

**LAPORAN PRAKTIKUM
PEMROGRAMAN WEB LANJUT**

Jobsheet - 14

Membuat RESTful API Laravel



Oleh:

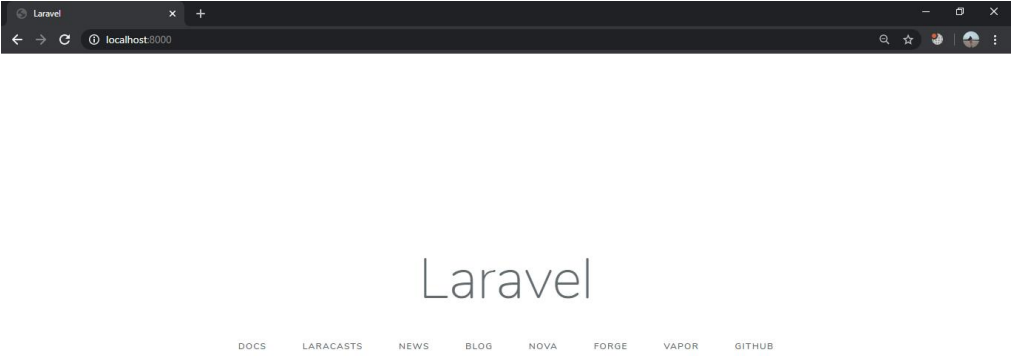
Farradila Ayu Damayanti

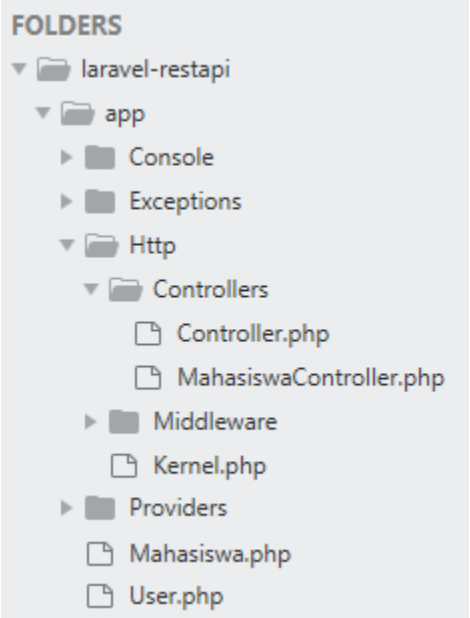
1841720074

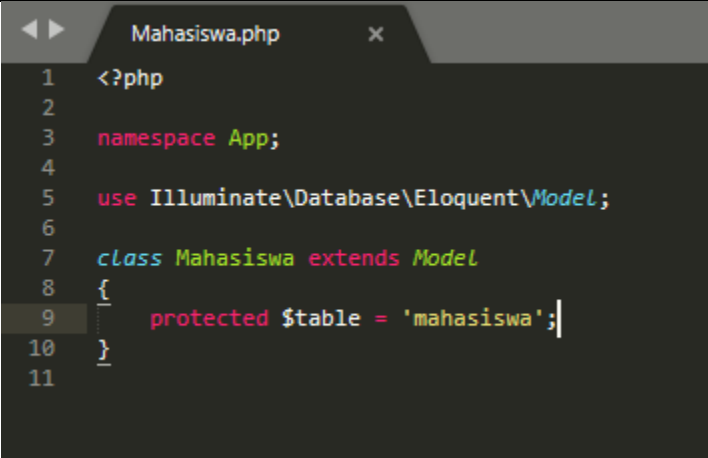
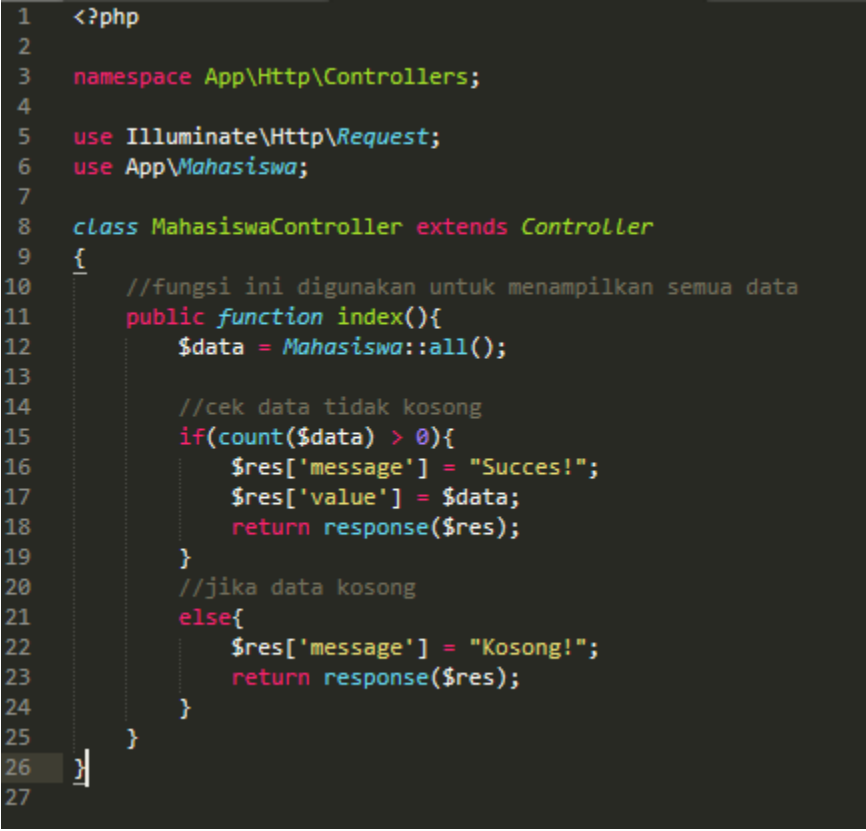
TI-2B

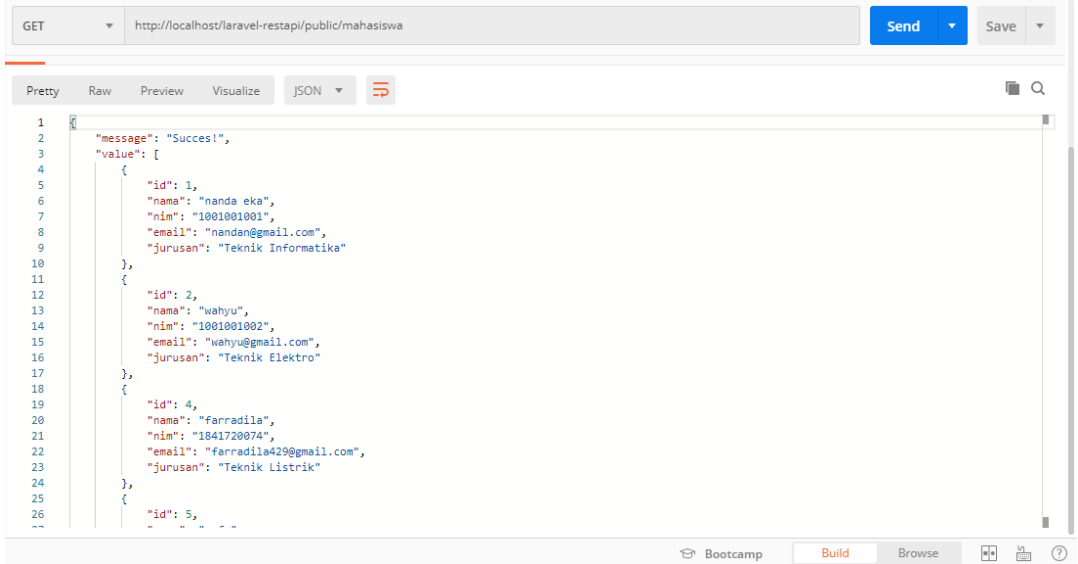
**PROGRAM STUDI D-IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2020**

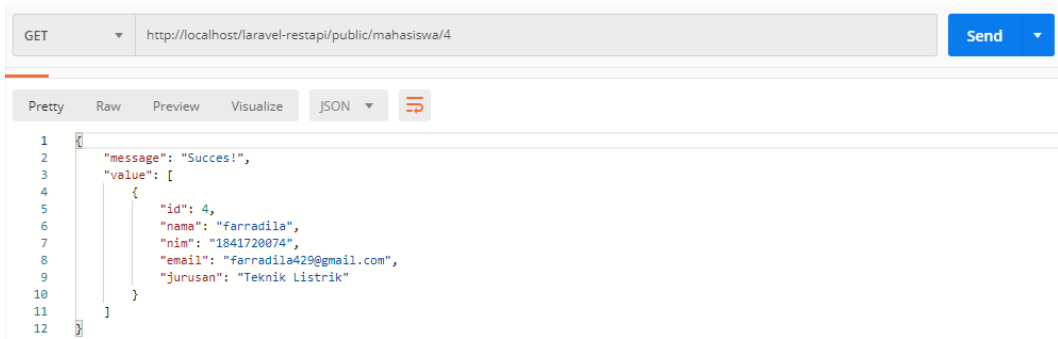
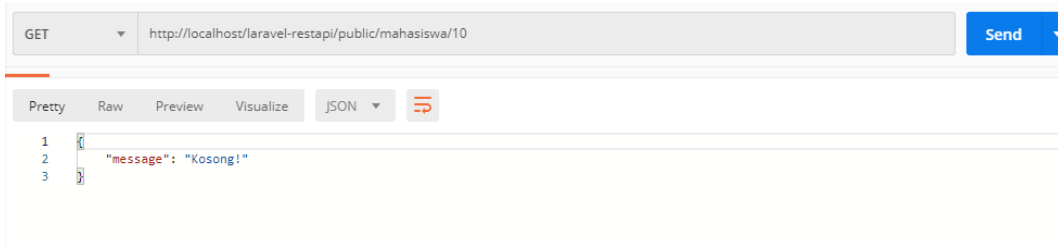
Praktikum: Membuat RESTful API di Laravel

Langkah	Keterangan
1	<p>Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut. cd C:\xampp\htdocs laravel new laravel-restapi</p> <pre>C:\Program Files\Ampms\www>laravel new laravel-restapi 'laravel' is not recognized as an internal or external command, operable program or batch file. C:\Program Files\Ampms\www>composer create-project --prefer-dist laravel/laravel laravel-restapi Creating a "laravel/laravel" project at "./laravel-restapi" Installing laravel/laravel (v7.6.0) - Installing laravel/laravel (v7.6.0): Downloading (100%) Created project in C:\Program Files\Ampms\www\laravel-restapi > @php -r "file_exists('.env') copy('.env.example', '.env');" Loading composer repositories with package information Updating dependencies (including require-dev) Package operations: 92 installs, 0 updates, 0 removals - 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 (v4.1.5): Downloading (100%) - Installing symfony/css-selector (v5.0.8): 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 (v5.0.8): Downloading (100%)</pre>
2	<p>Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut. cd C:\laravel-restapi php artisan serve.</p> <p>Akan tampil halaman default Laravel seperti di bawah ini.</p> 
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>

	<pre> 8 9 DB_CONNECTION=mysql 10 DB_HOST=127.0.0.1 11 DB_PORT=3306 12 DB_DATABASE=latihan_laravel 13 DB_USERNAME=root 14 DB_PASSWORD=mysql 15 </pre>
4	<p>Buat model dengan nama Mahasiswa, buat juga controllernya. Untuk membuat model dan controllernya sekaligus tuliskan perintah berikut pada command prompt (terlebih dahulu keluar dari php artisan serve dengan mengetik ctrl+C pada keyboard)</p> <p>php artisan make:model Mahasiswa -c</p> <pre> C:\Program Files\Ampss\www\laravel-restapi>php artisan make:model Mahasiswa -c Model created successfully. Controller created successfully. C:\Program Files\Ampss\www\laravel-restapi> </pre> <p>Sehingga pada project laravel-restapi akan bertambah dua file yaitu model Mahasiswa.php serta controller MahasiswaController.php.</p> 
5	<p>Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.</p>

	 <pre> 1 <?php 2 3 namespace App; 4 5 use Illuminate\Database\Eloquent\Model; 6 7 class Mahasiswa extends Model 8 { 9 protected static \$table = 'mahasiswa'; 10 } 11 </pre>
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 ini digunakan untuk menampilkan semua data 11 public function index(){ 12 \$data = Mahasiswa::all(); 13 14 //cek data tidak kosong 15 if(count(\$data) > 0){ 16 \$res['message'] = "Sukses!"; 17 \$res['value'] = \$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>
7	<p>Tambahkan route untuk memanggil fungsi index pada file routes/api.php (Line 21).</p>

	<pre> 18 19 Route::middleware('auth:api')->get('/user', function (Request \$request){ 20 return \$request->user(); 21 }); 22 23 Route::get('mahasiswa', 'MahasiswaController@index'); </pre>
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": "Sukses!", 3 "value": [4 { 5 "id": 1, 6 "name": "nanda eka", 7 "nim": "1001001001", 8 "email": "nandan@gmail.com", 9 "jurusan": "Teknik Informatika" 10 }, 11 { 12 "id": 2, 13 "name": "wahyu", 14 "nim": "1001001002", 15 "email": "wahyu@gmail.com", 16 "jurusan": "Teknik Elektro" 17 }, 18 { 19 "id": 4, 20 "name": "farradila", 21 "nim": "1841720074", 22 "email": "farradila429@gmail.com", 23 "jurusan": "Teknik Listrik" 24 }, 25 { 26 "id": 5, 27 ... </pre>
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 kosong 39 else{ 40 \$res['message'] = "Gagal!"; 41 return response(\$res); 42 } 43 } 44 </pre>

10	<p>Tambahkan route untuk memanggil fungsi getId pada routes/api.php</p> <pre> 22 23 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId'); </pre>
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/laravel-restapi/public/mahasiswa/4</p>  <p>Ketika mencoba menampilkan ID=10 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut.</p> 
12	<p>Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php.</p>

```

26
27 ▼ public function create(Request $request){
28     $mhs = new Mahasiswa();
29     $mhs->nama = $request->nama;
30     $mhs->nim = $request->nim;
31     $mhs->email = $request->email;
32     $mhs->jurusan = $request->jurusan;
33
34 ▼     if($mhs->save()){
35         $res['message'] = "Data berhasil ditambah!";
36         $res['value'] = "$mhs";
37         return response($res);
38     }
39
40 }

```

13

Tambahkan route untuk memanggil fungsi create pada routes/api.php

```

24
25 Route::post('/mahasiswa','MahasiswaController@create');|

```

14

Kita coba untuk menambahkan data melalui Postman.

POST http://localhost/laravel-restapi/public/api/mahasiswa

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

Body Cookies (2) Headers (10) Test Results Status: 200 OK Time: 20.85 s Size: 566 B Save Response

```

1
2 {
3   "message": "Data berhasil ditambah!",
4   "values": {
5     "nama": "khilda",
6     "nim": "1001001011",
7     "email": "khilda@gmail.com",
8     "jurusan": "Teknik Informatika",
9     "updated_at": "2020-05-03T23:48:12.000000Z",
10    "created_at": "2020-05-03T23:48:12.000000Z",
11    "id": 8
12  }
13 }

```

Kemudian coba untuk tampilkan semua data, kita lihat apakah data baru sudah masuk ke database.

	 <pre> GET http://localhost/laravel-restapi/public/api/mahasiswa Pretty Raw Preview Visualize JSON [{ "id": 7, "nama": "ade ismail", "nim": "0804030028", "email": "adeismail@gmail.com", "jurusan": "Teknik Informatika", "created_at": "-000001-11-30T00:00:00.000000Z", "updated_at": "-000001-11-30T00:00:00.000000Z" }, { "id": 8, "nama": "khilda", "nim": "1001001011", "email": "khilda@gmail.com", "jurusan": "Teknik Informatika", "created_at": "2020-05-03T23:48:12.000000Z", "updated_at": "2020-05-03T23:48:12.000000Z" }] </pre>
15	<p>Selanjutnya kita akan menambahkan fungsi untuk mengubah data dari database. Buat fungsi update pada MahasiswaController.php.</p> <pre> 62 //fungsi untuk mengubah data 63 public function update(Request \$request, \$id) 64 { 65 \$nama = \$request->nama; 66 \$nim = \$request->nim; 67 \$email = \$request->email; 68 \$jurusan = \$request->jurusan; 69 70 \$mhs = Mahasiswa::find(\$id); 71 \$mhs->nama = \$nama; 72 \$mhs->nim = \$nim; 73 \$mhs->email = \$email; 74 \$mhs->jurusan = \$jurusan; 75 76 if (\$mhs->save()) { 77 \$res['message'] = "Data berhasil diubah!"; 78 \$res['values'] = \$mhs; 79 return response(\$res); 80 }else{ 81 \$res['message'] = "Gagal!"; 82 return response(\$res); 83 } 84 } </pre>
16	<p>Tambahkan route untuk memanggil fungsi update pada routes/api.php</p>

	<pre> 26 27 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update'); 28 </pre>
17	<p>Sekarang kita coba untuk mengubah data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah PUT untuk mengubah data.</p> <p>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.</p>  <pre> 1 { 2 "message": "Data berhasil diubah!", 3 "values": { 4 "id": 8, 5 "nama": "khilda", 6 "nim": "1001001011", 7 "email": "khilda@gmail.com", 8 "jurusan": "Teknik Elektro", 9 "created_at": "2020-05-03T23:48:12.000000Z", 10 "updated_at": "2020-05-03T23:59:42.000000Z" 11 } 12 } </pre>
18	<p>Terakhir kita akan membuat fungsi untuk menghapus data dengan nama delete di MahasiswaController.php.</p> <pre> 86 //fungsi untuk menghapus data 87 public function delete(\$id) 88 { 89 \$mhs = Mahasiswa::where('id',\$id); 90 91 if (\$mhs->delete()) { 92 \$res['message'] = "Data berhasil dihapus!"; 93 return response(\$res); 94 }else{ 95 \$res['message'] = "Gagal!"; 96 return response(\$res); 97 } 98 } 99 </pre>
19	Tambahkan route untuk memanggil fungsi delete pada routes/api.php

```
28  
29 Route::delete('/mahasiswa/{id}','MahasiswaController@delete');
```

20

Buka Postman untuk mencoba menghapus data dari database. Gunakan perintah DELETE untuk mengubah data.

