

LAPORAN PRAKTIKUM
PEMROGRAMAN WEB LANJUT

Jobsheet-14

RestApi Laravel



NAMA : Subhan Indra Prayoga
NIM : 1841720182
KELAS : 2B

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

2020



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

Topik

Membuat RESTful API dengan Framework Laravel

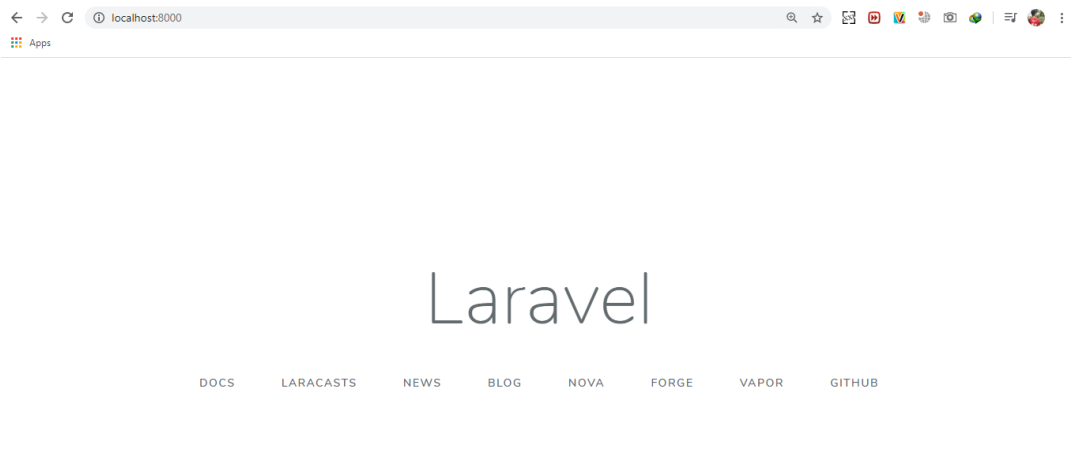
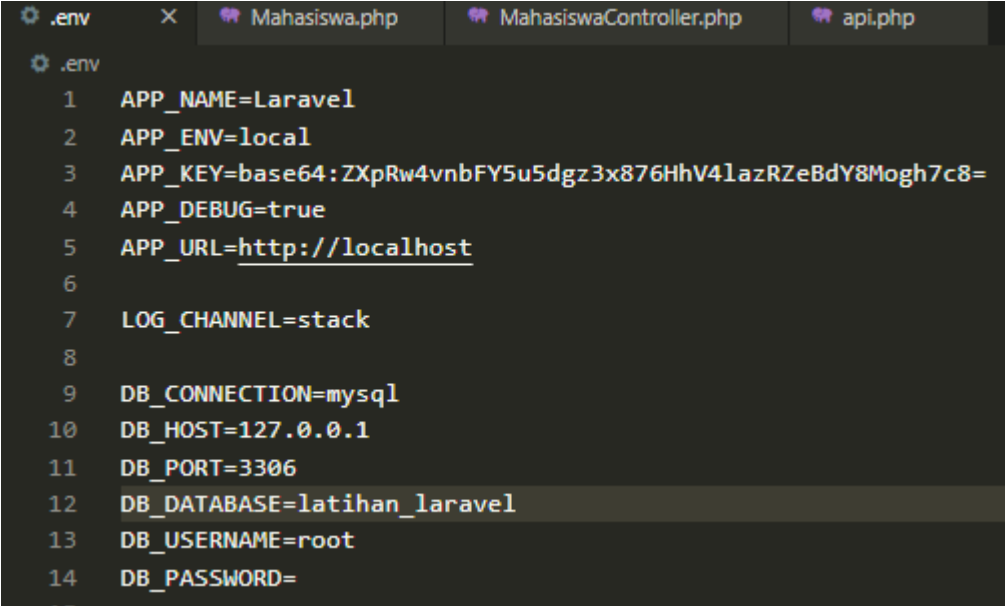
Tujuan

Mahasiswa diharapkan dapat:

1. Memahami bagaimana cara membuat RESTful API menggunakan Laravel

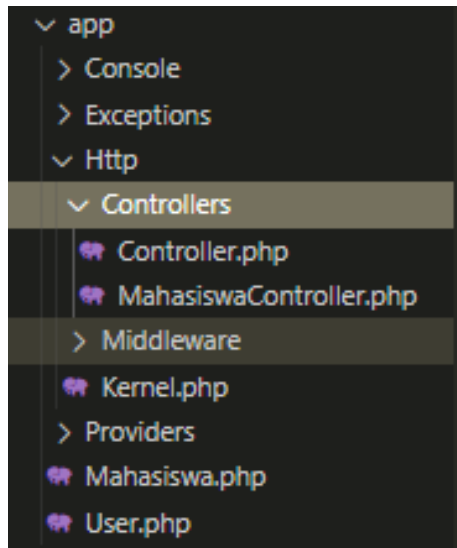
Praktikum: Membuat RESTful API di Laravel

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> <pre>C:\Users\De_Hansingingslasher>cd C:\xampp\htdocs C:\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</pre>
2	<p>Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut. cd C:\laravel-restapi php artisan serve</p> <pre>C:\xampp\htdocs>cd laravel-restapi C:\xampp\htdocs\laravel-restapi>php artisan serve Laravel development server started: http://127.0.0.1:8000</pre>

	 <p>Akan tampil halaman default Laravel seperti di bawah ini.</p>
3	<p>Kemudian lakukan konfigurasi <i>database</i> 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> 1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:ZXpRw4vnbFY5u5dgz3x876HhV41azRZeBdY8Mogh7c8= 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 <i>command prompt</i> (terlebih dahulu keluar dari php artisan serve dengan mengetik ctrl+C pada keyboard)</p> <p>php artisan make:model Mahasiswa -c</p> <p>Keterangan :</p> <p>-c merupakan perintah untuk menyertakan pembuatan <i>controller</i></p>

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

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.

```
.env  Mahasiswa.php
app > Mahasiswa.php
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
```

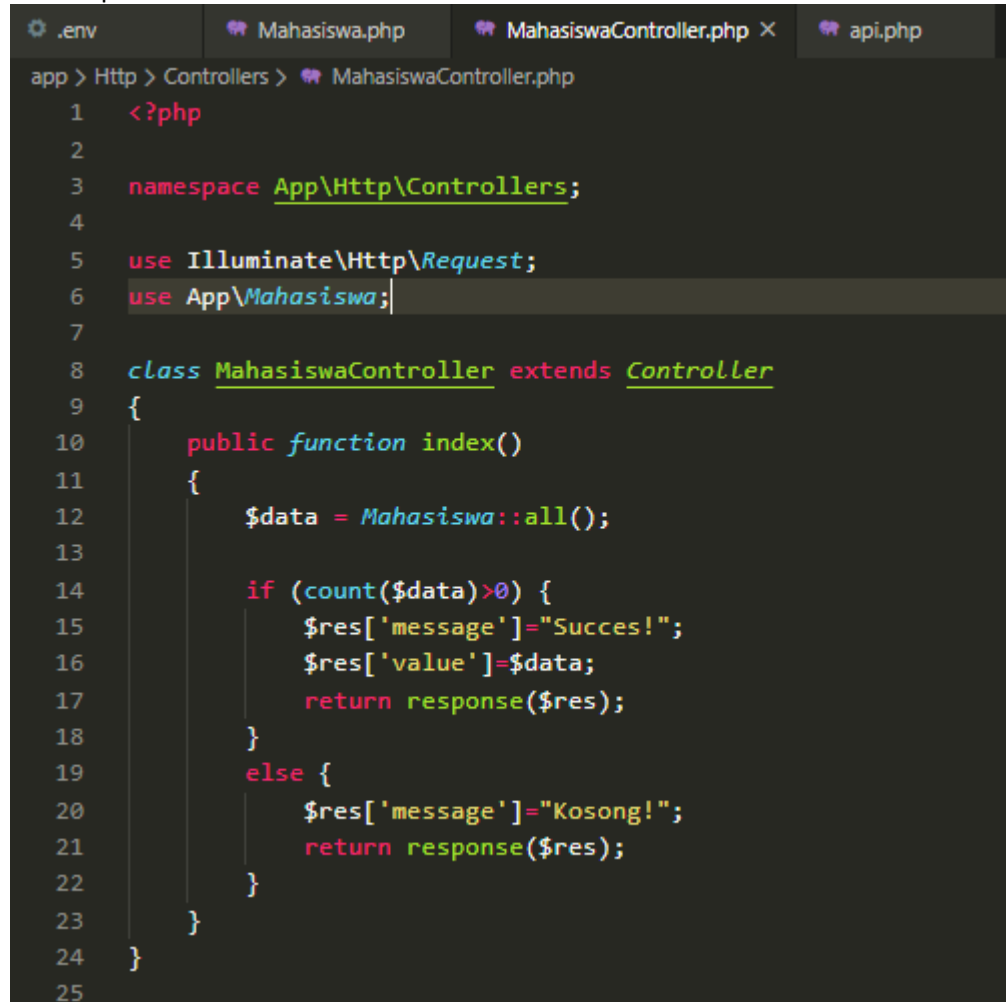
Keterangan:

Model ini akan mengelola tabel "mahasiswa" yang terdapat pada database latihan_laravel

6

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.



```
.env  Mahasiswa.php  MahasiswaController.php X  api.php
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         $data = Mahasiswa::all();
13
14         if (count($data)>0) {
15             $res['message']="Sukses!";
16             $res['value']=$data;
17             return response($res);
18         }
19         else {
20             $res['message']="Kosong!";
21             return response($res);
22         }
23     }
24 }
25
```

Keterangan:

- 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

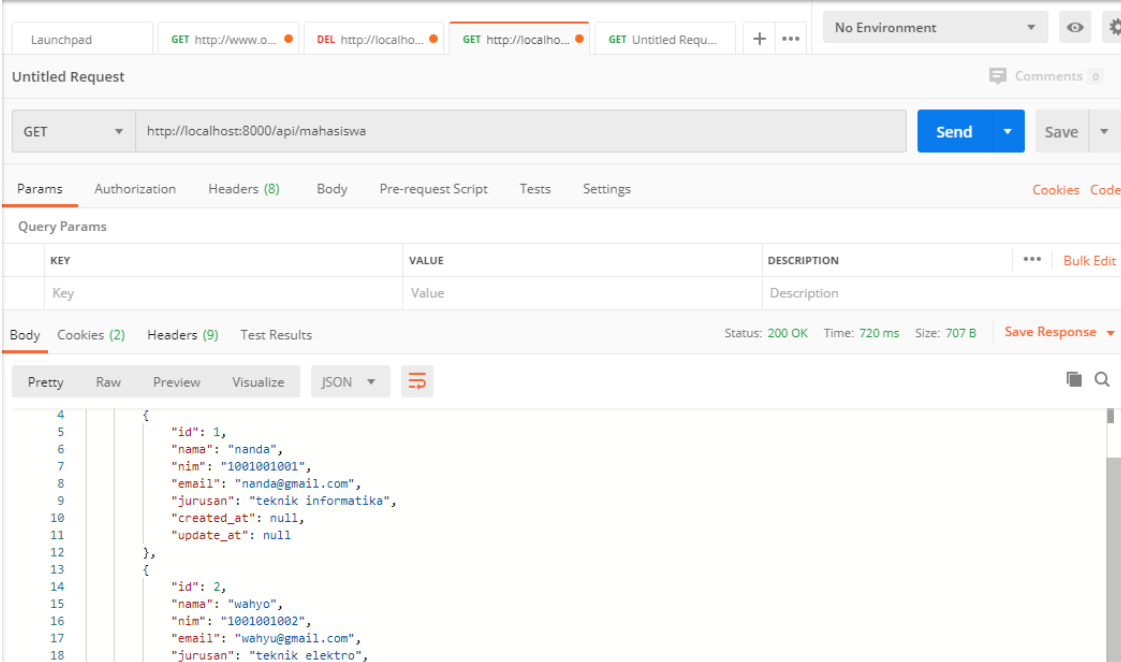
- 7 Tambahkan route untuk memanggil fungsi index pada file routes/api.php (Line 21).

```
.env  Mahasiswa.php  MahasiswaController.php  api.php x
routes > api.php
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
```

Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'.

- 8 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.

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



The screenshot shows the Postman interface with a GET request to `http://localhost:8000/api/mahasiswa` sent successfully. The response is a JSON array containing two student records.

KEY	VALUE	DESCRIPTION
id	1	
nama	"nanda"	
nim	"1001001001"	
email	"nanda@gmail.com"	
jurusan	"teknik informatika"	
created_at	null	
update_at	null	

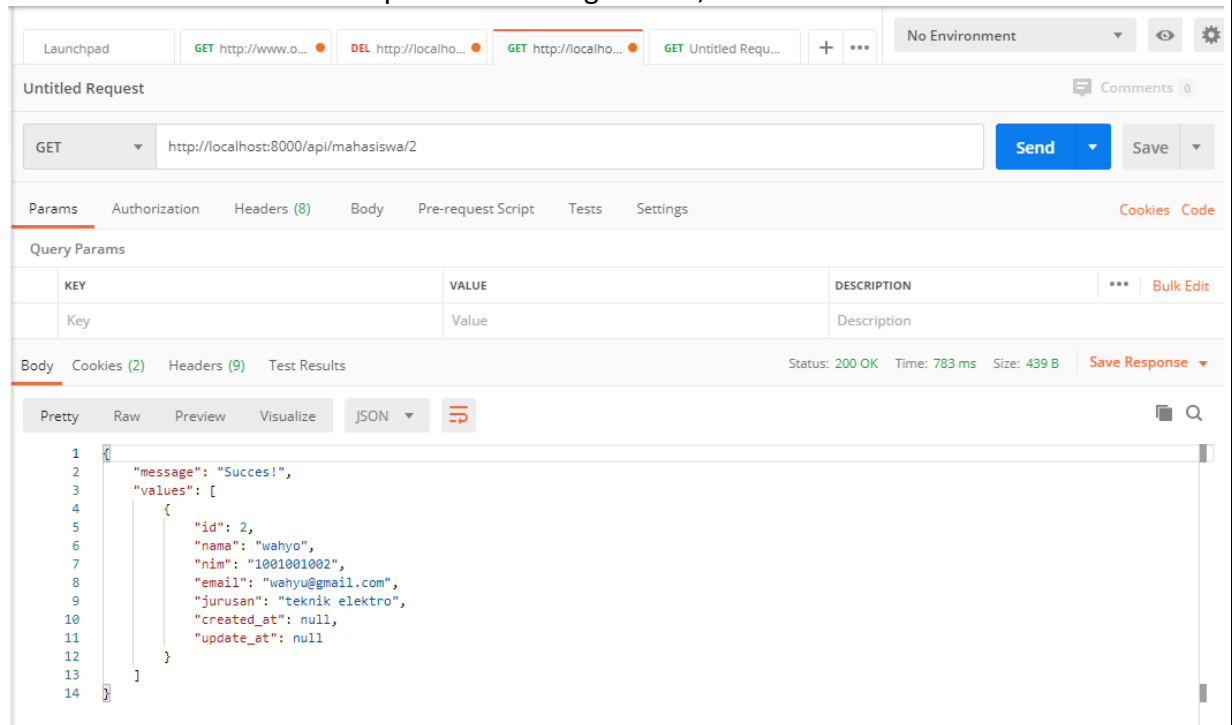
KEY	VALUE	DESCRIPTION
id	2	
nama	"wahyo"	
nim	"1001001002"	
email	"wahyu@gmail.com"	
jurusan	"teknik elektro"	

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</p> <pre> 25 public function getId(\$id) 26 { 27 \$data=Mahasiswa::where('id',\$id)->get(); 28 29 if (count(\$data)>0) { 30 \$res['message']="Sukses!"; 31 \$res['values']=\$data; 32 return response(\$res); 33 } 34 35 else { 36 \$res['message']="Gagal!"; 37 return response(\$res); 38 } 39 } 40 } 41 </pre> <p>tertentu. Buat fungsi baru yaitu getId pada MahasiswaController.php.</p> <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 <i>route</i> 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>

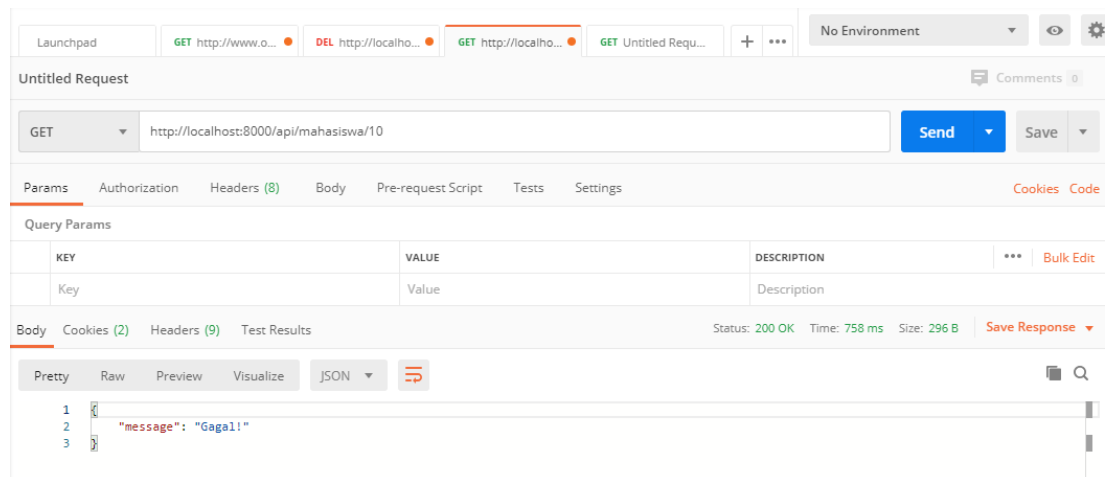
11

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**.


```

41     public function create(Request $request)
42     {
43         $mhs = new Mahasiswa();
44         $mhs->nama=$request->nama;
45         $mhs->nim=$request->nim;
46         $mhs->email=$request->email;
47         $mhs->jurusan=$request->jurusan;
48
49         if ($mhs->save()) {
50             $res['message']="Data Berhasil ditambahkan!";
51             $res['values']=$mhs;
52             return response($res);
53         }
54     }
55 }
56

```

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`.

Launchpad GET http://www.omdbapi... DEL http://localhost/UTS/... POST http://localhost:8000...

POST http://localhost:8000/api/mahasiswa/ Send Save

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	Bambang	
<input checked="" type="checkbox"/> nim	1841720123	
<input checked="" type="checkbox"/> email	bambang@gmail.com	
<input checked="" type="checkbox"/> jurusan	Teknik Industri	
Key	Value	Description

Body Cookies (2) Headers (9) Test Results Status: 200 OK Time: 870 ms Size: 511 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "message": "Data Berhasil ditambahkan!",
3    "value": {
4      "nama": "Bambang",
5      "nim": "1841720123",
6      "email": "bambang@gmail.com",
7      "jurusan": "Teknik Industri",
8      "updated_at": "2020-05-03T19:47:49.000000Z",
9      "created_at": "2020-05-03T19:47:49.000000Z",
10     "id": 9
11   }
12 }

```

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.

Launchpad GET http://www.omdbapi... DEL http://localhost/UTS/... GET http://localhost:8000...

GET http://localhost:8000/api/mahasiswa/9 Send Save

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	Bambang	
<input checked="" type="checkbox"/> nim	1841720123	
<input checked="" type="checkbox"/> email	bambang@gmail.com	
<input checked="" type="checkbox"/> jurusan	Teknik Industri	
Key	Value	Description

Body Cookies (2) Headers (9) Test Results Status: 200 OK Time: 804 ms Size: 495 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "message": "Sukses!",
3    "values": [
4      {
5        "id": 9,
6        "nama": "Bambang",
7        "nim": "1841720123",
8        "email": "bambang@gmail.com",
9        "jurusan": "Teknik Industri",
10       "created_at": "2020-05-03T19:47:49.000000Z",
11       "updated_at": "2020-05-03T19:47:49.000000Z"
12     }
13   ]
14 }

```

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

```
57     public function update(Request $request, $id)
58     {
59         $nama = $request->nama;
60         $nim = $request->nim;
61         $email = $request->email;
62         $jurusan = $request->jurusan;
63
64         $mhs = Mahasiswa::find($id);
65         $mhs->nama = $nama;
66         $mhs->nim = $nim;
67         $mhs->email = $email;
68         $mhs->jurusan = $jurusan;
69
70         if ($mhs->save()) {
71             $res['message'] = "Data Berhasil diubah!";
72             $res['value'] = $mhs;
73             return response($res);
74         }
75
76         else {
77             $res['message'] = "Gagal!";
78             return response($res);
79         }
80     }
81
82 }
83
```

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**

```
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**.

The screenshot shows the Postman interface for a PUT request to `http://localhost:8000/api/mahasiswa/update/9`. The request body is set to `x-www-form-urlencoded` and contains the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	Bambang Pamungkas	
<input checked="" type="checkbox"/> nim	1841720123	
<input checked="" type="checkbox"/> email	bambang@gmail.com	
<input checked="" type="checkbox"/> jurusan	Teknik Kimia	

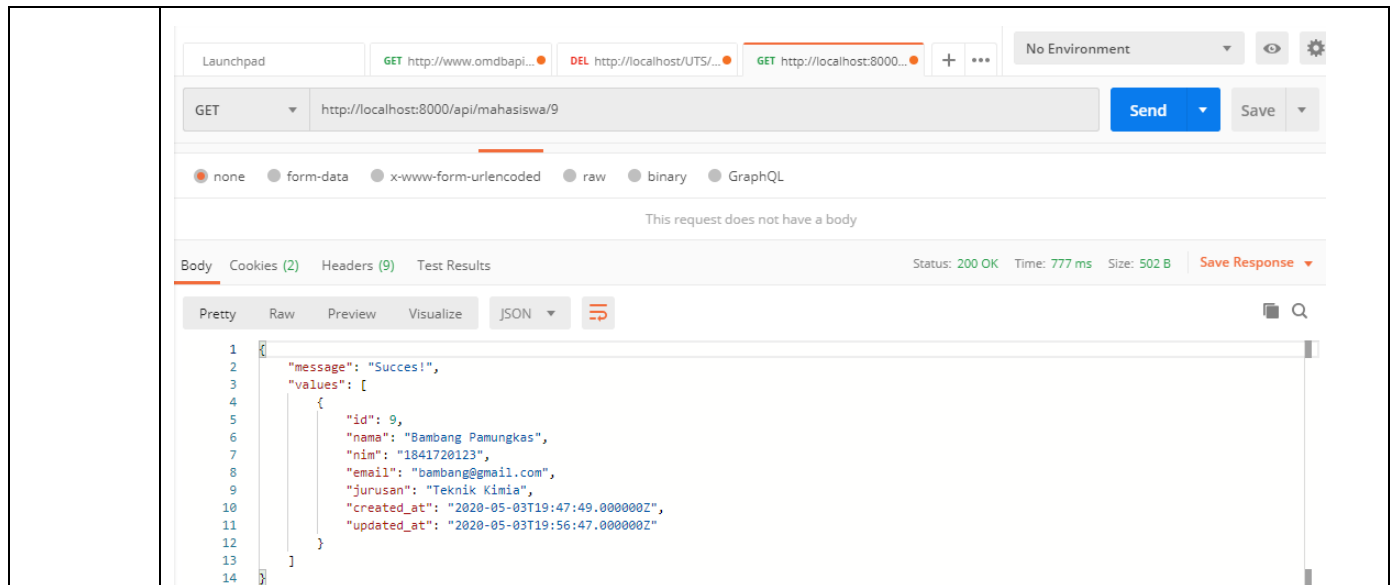
The response is shown in the Body tab, displaying a JSON object:

```

1 {
2   "message": "Data Berhasil diubah!",
3   "value": {
4     "id": 9,
5     "name": "Bambang Pamungkas",
6     "nim": "1841720123",
7     "email": "bambang@gmail.com",
8     "jurusan": "Teknik Kimia",
9     "created_at": "2020-05-03T19:47:49.000000Z",
10    "updated_at": "2020-05-03T19:56:47.000000Z"
11  }
12 }
```

Akan muncul pesan berhasil serta perubahan data dari ID=9

Kemudian coba untuk menampilkan data dengan ID=9 untuk melihat apakah data sudah ter-*update*.



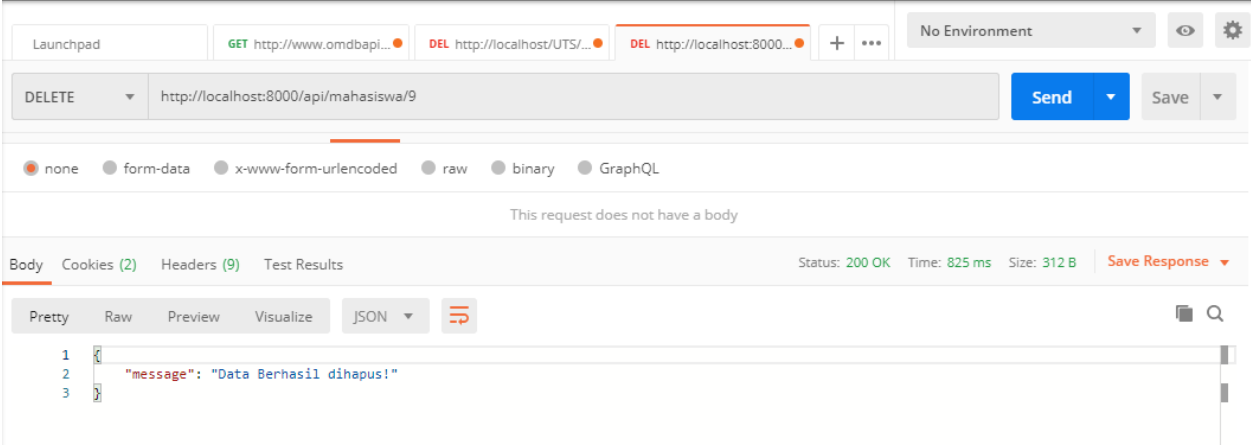
18

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

```
82     public function delete($id)
83     {
84         $mhs = Mahasiswa::where('id', $id);
85
86         if ($mhs->delete()) {
87             $res['message'] = "Data Berhasil dihapus!";
88             return response($res);
89         }
90         else {
91             $res['message'] = "Gagal!";
92             return response($res);
93         }
94     }
95
96 }
97
```

Keterangan:

- 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>Tambahkan <i>route</i> untuk memanggil fungsi delete pada routes/api.php</p> <pre> 28 29 Route::delete('/mahasiswa/{id}', 'MahasiswaController@delete'); 30 </pre> <p>Karena kita ingin menghapus data, maka perintah yang dipakai adalah 'delete.</p>
20	<p>Buka Postman untuk mencoba menghapus data dari database. Gunakan perintah DELETE untuk mengubah data.</p> <p>Berikut adalah contoh untuk menghapus data dengan ID=10, maka url diisi : <u><i>http://localhost:8000/api/mahasiswa /10</i></u></p>  <p>Muncul pesan berhasil ketika data terhapus dari database.</p>

-- Selamat Mengerjakan --