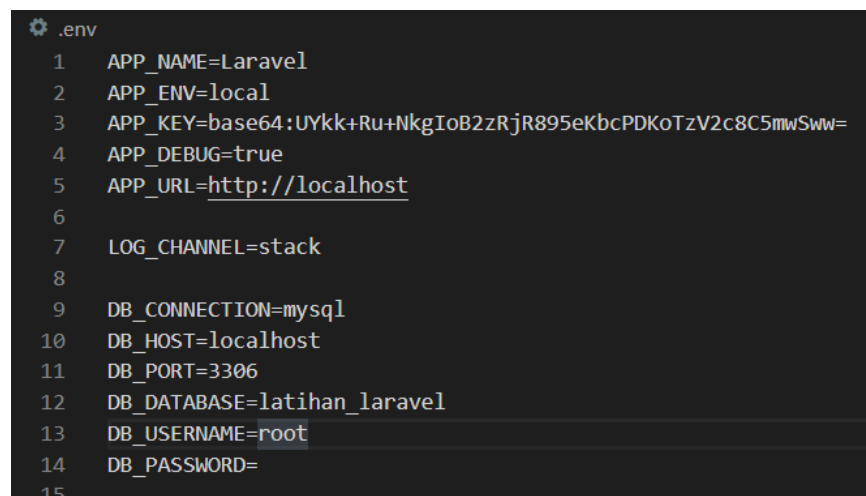
```
laravel new laravel-crud-kedua
```



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

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
- Installing phpoption/phpoption (1.7.3): Loading from cache
- Installing vlucas/phpdotenv (v4.1.4): Loading from cache
- Installing symfony/css-selector (v5.0.7): Loading from cache
- Installing tijsverkoyen/css-to-inline-styles (2.2.2): Loading from cache
- Installing symfony/polyfill-mbstring (v1.15.0): Loading from cache
- Installing symfony/var-dumper (v5.0.7): Loading from cache
- Installing symfony/routing (v5.0.7): Loading from cache
- Installing symfony/process (v5.0.7): Loading from cache
- Installing symfony/polyfill-php72 (v1.15.0): Loading from cache
- Installing symfony/polyfill-intl-idn (v1.15.0): Loading from cache
- Installing symfony/polyfill-intl-normalizer (v1.15.0): Loading from cache
- Installing symfony/mime (v5.0.7): Loading from cache
- Installing symfony/polyfill-php73 (v1.15.0): Loading from cache
- Installing symfony/http-foundation (v5.0.7): Loading from cache
- Installing psr/event-dispatcher (1.0.0): Loading from cache
- Installing psr/event-dispatcher-contracts (v2.0.1): Loading from cache
- Installing symfony/event-dispatcher (v5.0.7): Loading from cache
- Installing psr/log (1.1.3): Loading from cache
```

- 2) Selanjutnya kita lakukan konfigurasi database di Laravel. Untuk melakukan konfigurasi database, bukalah file .env pada project laravel-crud. Ubah seperti di bawah ini.



```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:UYkk+Ru+NkgIoB2zRjR895eKbcPDKoTzV2c8C5mmwSww=
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=
15
```

Keterangan:

- Nama database yang akan digunakan adalah latihan_laravel dengan username root.

- 3) Pada project ini kita gunakan database dan tabel yang sebelumnya digunakan pada Praktikum Bagian 1. Tambahkan kolom `created_at` dan `updated_at` yang bertipe `TIMESTAMP` dan default nilainya `NULL`

✓ Table mahasiswa has been altered successfully.

```
ALTER TABLE `mahasiswa` CHANGE `created_at` `created_at` TIMESTAMP on update CURRENT_TIMESTAMP NULL DEFAULT NULL;
```

[Edit inline]

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id	int(11)			No	None		AUTO_INCREMENT	Change Drop More
2	nama	varchar(100)	latin1_swedish_ci		No	None			Change Drop More
3	nim	varchar(10)	latin1_swedish_ci		No	None			Change Drop More
4	email	varchar(50)	latin1_swedish_ci		No	None			Change Drop More
5	jurusan	varchar(20)	latin1_swedish_ci		No	None			Change Drop More
6	created_at	timestamp		on update CURRENT_TIMESTAMP	Yes	NULL		ON UPDATE CURRENT_TIMESTAMP	Change Drop More
7	updated_at	timestamp			Yes	NULL			Change Drop More

b. Menampilkan Data dari Database

- 1) Setelah kita memiliki beberapa data pada tabel mahasiswa, kita akan mencoba untuk menampilkan data tersebut ketika project dijalankan.

Pertama, buatlah route pada `routes/web.php` sehingga ketika pertama kali project dijalankan akan terbuka halaman yang menampilkan data.

```
20
21 Route::get('/', 'MahasiswaController@index');
22 Route::get('/mahasiswa', 'MahasiswaController@index');
```

- 2) Buat model menggunakan command prompt dengan nama Mahasiswa menggunakan php artisan

`cd laravel-crud-kedua`

`php artisan make:model Mahasiswa`

```
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\xampp\htdocs\laravel-crud-kedua>php artisan make:model Mahasiswa
Model created successfully.
```

- 3) Ubah model Mahasiswa.php pada folder App menjadi seperti berikut.

```
app > Mahasiswa.php > PHP Intelephense > Mahasiswa
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Mahasiswa extends Model
8  {
9      protected $table = "mahasiswa";
10 }
11
```

Keterangan:

- Model Mahasiswa akan menangani tabel mahasiswa
- 4) Buat controller baru yaitu MahasiswaController menggunakan php artisan
php artisan make:controller MahasiswaController

```
C:\xampp\htdocs\laravel-crud-kedua>php artisan make:controller MahasiswaController
Controller created successfully.
```

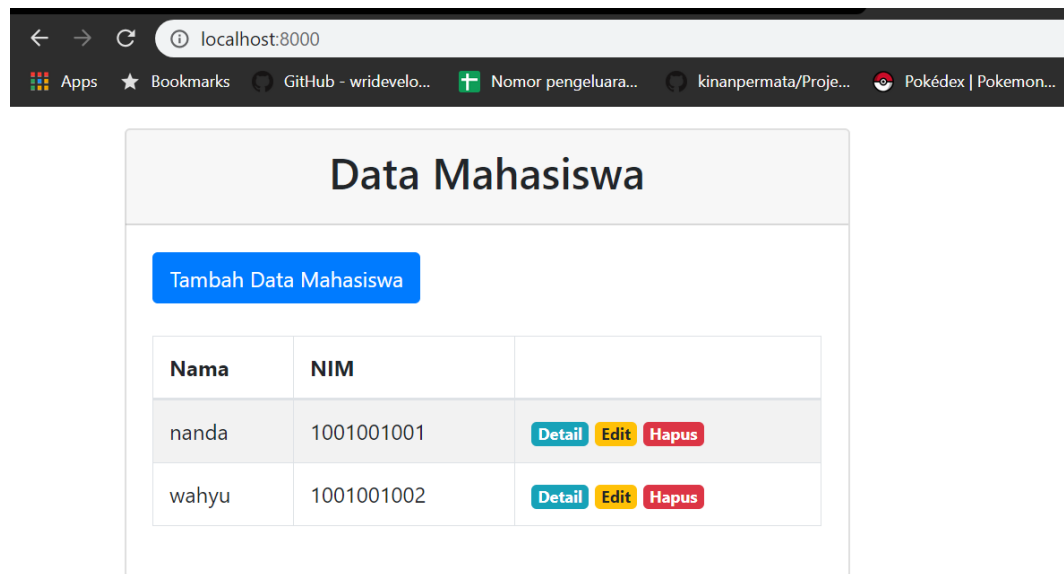
- 5) Buat method index pada MahasiswaController.php pada folder app/Http/Controllers

```
app > Http > Controllers > MahasiswaController.php > PHP Intelephense > Mahas
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 }
16
```

Keterangan:

- 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
- 6) 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)
- 7) Jalankan command prompt, tuliskan perintah untuk menjalankan project laravel-crud php artisan serve
- Buka browser dan ketikkan localhost:8000, maka akan tampil sebagai berikut



c. Memasukkan Data (Create) ke Database

- 1) Buatlah route baru pada routes/web.php dengan nama /mahasiswa/tambah yang akan menjalankan fungsi tambah pada MahasiswaController ketika tombol Tambah Data Mahasiswa ditekan.

```
22 Route::get('/mahasiswa', 'MahasiswaController@index');
23 Route::get('/mahasiswa/tambah', 'MahasiswaController@tambah');
```

- 2) Buat method tambah pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan view tambah.

```
16 public function tambah()
17 {
18     return view('tambah');
19 }
```

- 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) Ketika tombol simpan ditekan, akan dipanggil routes /mahasiswa/simpan. Oleh karena itu, kita buat terlebih dahulu route tersebut.

```
24
25 Route::post('/mahasiswa/simpan', 'MahasiswaController@simpan');
```

Keterangan:

- Pada route ini menggunakan metode post karena data mahasiswa dari form akan dikirim ke method simpan di MahasiswaController.php
- 5) Buat method simpan pada MahasiswaController untuk menyimpan data ke database.

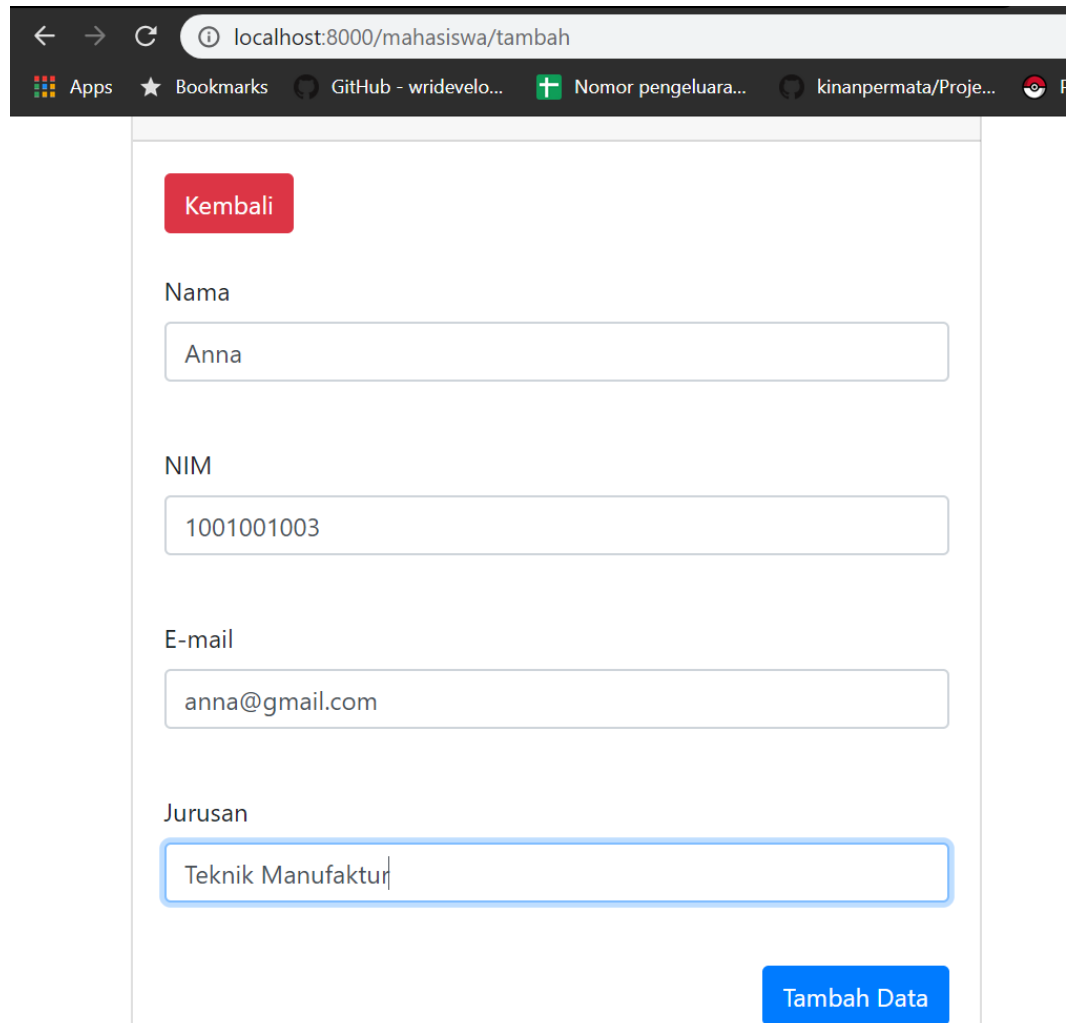
```
21 public function simpan(Request $request)
22 {
23     Mahasiswa::create([
24         'nama' => $request->namamhs,
25         'nim' => $request->nimmhs,
26         'email' => $request->emailmhs,
27         'jurusan' => $request->jurusanmhs,
28     ]);
29
30     return redirect('/mahasiswa');
31 }
```

Keterangan:

- 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
- 6) 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.

```
7 class Mahasiswa extends Model
8 {
9     protected $table = "mahasiswa";
10    protected $fillable = ['nama', 'nim', 'email', 'jurusan'];
11 }
12
```

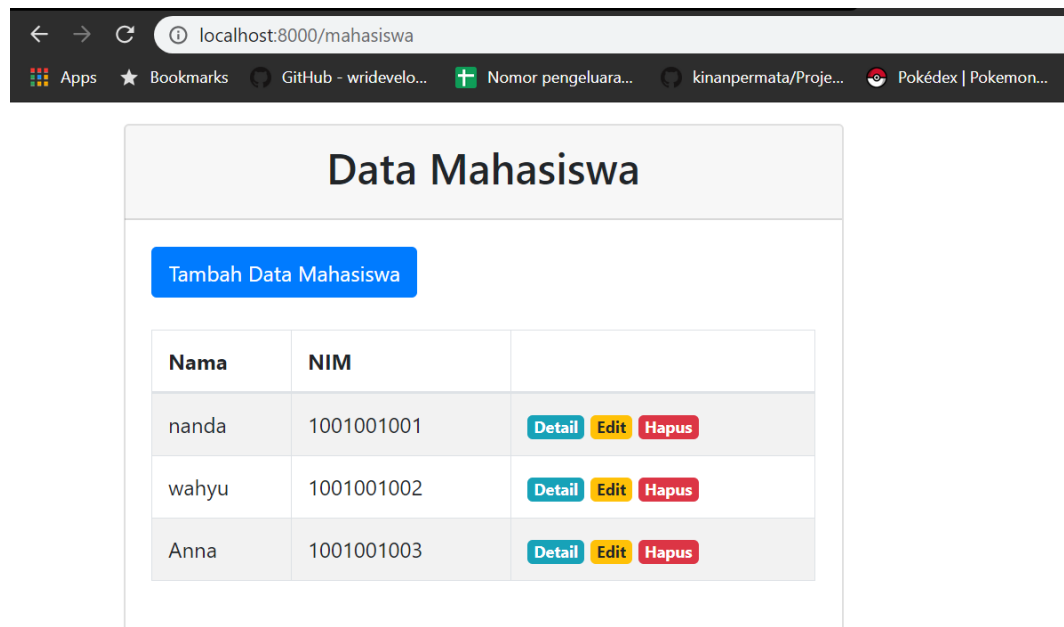
- 7) Kembali coba jalankan localhost:8000 dan pilih tombol ‘Tambah Data Mahasiswa’, maka akan ditampilkan halaman tambah yang berisi form untuk memasukkan data baru. Kita coba isikan data pada form tersebut.



The screenshot shows a web browser window with the address bar displaying `localhost:8000/mahasiswa/tambah`. The browser's bookmark bar includes links for 'Apps', 'Bookmarks', 'GitHub - widevelo...', 'Nomor pengeluaran...', 'kinanpermata/Proje...', and a 'P' icon. The web page itself has a light gray background and contains a form with the following elements:

- A red button labeled 'Kembali' at the top left.
- A text input field labeled 'Nama' containing the text 'Anna'.
- A text input field labeled 'NIM' containing the text '1001001003'.
- A text input field labeled 'E-mail' containing the text 'anna@gmail.com'.
- A text input field labeled 'Jurusan' containing the text 'Teknik Manufaktur'.
- A blue button labeled 'Tambah Data' at the bottom right.

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



d. Melihat Detail Data dari Database

- 1) Buatlah route baru pada routes/web.php dengan nama /mahasiswa/detail yang akan menjalankan fungsi detail pada MahasiswaController

```
26 |  
27 Route::get('/mahasiswa/detail/{id}', 'MahasiswaController@detail');
```

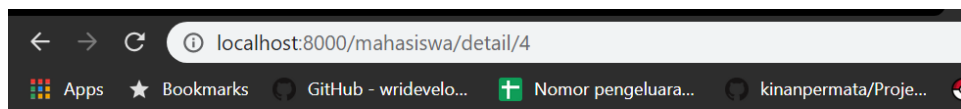
- 2) Buat method detail pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan data mahasiswa pada view detail.blade.php.

```
33 public function detail($id)  
34 {  
35     $mahasiswa = Mahasiswa::find($id);  
36     return view('detail', ['mahasiswa' => $mahasiswa]);  
37 }
```

- 3) Kemudian buatlah view detail.blade.php yang menampilkan detail data mahasiswa pada folder resources/views. View detail juga mengaplikasikan master.blade.php.

```
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     <a href="/" class="btn btn-danger">Kembali</a>
12
13     <br/>
14     <br/>
15
16     <h5 class="card-title">{{ $mahasiswa->nama }} </h5>
17     <p class="card-text">
18         <label for=""><b> NIM : </b></label>
19         {{ $mahasiswa->nim }} </p>
20     <p class="card-text">
21         <label for=""><b> E-mail : </b></label>
22         {{ $mahasiswa->email }} </p>
23     <p class="card-text">
24         <label for=""><b> Jurusan : </b></label>
25         {{ $mahasiswa->jurusan }} </p>
26 @endsection
```

- 4) Jalankan localhost:8000 dan pilih tombol 'Detail' di suatu data yang ingin kita lihat detailnya.



Detail Data Mahasiswa

[Kembali](#)

Anna
NIM : 1001001003
E-mail : anna@gmail.com
Jurusan : Teknik Manufaktur

e. Mengubah Data (Update) dari Database

- 1) Buatlah route baru pada routes/web.php dengan nama /mahasiswa/edit yang akan menjalankan fungsi edit pada MahasiswaController

```
26
27 Route::get('/mahasiswa/detail/{id}', 'MahasiswaController@detail');
28 Route::get('/mahasiswa/edit/{id}', 'MahasiswaController@edit');
```

- 2) Buat method edit pada MahasiswaController.php di folder app/Http/Controllers yang akan menampilkan view edit.

```
39 public function edit($id)
40 {
41     $mahasiswa = Mahasiswa::find($id);
42     return view('edit', ['mahasiswa' => $mahasiswa]);
43 }
```

Keterangan :

- Line 41 : fungsi eloquent untuk mengambil data mahasiswa berdasarkan id yang dipilih
- 3) Kemudian buatlah view edit.blade.php yang berisi form untuk mengubah data pada folder resources/views. View edit juga mengaplikasikan master.blade.php.

```
8 <!-- Isi bagian konten-->
9 @section('konten')
10 <a href="/" class="btn btn-danger">Kembali</a>
11 <br/>
12 <br/>
13 <form action="/mahasiswa/update/{{ $mahasiswa->id }}" method="POST">
14     {{ csrf_field() }}
15     <input type="hidden" name="id" value="{{ $mahasiswa->id }}"> <br>
16     <div class="form-group">
17         <label for="namamhs">Nama</label>
18         <input type="text" class="form-control" required="required" name="namamhs" value="{{ $mahasiswa->nama }}"> <br>
19     </div>
20     <div class="form-group">
21         <label for="nimhs">Nim</label>
22         <input type="number" class="form-control" required="required" name="nimhs" value="{{ $mahasiswa->nim }}"> <br>
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" value="{{ $mahasiswa->email }}"> <br>
27     </div>
28     <div class="form-group">
29         <label for="jurusanmhs">Jurusan</label>
30         <input type="text" class="form-control" required="required" name="jurusanmhs" value="{{ $mahasiswa->jurusan }}"> <br>
31     </div>
32     <button type="submit" name="edit" class="btn btn-primary float-right">Simpan Data</button>
33 </form>
34 @endsection
```

Keterangan:

- 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
- 4) Ketika tombol simpan ditekan, akan dipanggil routes /mahasiswa/update/{id}. Oleh karena itu, kita buat terlebih dahulu route tersebut.

```
29
30 Route::post('/mahasiswa/update/{id}', 'MahasiswaController@update');
31
```

Keterangan:

- Pada route ini menggunakan metode post karena data mahasiswa dari form akan dikirim ke method update di MahasiswaController.php
- 5) Buat method update pada MahasiswaController untuk menyimpan data yang diubah ke database.

```
45 public function update($id, Request $request)
46 {
47     $mahasiswa = Mahasiswa::find($id);
48     $mahasiswa->nama = $request->namamhs;
49     $mahasiswa->nim = $request->nimmhs;
50     $mahasiswa->email = $request->emailmhs;
51     $mahasiswa->jurusan = $request->jurusanmhs;
52     $mahasiswa->save();
53     return redirect('/mahasiswa');
54 }
```

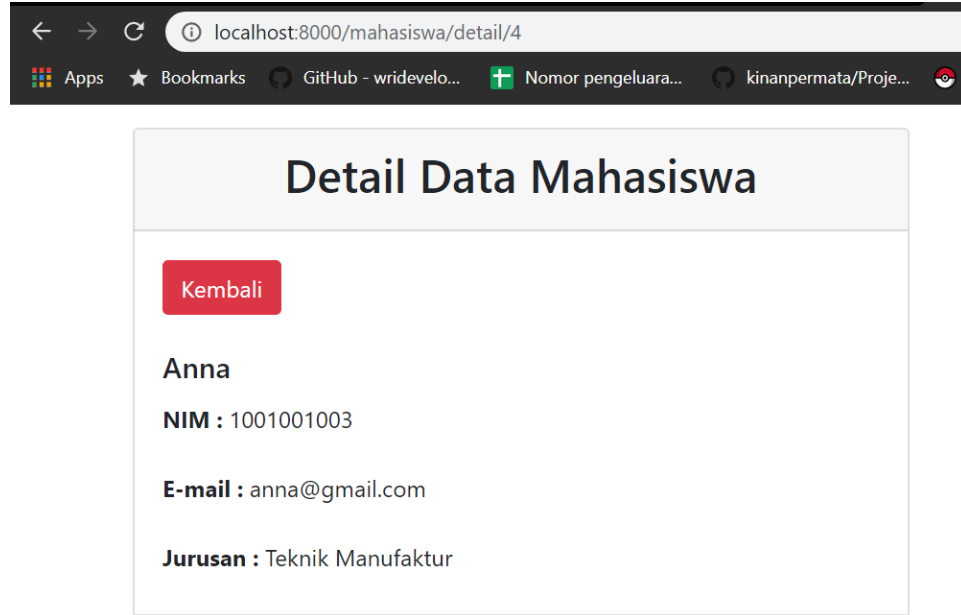
Keterangan:

- Variabel \$request untuk menerima data yang akan ditambahkan ke database
- Line 48-52 merupakan fungsi eloquent untuk update data ke tabel mahasiswa

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

Cobalah untuk mengedit data.

Sebelum:



← → ↻ ⓘ localhost:8000/mahasiswa/detail/4

Apps ★ Bookmarks GitHub - widevelo... + Nomor pengeluaran... kinanpermata/Proje...

Detail Data Mahasiswa

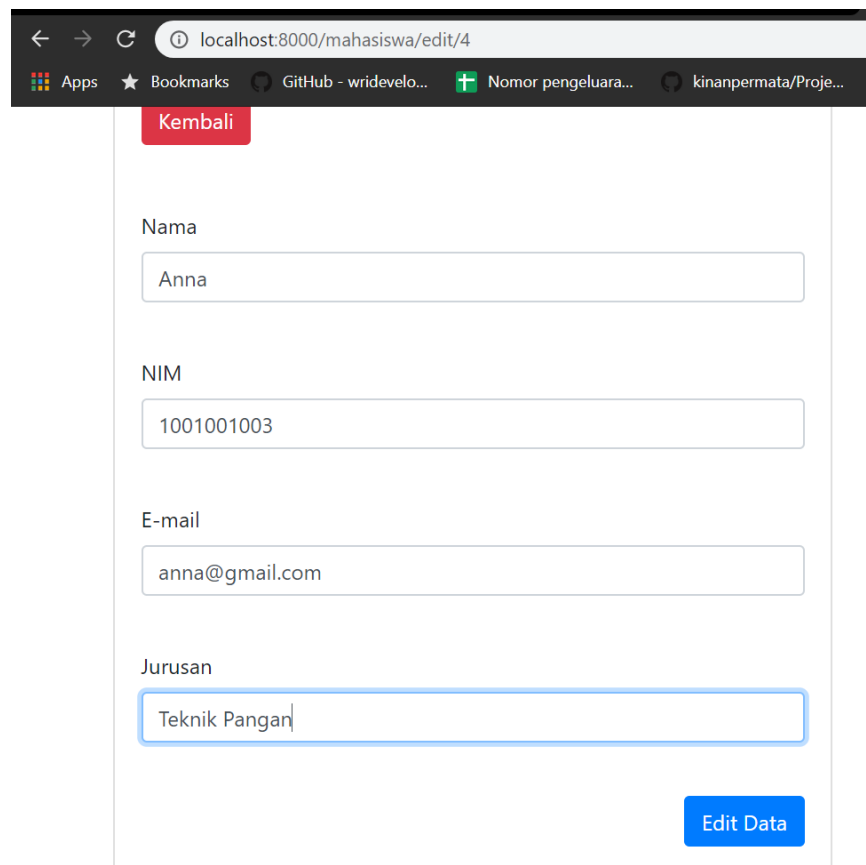
Kembali

Anna

NIM : 1001001003

E-mail : anna@gmail.com

Jurusan : Teknik Manufaktur



← → ↻ ⓘ localhost:8000/mahasiswa/edit/4

Apps ★ Bookmarks GitHub - widevelo... + Nomor pengeluaran... kinanpermata/Proje...

Kembali

Nama

Anna

NIM

1001001003

E-mail

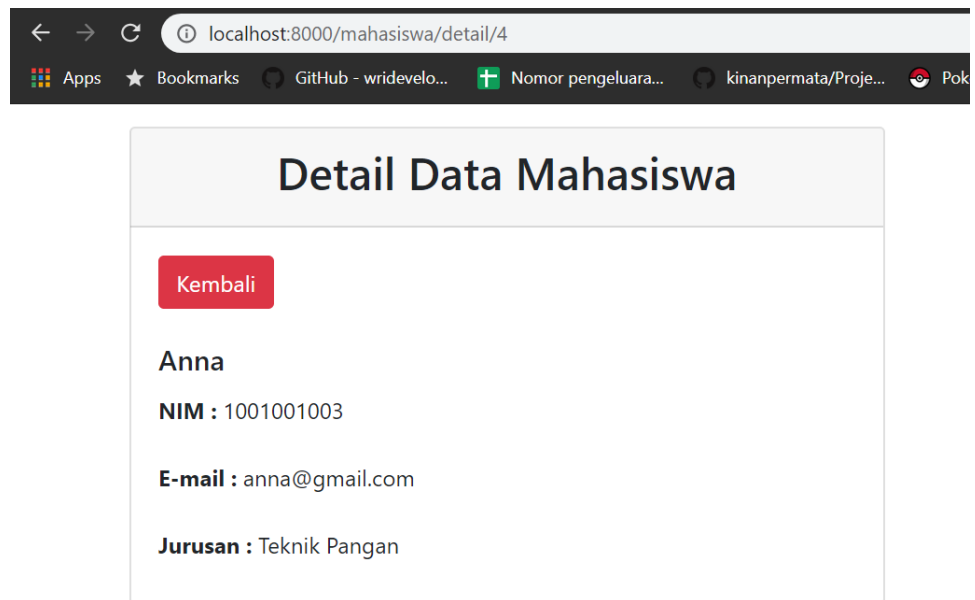
anna@gmail.com

Jurusan

Teknik Pangan

Edit Data

Sesudah:



f. Menghapus Data (Delete) dari Database

- 1) Buatlah route baru pada routes/web.php dengan nama /mahasiswa/hapus yang akan menjalankan fungsi hapus pada MahasiswaController

```
31  
32 Route::get('/mahasiswa/hapus/{id}', 'MahasiswaController@hapus');
```

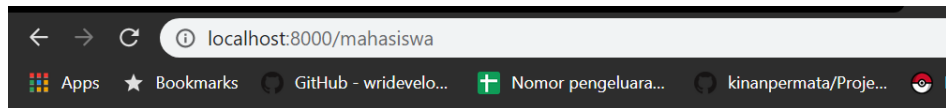
- 2) Buat method hapus pada MahasiswaController.php di folder app/Http/Controllers yang akan menjalankan fungsi hapus.

```
56 public function hapus($id)  
57 {  
58     $mahasiswa = Mahasiswa::find($id);  
59     $mahasiswa->delete();  
60     return redirect('/mahasiswa');  
61 }
```

Keterangan :

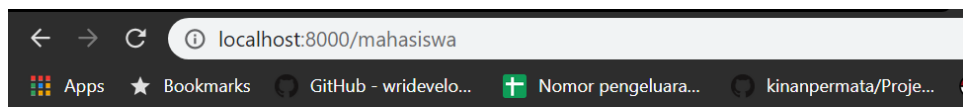
- Line 59 :fungsi eloquent untuk menghapus data mahasiswa berdasarkan id yang dipilih
- 3) Jalankan localhost:8000 dan pilih tombol 'Hapus', maka data yang terpilih akan dihapus.

Sebelum:



Data Mahasiswa		
Tambah Data Mahasiswa		
Nama	NIM	
nanda	1001001001	Detail Edit Hapus
wahyu	1001001002	Detail Edit Hapus
Anna	1001001003	Detail Edit Hapus
Budi	1001001004	Detail Edit Hapus

Sesudah:



Data Mahasiswa		
Tambah Data Mahasiswa		
Nama	NIM	
nanda	1001001001	Detail Edit Hapus
wahyu	1001001002	Detail Edit Hapus
Anna	1001001003	Detail Edit Hapus