



Jurusan Teknologi Informasi Politeknik Negeri Malang

Jobsheet-14: Membuat RESTful API Laravel

Mata Kuliah Pemrograman Web Lanjut

Pengampu: Tim Ajar Pemrograman Web Lanjut

Mei 2020

NAMA : Dina Lisuardi

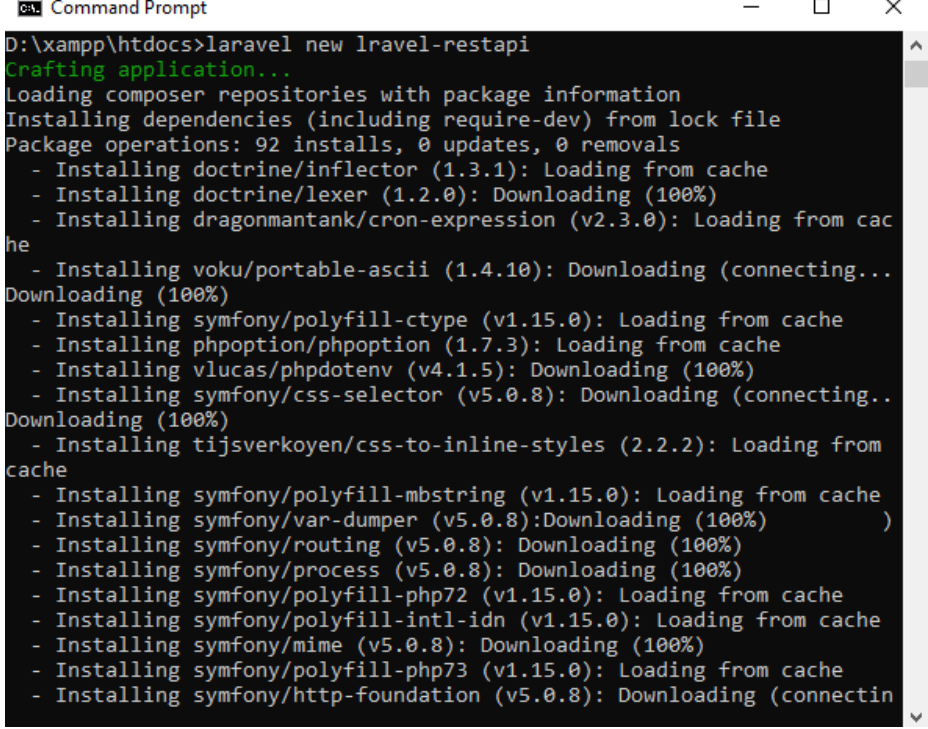
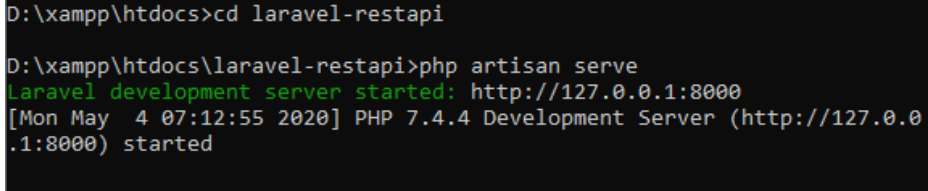
NIM : 1941723004

Kelas : TI 2B

Isi Laporan : Jobsheet13 – RestFull API Laravel

Praktikum: Membuat RESTful API di Laravel

Langkah	Keterangan
1	Buat project baru dengan nama “ laravel-restapi ”. Buka command prompt, tuliskan perintah berikut. cd C:\xampp\htdocs laravel new laravel- restapi

	 <pre> D:\xampp\htdocs>laravel new laravel-restapi 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): Downloading (100%) - Installing dragonmantank/cron-expression (v2.3.0): Loading from cache - Installing voku/portable-ascii (1.4.10): Downloading (connecting... Downloading (100%) - Installing symfony/polyfill-ctype (v1.15.0): Loading from cache - Installing phpoption/phpooption (1.7.3): Loading from cache - Installing vlucas/phpdotenv (v4.1.5): Downloading (100%) - Installing symfony/css-selector (v5.0.8): Downloading (connecting.. Downloading (100%) - 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 (v5.0.8): Downloading (100%) - Installing symfony/routing (v5.0.8): Downloading (100%) - Installing symfony/process (v5.0.8): Downloading (100%) - Installing symfony/polyfill-php72 (v1.15.0): Loading from cache - Installing symfony/polyfill-intl-idn (v1.15.0): Loading from cache - Installing symfony/mime (v5.0.8): Downloading (100%) - Installing symfony/polyfill-php73 (v1.15.0): Loading from cache - Installing symfony/http-foundation (v5.0.8): Downloading (connectin </pre>
2	<p>Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut. cd C:\laravel-restapi php artisan serve Akan tampil halaman default Laravel seperti di bawah ini.</p>  <pre> D:\xampp\htdocs>cd laravel-restapi D:\xampp\htdocs\laravel-restapi>php artisan serve Laravel development server started: http://127.0.0.1:8000 [Mon May 4 07:12:55 2020] PHP 7.4.4 Development Server (http://127.0.0.1:8000) started </pre>

	
3	<p>Kemudian lakukan konfigurasi database pada file .env. Isikan nama database, username, dan password yang akan digunakan. Pada project ini, kita gunakan database dari latihan di minggu-minggu sebelumnya yaitu “latihan_laravel”</p> 
4	<p>Buat model dengan nama Mahasiswa, buat juga controllernya. Untuk membuat model dan controllernya sekaligus tuliskan perintah berikut pada command prompt</p>

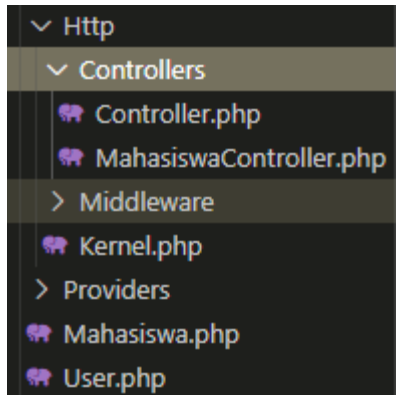
(terlebih dahulu keluar dari php artisan serve dengan mengetik ctrl+C pada keyboard)

php artisan make:model Mahasiswa -c

- Keterangan : -c merupakan perintah untuk menyertakan pembuatan controller

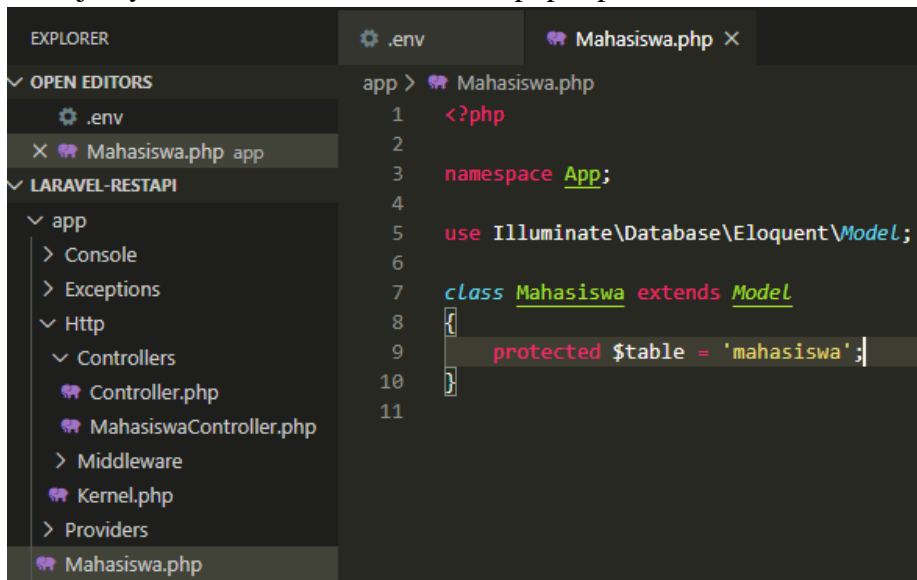
```
D:\xampp\htdocs\laravel-restapi>php artisan make:model Mahasiswa -c
Model created successfully.
Controller created successfully.
D:\xampp\htdocs\laravel-restapi>
```

Sehingga pada project laravel-restapi akan bertambah dua file yaitu model Mahasiswa.php serta controller MahasiswaController.php.



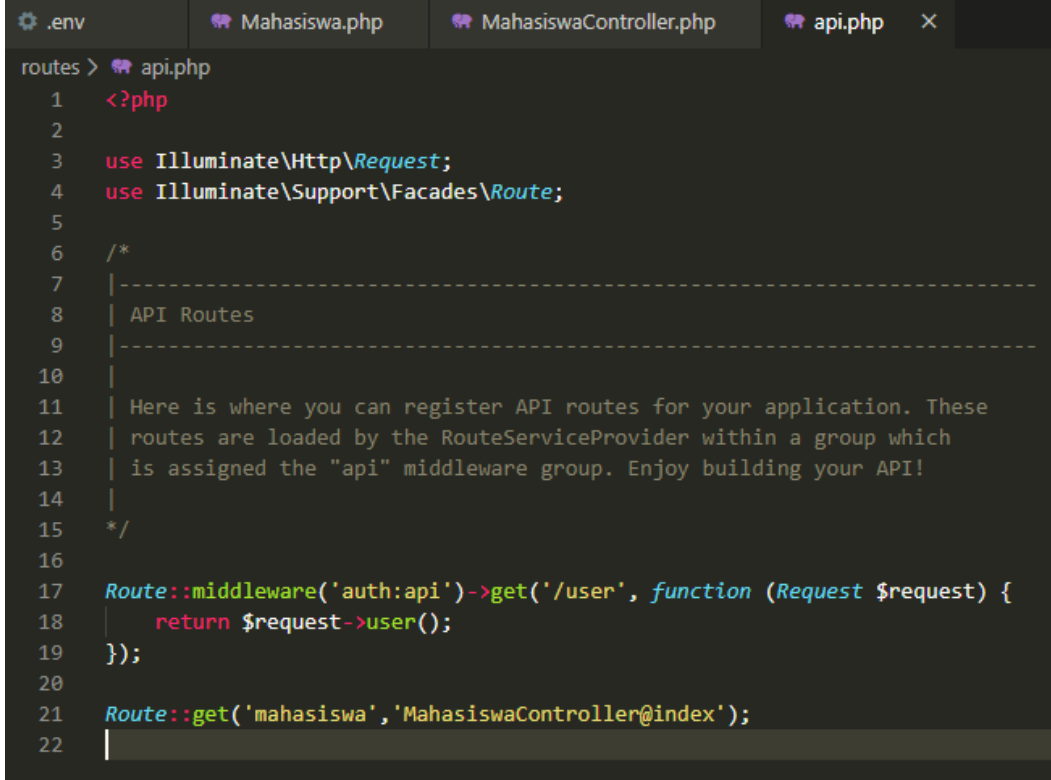
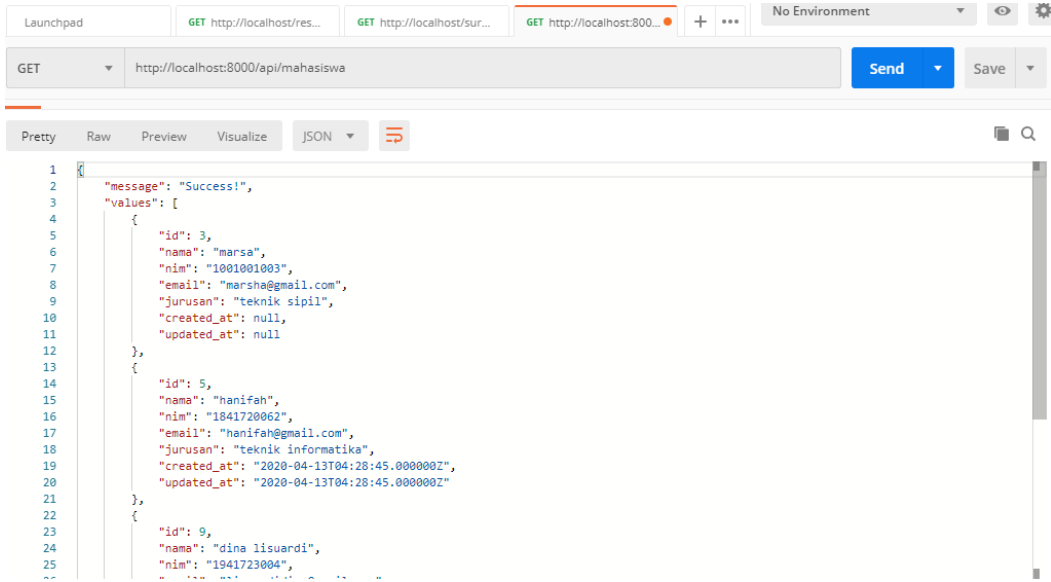
5

Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.



Keterangan:

	<ul style="list-style-type: none"> Model ini akan mengelola tabel “mahasiswa” yang terdapat pada database latihan_laravel
6	<p>Kemudian kita akan memodifikasi isi dari MahasiswaController.php untuk dapat mengolah data pada tabel ‘mahasiswa’. Pada controller ini, kita akan melakukan operasi untuk menampilkan, menambah, mengubah, dan menghapus data. Pertama, kita akan mengubah fungsi index agar saat fungsi index dipanggil, maka aplikasi akan menampilkan seluruh data dari tabel mahasiswa.</p>  <pre> 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 // fungsi index digunakan untuk menampilkan semua data mahasiswa 11 public function index(){ 12 \$data = Mahasiswa::all(); 13 14 // cek data tidak kosong 15 if(count(\$data) > 0){ 16 \$res['message'] = "Success!"; 17 \$res['values'] = \$data; 18 return response(\$res); 19 } 20 // jika data kosong 21 else{ 22 \$res['message'] = "Kosong!"; 23 return response(\$res); 24 } 25 } 26 } 27 </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> Tambahkan line 6 agar model Mahasiswa dapat digunakan pada MahasiswaController Line 15-19 digunakan untuk memeriksa apakah data>0 atau data tidak kosong Variabel \$res[message] digunakan untuk menampilkan pesan apakah ada data atau tidak ada data di tabel mahasiswa

	<ul style="list-style-type: none"> • Variabel \$array[values] akan menyimpan semua baris data pada tabel mahasiswa
7	<p>Tambahkan route untuk memanggil fungsi index pada file routes/api.php (Line 21).</p>  <pre> 1 <?php 2 3 use Illuminate\Http\Request; 4 use Illuminate\Support\Facades\Route; 5 6 /* 7 ----- 8 API Routes 9 ----- 10 11 Here is where you can register API routes for your application. These 12 routes are loaded by the RouteServiceProvider within a group which 13 is assigned the "api" middleware group. Enjoy building your API! 14 15 */ 16 17 Route::middleware('auth:api')->get('/user', function (Request \$request) { 18 return \$request->user(); 19 }); 20 21 Route::get('mahasiswa', 'MahasiswaController@index'); 22 </pre> <p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'.</p>
8	<p>Ketikkan perintah php artisan serve pada command prompt. Lalu kita coba menguji fungsi untuk menampilkan data menggunakan aplikasi Postman. Gunakan perintah GET, isikan url : http://localhost:8000/api/mahasiswa Berikut adalah tampilan dari aplikasi Postman.</p>  <pre> 1 { 2 "message": "Success!", 3 "values": [4 { 5 "id": 3, 6 "nama": "marsa", 7 "nim": "1001001003", 8 "email": "marsha@gmail.com", 9 "jurusan": "teknik sipil", 10 "created_at": null, 11 "updated_at": null 12 }, 13 { 14 "id": 5, 15 "nama": "hanifah", 16 "nim": "1841720062", 17 "email": "hanifah@gmail.com", 18 "jurusan": "teknik informatika", 19 "created_at": "2020-04-13T04:28:45.000000Z", 20 "updated_at": "2020-04-13T04:28:45.000000Z" 21 }, 22 { 23 "id": 9, 24 "nama": "dina lisuardi", 25 "nim": "1941723004", 26 "email": "dina_lisuardi@gmail.com", 27 "jurusan": "teknik informatika", 28 "created_at": "2020-04-13T04:28:45.000000Z", 29 "updated_at": "2020-04-13T04:28:45.000000Z" 30 } 31] 32 } </pre>

	Semua data pada tabel mahasiswa akan tampil, ditampilkan juga pesan sukses
9	<p>Selanjutnya kita akan menambahkan fungsi untuk melihat suatu data ketika dipilih ID tertentu. Buat fungsi baru yaitu getId pada MahasiswaController.php.</p> <pre> 27 // fungsi untuk menampilkan data dari sebuah ID 28 public function getId(\$id) 29 { 30 \$data = Mahasiswa::where('id',\$id)->get(); 31 32 // cek jika data ditemukan 33 if(count(\$data) > 0){ 34 \$res['message'] = "Success!"; 35 \$res['values'] = \$data; 36 return response(\$res); 37 } 38 // jika data tidak ditemukan 39 else{ 40 \$res['message'] = "Gagal!"; 41 return response(\$res); 42 } 43 } 44 } 45 </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> • Fungsi getId menerima parameter \$id yang menunjukkan ID mahasiswa yang dipilih • Line 30 merupakan pemanggilan model untuk membaca data berdasarkan ID
10	<p>Tambahkan route untuk memanggil fungsi getId pada routes/api.php</p> <pre> Route::get('/mahasiswa/{id}','MahasiswaController@getId'); </pre> <p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'</p>
11	<p>Sekarang kita coba untuk menampilkan data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah GET untuk menampilkan data.</p> <p>Di bawah ini adalah contoh untuk menampilkan data dengan ID=2, maka url diisi : http://localhost:8000/api/mahasiswa/9</p>

```
GET http://localhost:8000/api/mahasiswa/9

Pretty Raw Preview Visualize JSON

1 {
2   "message": "Success!",
3   "values": [
4     {
5       "id": 9,
6       "nama": "dina lisuardi",
7       "nim": "1941723004",
8       "email": "lisuardidina@gmail.com",
9       "jurusan": "Teknik Informatika",
10      "created_at": null,
11      "updated_at": null
12    }
13  ]
14 }
```

Ketika mencoba menampilkan ID=2 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut.

```
GET http://localhost:8000/api/mahasiswa/2

Pretty Raw Preview Visualize JSON

1 {
2   "message": "Gagal!"
3 }
```

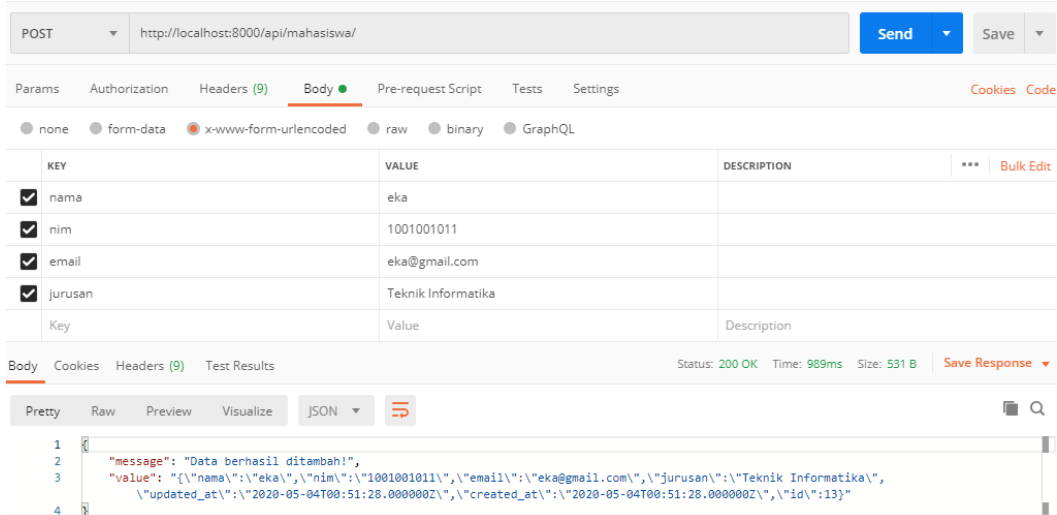
12

Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php

```
// fungsi tambah data
public function create(Request $request)
{
    $mhs = new Mahasiswa();
    $mhs->nama = $request->nama;
    $mhs->nim = $request->nim;
    $mhs->email = $request->email;
    $mhs->jurusan = $request->jurusan;

    //jika data berhasil tersimpan
    if($mhs->save()){
        $res['message'] = "Data berhasil ditambah!";
        $res['value'] = "$mhs";
        return response($res);
    }
}
```

Keterangan:

	<ul style="list-style-type: none">• Fungsi create menerima parameter Request yang menampung isian data mahasiswa yang akan ditambahkan ke database.• Line 55-59 : <code>\$mhs->save()</code> digunakan untuk menyimpan data ke database, apabila <code>save()</code> berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang ditambah.
13	<p>Tambahkan route untuk memanggil fungsi create pada routes/api.php</p> <pre>Route::post('/mahasiswa', 'MahasiswaController@create');</pre>
14	<p>Kita coba untuk menambahkan data melalui Postman.</p>  <ul style="list-style-type: none">• Isikan url : <code>http://localhost:8000/api/mahasiswa</code>. Karena kita ingin mengirim data ke database, maka perintah yang dipakai adalah ‘POST’.• Pilih tab Body dan pilih radio button x-www-form-urlencoded. Isikan nama kolom pada database pada KEY, untuk isian datanya tuliskan pada VALUE. <p>Kemudian coba untuk tampilkan semua data, kita lihat apakah data baru sudah masuk ke database.</p>

	<div data-bbox="349 205 1396 882"> <div>GET http://localhost:8000/api/mahasiswa/</div> <div> <div>Pretty Raw Preview Visualize JSON ↺</div> <pre> 25 "email": "lisuardidina@gmail.com", 26 "jurusan": "Teknik Informatika", 27 "created_at": null, 28 "updated_at": null 29 }, 30], 31 { 32 "id": 11, 33 "nama": "ade ismail", 34 "nim": "4667758692", 35 "email": "adeismail@gmail.com", 36 "jurusan": "Teknik Informatika", 37 "created_at": null, 38 "updated_at": null 39 }, 40 { 41 "id": 13, 42 "nama": "eka", 43 "nim": "1001001011", 44 "email": "eka@gmail.com", 45 "jurusan": "Teknik Informatika", 46 "created_at": "2020-05-04T00:51:28.000000Z", 47 "updated_at": "2020-05-04T00:51:28.000000Z" 48 } 49] 50 }</pre> </div> </div>
15	<p>Selanjutnya kita akan menambahkan fungsi untuk mengubah data dari database. Buat fungsi update pada MahasiswaController.php.</p>

```
//fungsi untuk mengubah data
public function update(Request $request, $id)
{
    $nama = $request->nama;
    $nim = $request->nim;
    $email = $request->email;
    $jurusan = $request->jurusan;

    $mhs = Mahasiswa::find($id);
    $mhs->nama = $nama;
    $mhs->nim = $nim;
    $mhs->email = $email;
    $mhs->jurusan = $jurusan;

    if ($mhs->save()) {
        $res['message'] = "Data berhasil diubah!";
        $res['value'] = "$mhs";
        return response($res);
    }
    else {
        $res['message'] = "Gagal!";
        return response($res);
    }
}
```

Keterangan:

- Fungsi update menerima parameter Request yang menampung isian data mahasiswa yang akan diubah dan parameter id yang menunjukkan ID yang dipilih.
- Line 70 : Mahasiswa::find(\$id) digunakan untuk pencarian data pada tabel mahasiswa berdasarkan \$id.
- Line 76-80 : \$mhs->save() digunakan untuk menyimpan perubahan data ke database, apabila save() berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang diubah.

16 Tambahkan route untuk memanggil fungsi update pada routes/api.php

```
Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');
```

Karena kita ingin memasukkan perubahan data, maka perintah yang dipakai adalah 'put'.

17 Sekarang kita coba untuk mengubah data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah PUT untuk mengubah data.

Berikut adalah contoh untuk mengubah data dengan ID=2, maka url diisi : <http://localhost:8000/api/mahasiswa/update/2>. Pilih tab Body dan pilih radio button x-www-form-urlencoded. Isikan nama kolom pada database pada KEY, untuk isian data yang diubah tuliskan pada VALUE.

The screenshot shows a REST client interface with a PUT request to `http://localhost:8000/api/mahasiswa/update/13`. The 'Body' tab is selected, and the 'x-www-form-urlencoded' radio button is chosen. A table lists the data to be updated:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	eka kartika	
<input checked="" type="checkbox"/> nim	1001001011	
<input checked="" type="checkbox"/> email	ekakartika@gmail.com	
<input checked="" type="checkbox"/> jurusan	Teknik Informatika	
Key	Value	Description

The response status is 200 OK. The JSON response body is:

```
{
  "message": "Data berhasil diubah!",
  "value": "{\n\"id\":13,\n\"nama\":\n\"eka kartika\", \n\"nim\":\n\"1001001011\", \n\"email\":\n\"ekakartika@gmail.com\", \n\"jurusan\":\n\"Teknik Informatika\", \n\"created_at\":\n\"2020-05-04T00:51:28.000000Z\", \n\"updated_at\":\n\"2020-05-04T01:03:53.000000Z\"}"
}
```

Akan muncul pesan berhasil serta perubahan data dari ID=13

Kemudian coba untuk menampilkan data dengan ID=2 untuk melihat apakah data sudah ter-update.

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/mahasiswa/13`. The 'Pretty' tab is selected, and the JSON response is displayed:

```
{
  "message": "Success!",
  "values": [
    {
      "id": 13,
      "nama": "eka kartika",
      "nim": "1001001011",
      "email": "ekakartika@gmail.com",
      "jurusan": "Teknik Informatika",
      "created_at": "2020-05-04T00:51:28.000000Z",
      "updated_at": "2020-05-04T01:03:53.000000Z"
    }
  ]
}
```

18 Terakhir kita akan membuat fungsi untuk menghapus data dengan nama delete di MahasiswaController.php.

	<pre data-bbox="349 195 1237 768"> //fungsi untuk menghapus data public function delete(\$id) { \$mhs = Mahasiswa::where('id', \$id); if (\$mhs->delete()) { \$res['message'] = "Data berhasil dihapus"; return response(\$res); } else { \$res['message'] = "Gagal!"; return response(\$res); } } </pre> <p data-bbox="349 779 500 810">Keterangan:</p> <ul data-bbox="394 821 1398 982" style="list-style-type: none"> • Fungsi delete menerima parameter id yang menunjukkan ID yang dipilih. • Line 92-99 : \$mhs->delete() digunakan untuk menghapus data dari database, apabila delete() berhasil dijalankan maka akan ditampilkan pesan berhasil.
19	<p data-bbox="349 1041 1230 1073">Tambahkan route untuk memanggil fungsi delete pada routes/api.php</p> <pre data-bbox="349 1077 1325 1199"> Route::delete('/mahasiswa/{id}', 'MahasiswaController@delete'); </pre> <p data-bbox="349 1209 1344 1241">Karena kita ingin menghapus data, maka perintah yang dipakai adalah 'delete.</p>
20	<p data-bbox="349 1262 1365 1335">Buka Postman untuk mencoba menghapus data dari database. Gunakan perintah DELETE untuk mengubah data.</p> <p data-bbox="349 1377 1321 1455">Berikut adalah contoh untuk menghapus data dengan ID=11, maka url diisi : http://localhost:8000/api/mahasiswa /11</p>

DELETE

http://localhost:8000/api/mahasiswa/11

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE
Key	Value

Body

Cookies

Headers (9)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

{

2

"message": "Data berhasil dihapus"

3

}

Muncul pesan berhasil ketika data terhapus dari database.

-- Selamat Mengerjakan --