

**LAPORAN PRAKTIKUM
PEMROGRAMAN WEB LANJUT**

Jobsheet - 14

Membuat RESTful API Laravel



Disusun oleh:
Livia Yurike Khuril Maula
D4 TI – 2A / 16
NIM : 1841720025

**POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI D4 TEKNIK INFORMATIKA**

MEI 2020

Topik

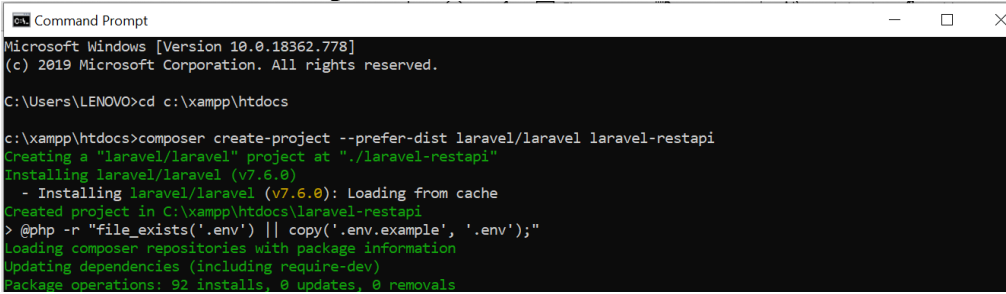
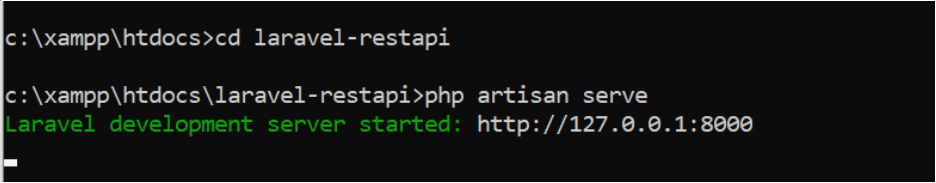
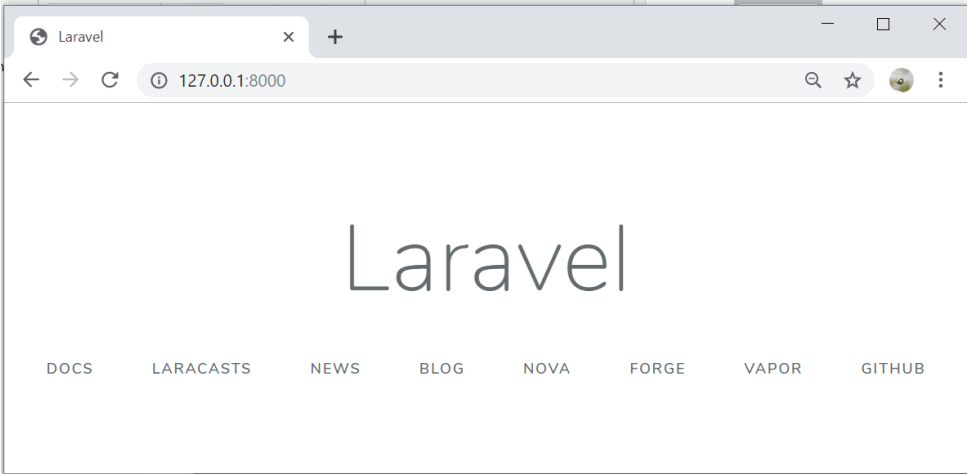
Membuat RESTful API dengan Framework Laravel

Tujuan

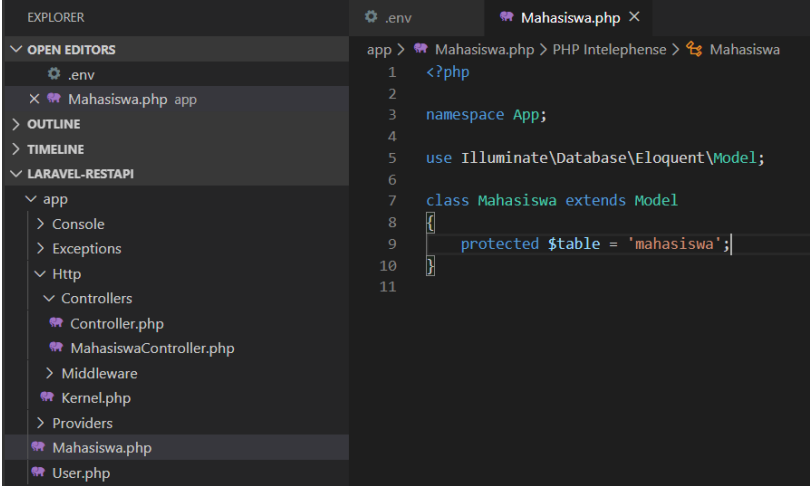
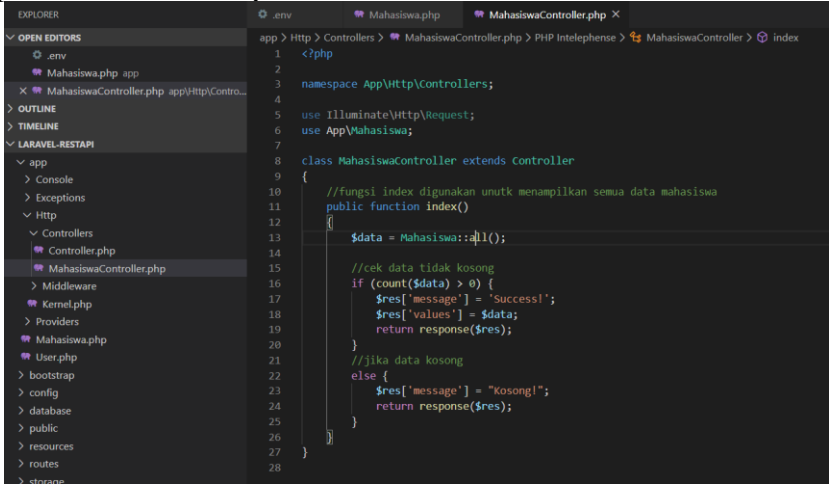
Mahasiswa diharapkan dapat:

1. Memahami bagaimana cara membuat RESTful API menggunakan Laravel

Praktikum: Membuat RESTful API di Laravel

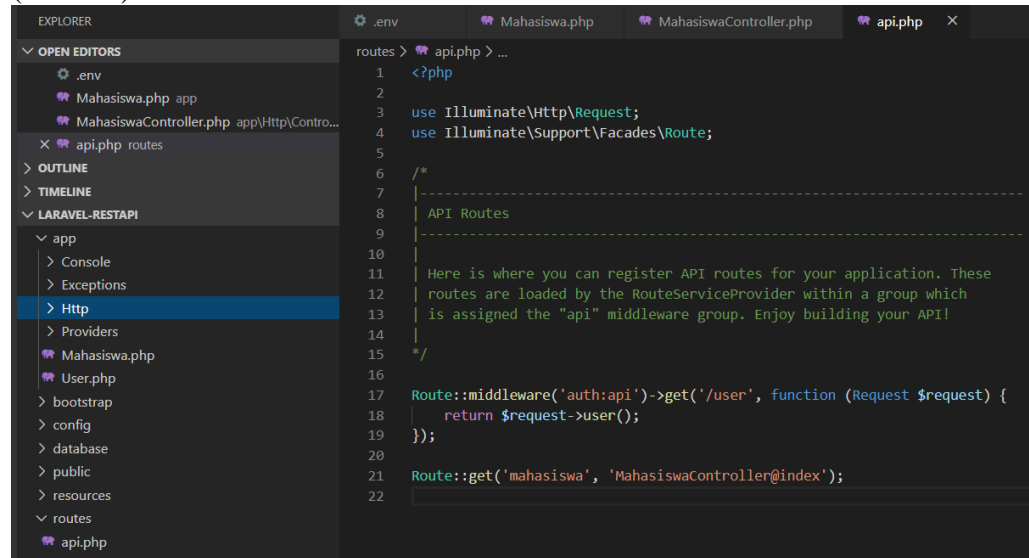
Langkah	Keterangan
1	<p>Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut.</p> <p>cd C:\xampp\htdocs laravel new laravel-restapi</p> 
2	<p>Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut.</p> <p>cd C:\laravel-restapi php 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> .env 1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:hlZtnBK/AwrMpAgkQku99BXYIvfE54wpiOTwgfeoZs= 4 APP_DEBUG=true 5 APP_URL=http://localhost 6 7 LOG_CHANNEL=stack 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= </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:\xampp\htdocs\laravel-restapi>php artisan make:model Mahasiswa -c Model created successfully. Controller created successfully. c:\xampp\htdocs\laravel-restapi> </pre> <p>Keterangan :</p> <ul style="list-style-type: none"> • -c merupakan perintah untuk menyertakan pembuatan controller <p>Sehingga pada project laravel-restapi akan bertambah dua file yaitu model Mahasiswa.php serta controller MahasiswaController.php.</p> <pre> app ├── Console ├── Exceptions ├── Http │ ├── Controllers │ │ ├── Controller.php │ │ └── MahasiswaController.php │ ├── Middleware │ ├── Kernel.php │ ├── Providers │ ├── Mahasiswa.php │ └── User.php </pre>

5	<p>Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.</p>  <p>Keterangan:</p> <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.</p> <p>Pertama, kita akan mengubah fungsi index agar saat fungsi index dipanggil, maka aplikasi akan menampilkan seluruh data dari tabel mahasiswa.</p>  <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 Variabel \$array[values] akan menyimpan semua baris data pada tabel mahasiswa

7

Tambahkan route untuk memanggil fungsi index pada file **routes/api.php** (Line 21).



```

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

```

8

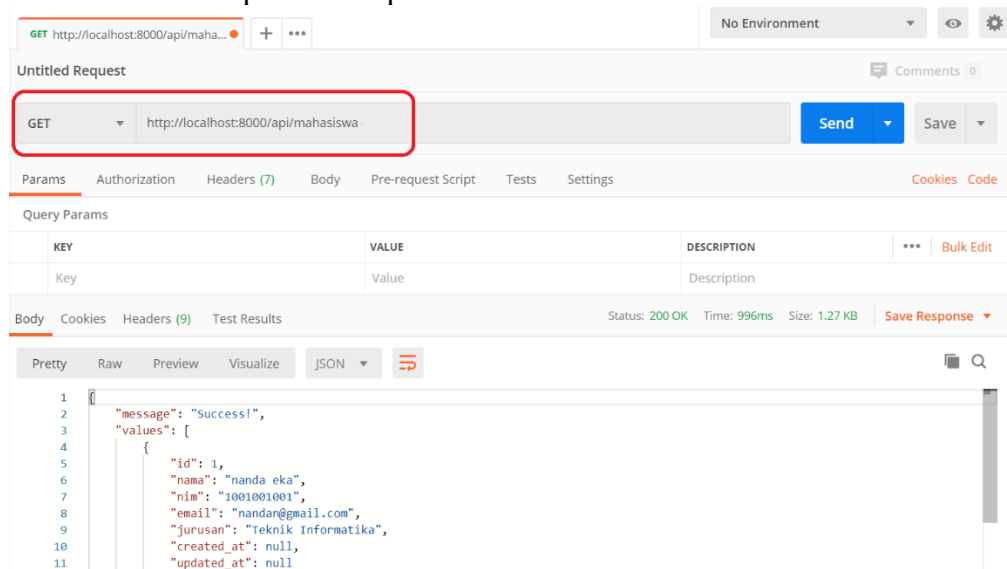
Ketikkan perintah **php artisan serve** pada command prompt. Lalu kita coba menguji fungsi untuk menampilkan data menggunakan aplikasi **Postman**.

```

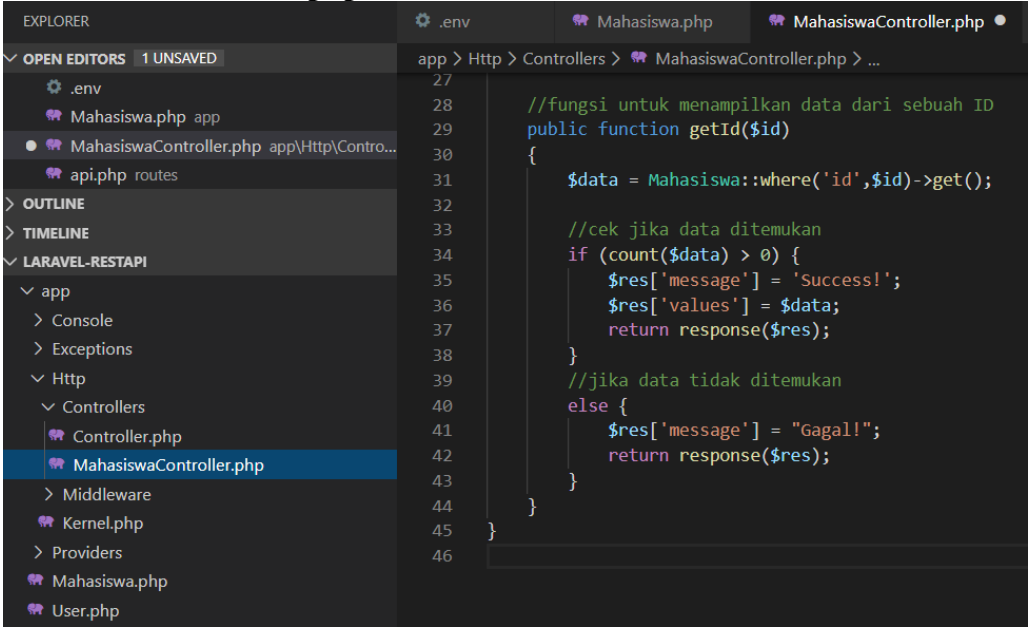
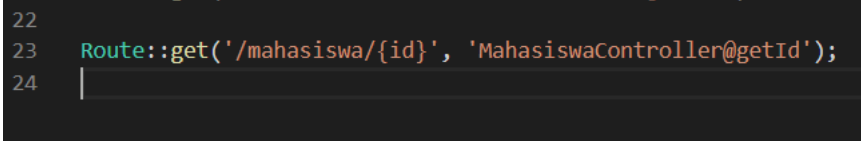
c:\xampp\htdocs\laravel-restapi>php artisan serve
Laravel development server started: http://127.0.0.1:8000

```

Gunakan perintah **GET**, isikan url : **http://localhost:8000/api/mahasiswa**
Berikut adalah tampilan dari aplikasi Postman.



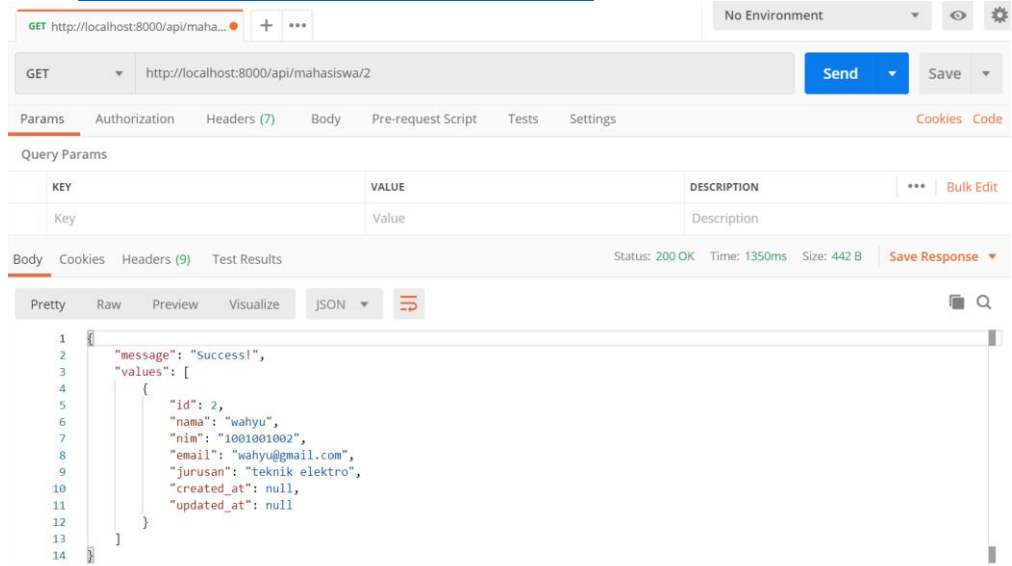
Semua data pada tabel mahasiswa akan tampil, ditampilkan juga pesan sukses.

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 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 //fungsi untuk menampilkan data dari sebuah ID public function getId(\$id) { \$data = Mahasiswa::where('id',\$id)->get(); //cek jika data ditemukan if (count(\$data) > 0) { \$res['message'] = 'Success!'; \$res['values'] = \$data; return response(\$res); } //jika data tidak ditemukan else { \$res['message'] = "Gagal!"; return response(\$res); } } </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> 22 23 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId'); 24 </pre> <p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'</p>

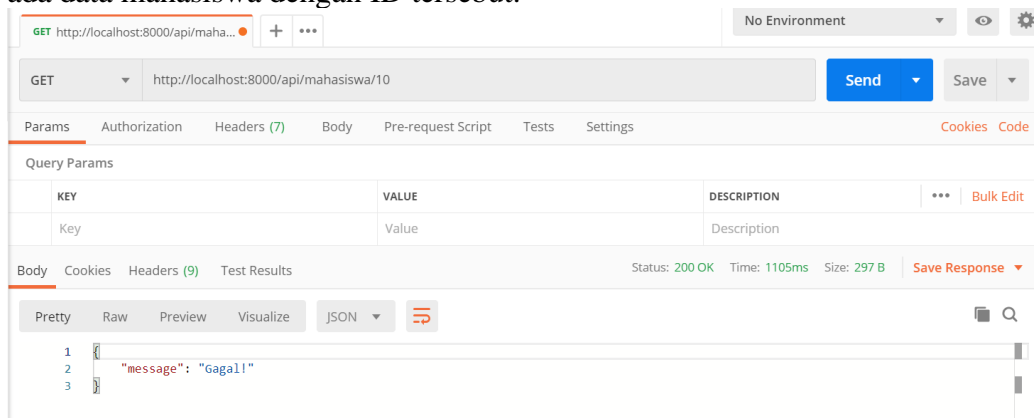
11

Sekarang kita coba untuk menampilkan data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah **GET** untuk menampilkan data.

Di bawah ini adalah contoh untuk menampilkan data dengan ID=2, maka url diisi : <http://localhost:8000/api/mahasiswa/2>

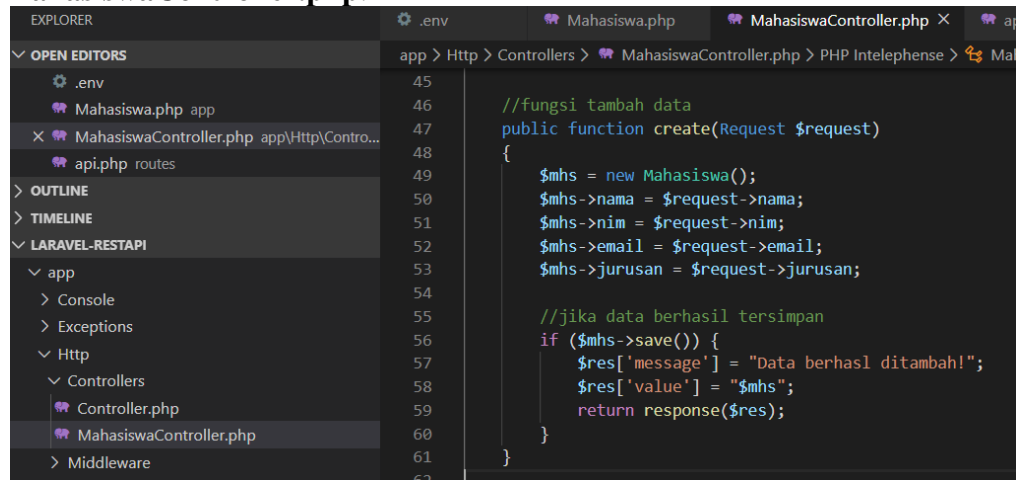


Ketika mencoba menampilkan ID=10 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut.



12

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



```

45
46
47 //fungsi tambah data
48 public function create(Request $request)
49 {
50     $mhs = new Mahasiswa();
51     $mhs->nama = $request->nama;
52     $mhs->nim = $request->nim;
53     $mhs->email = $request->email;
54     $mhs->jurusan = $request->jurusan;
55
56     //jika data berhasil tersimpan
57     if ($mhs->save()) {
58         $res['message'] = "Data berhasil ditambah!";
59         $res['value'] = "$mhs";
60         return response($res);
61     }
62

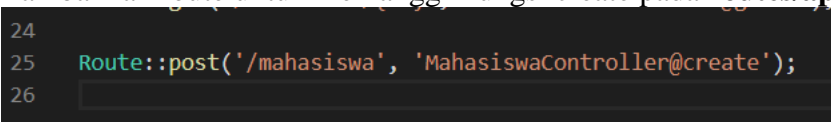
```

Keterangan:

- Fungsi create menerima parameter Request yang menampung isian data mahasiswa yang akan ditambahkan ke database.
- Line 55-59 : \$mhs->save() digunakan untuk menyimpan data ke database, apabila save() berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang ditambah.

13

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



```

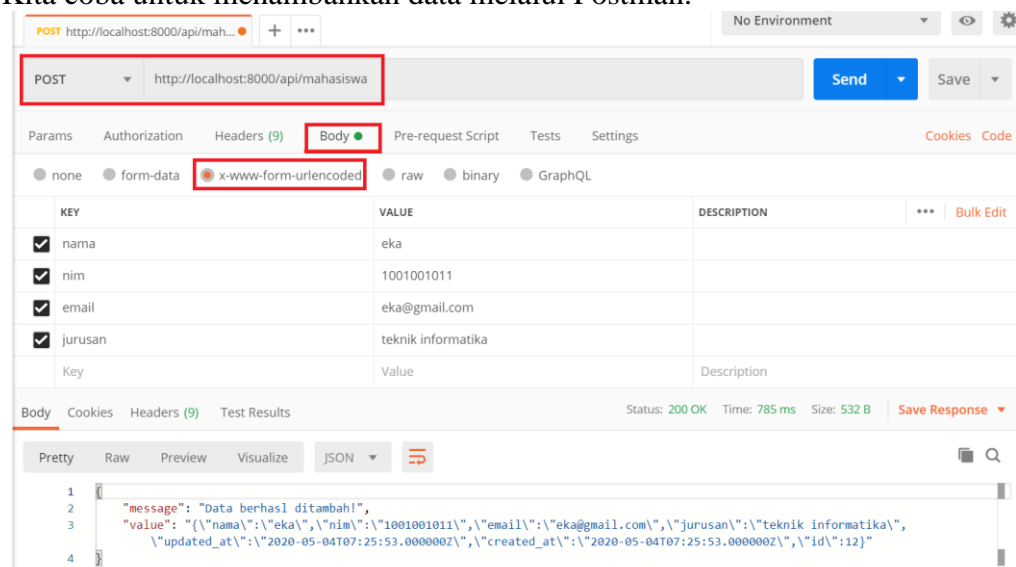
24
25 Route::post('/mahasiswa', 'MahasiswaController@create');
26

```

Karena kita ingin menambah data, maka perintah yang dipakai adalah 'post'.

14

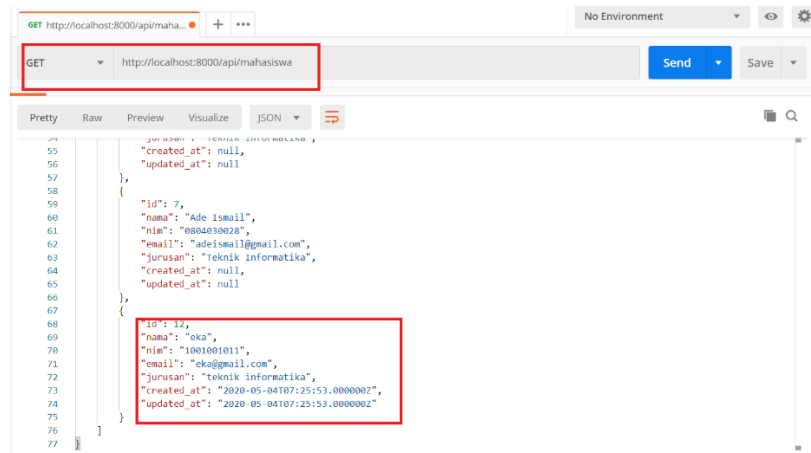
Kita coba untuk menambahkan data melalui Postman.



Isikan url : **http://localhost:8000/api/mahasiswa**. Karena kita ingin mengirim data ke database, maka perintah yang dipakai adalah **‘POST’**.

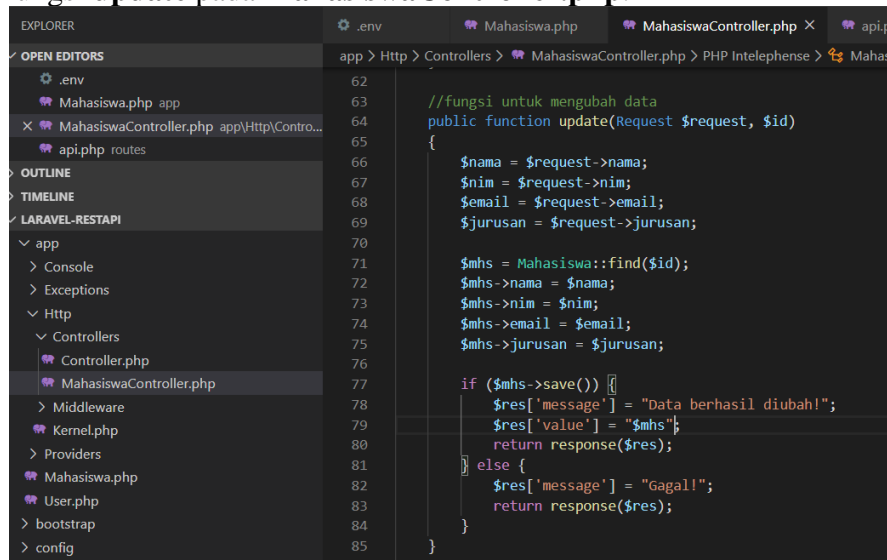
- Pilih tab **Body** dan pilih radio button **x-www-form-urlencoded**. Isikan nama kolom pada database pada **KEY**, untuk isian datanya tuliskan pada **VALUE**.

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



15

Selanjutnya kita akan menambahkan fungsi untuk mengubah data dari database. Buat fungsi **update** pada **MahasiswaController.php**.



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**

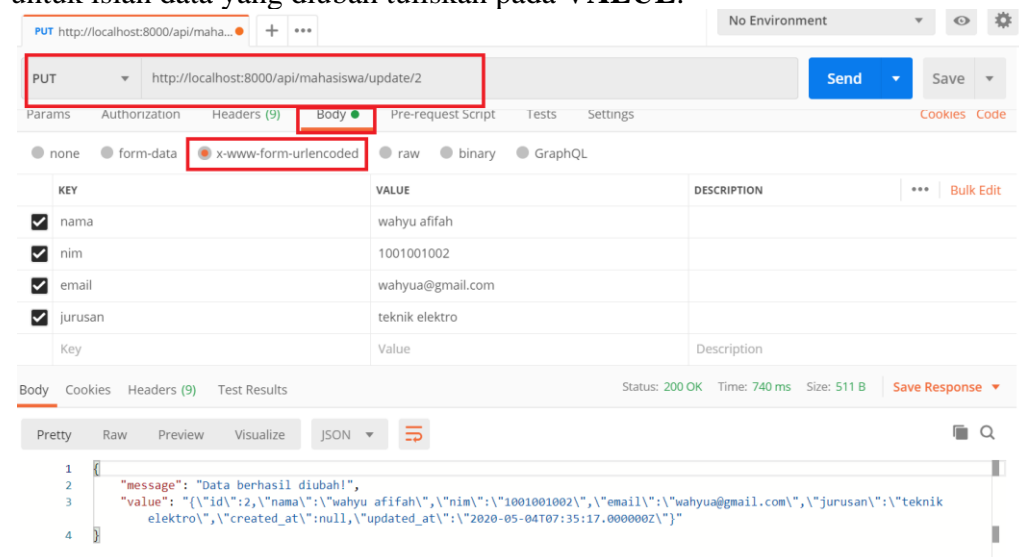
```
26
27 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');
28
```

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

17

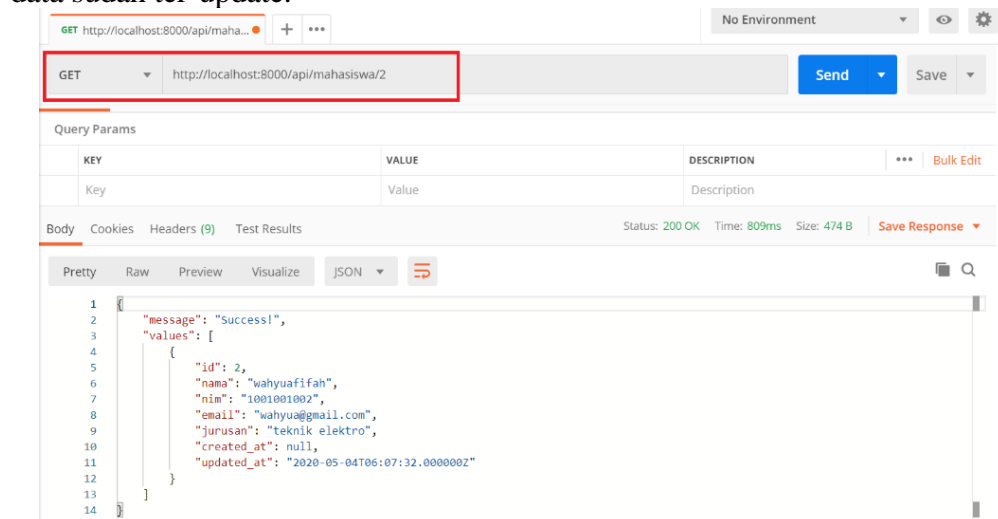
Sekarang kita coba untuk mengubah data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah **PUT** untuk mengubah data.

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**.



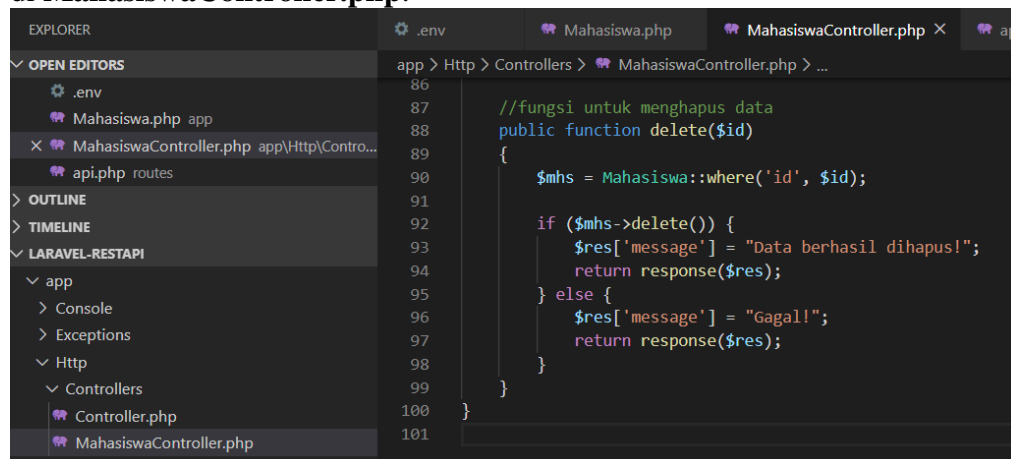
Akan muncul pesan berhasil serta perubahan data dari ID=2.

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



18

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



```

86
87 //fungsi untuk menghapus data
88 public function delete($id)
89 {
90     $mahasiswa = Mahasiswa::where('id', $id);
91
92     if ($mahasiswa->delete()) {
93         $res['message'] = "Data berhasil dihapus!";
94         return response($res);
95     } else {
96         $res['message'] = "Gagal!";
97         return response($res);
98     }
99 }
100
101

```

Keterangan:

- Fungsi **delete** menerima parameter **id** yang menunjukkan ID yang dipilih.
- Line 92-99 : `$mahasiswa->delete()` digunakan untuk menghapus data dari database, apabila `delete()` berhasil dijalankan maka akan ditampilkan pesan berhasil.

19

Tambahkan route untuk memanggil fungsi delete pada **routes/api.php**

```

28
29 Route::delete('/mahasiswa/{id}', 'MahasiswaController@delete');
30

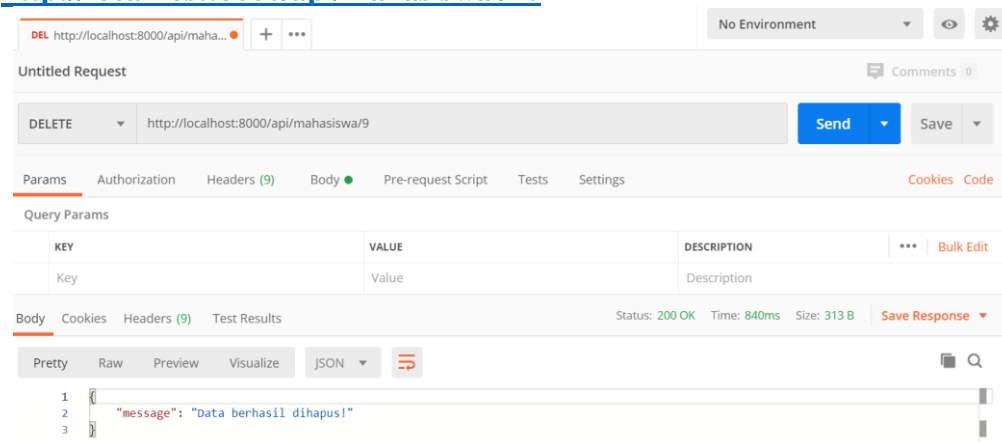
```

Karena kita ingin menghapus data, maka perintah yang dipakai adalah 'delete'.

20

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

Berikut adalah contoh untuk menghapus data dengan ID=10, maka url diisi : <http://localhost:8000/api/mahasiswa/10>



Muncul pesan berhasil ketika data terhapus dari database.