



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

Langka h	Keterangan
1	<p>Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut.</p> <pre>cd C:\xampp\htdocs laravel new laravel-restapi</pre>

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>  A screenshot of a web browser displaying the Laravel default homepage. The page has a white background with the word "Laravel" in a large, light gray font in the center. Below it, there is a horizontal line of links: "DOCS", "LARACASTS", "NEWS", "BLOG", "HOVA", "FORGE", "VAPOB", and "GITHUB". The browser's address bar shows "http://localhost:8080". <p></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>

```

laravel-restapi > .env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:maELZFfnEcwf/rLKfsRBdmQ+M2MVwvNi6aicr77RECU=
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=
15
16 BROADCAST_DRIVER=log
17 CACHE_DRIVER=file
18 QUEUE_CONNECTION=sync
19 SESSION_DRIVER=file
20 SESSION_LIFETIME=120
21
22 REDIS_HOST=127.0.0.1
23 REDIS_PASSWORD=null
24 REDIS_PORT=6379
25
26 MAIL_MAILER=smtp
27 MAIL_HOST=smtp.mailtrap.io
28 MAIL_PORT=2525
29 MAIL_USERNAME=null

```

4

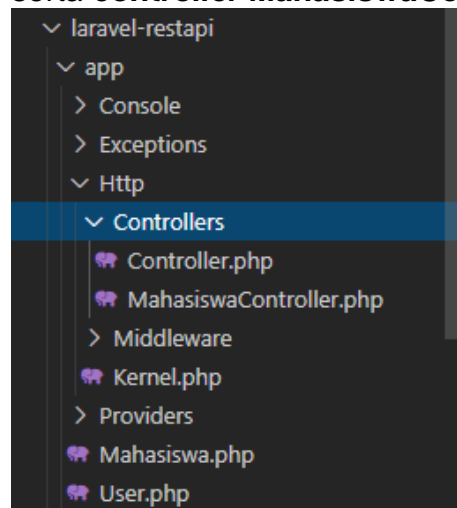
Buat **model** dengan nama **Mahasiswa**, buat juga **controllem**nya. Untuk membuat model dan **controllem**nya sekaligus tuliskan perintah berikut pada *command prompt* (terlebih dahulu keluar dari php artisan serve dengan mengetik ctrl+C pada keyboard)

php artisan make:model Mahasiswa -c

Keterangan :

- -c merupakan perintah untuk menyertakan pembuatan *controller*

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.

```
laravel-restapi > app > Mahasiswa.php > 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:

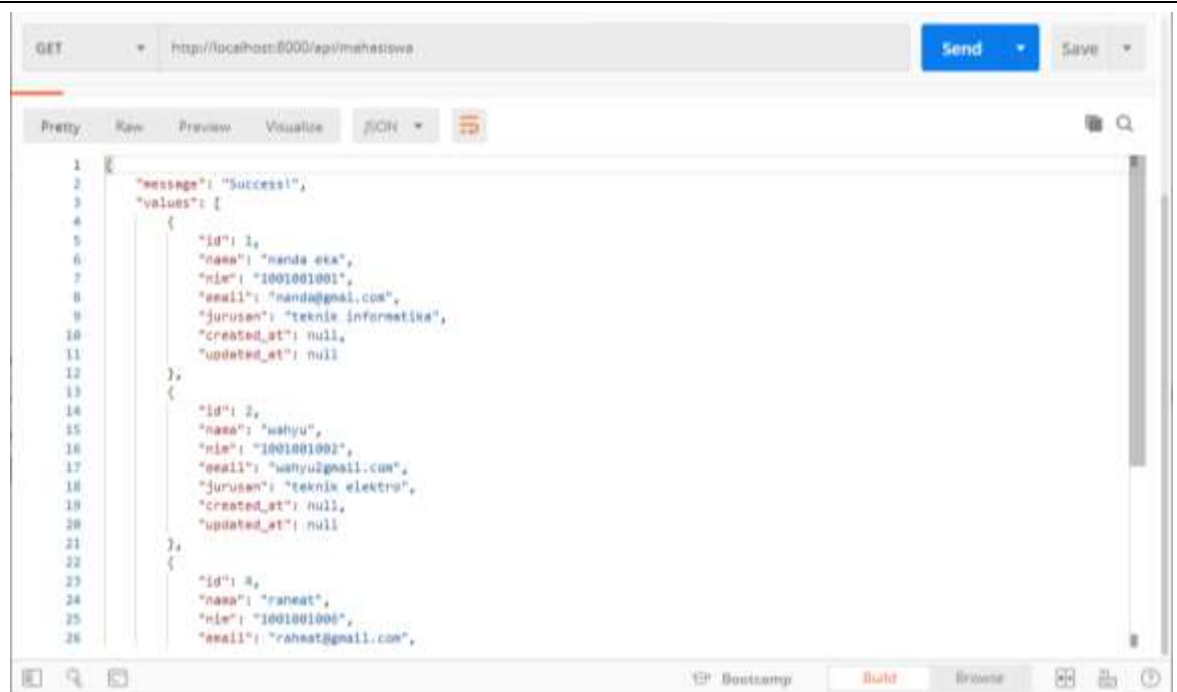
Model ini akan mengelola tabel “mahasiswa” yang terdapat pada database latihan_laravel

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.

```
laravel-restapi > 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'] = "Success!";
16             $res['values'] = $data;
17             return response($res);
18         }else {
19             $res['message'] = "Kosong!";
20             return response($res);
21         }
22     }
23 }
24
```

	<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	<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');</pre> <p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'.</p>
8	<p>Ketikkan perintah php artisan serve pada <i>command prompt</i>. 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>



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

9

Selanjutnya kita akan menambahkan fungsi untuk melihat suatu data ketika dipilih ID tertentu. Buat fungsi baru yaitu **getId** pada **MahasiswaController.php**.'

```

23
24     public function getId($id){
25
26         $data = Mahasiswa::where('id',$id)->get();
27
28         if(count($data) > 0){
29             $res['message'] = "Success!";
30             $res['values'] = $data;
31             return response($res);
32         }else {
33             $res['message'] = "Gagal!";
34             return response($res);
35         }
36     }
37 }

```

Keterangan:

- Fungsi getId menerima parameter \$id yang menunjukkan ID mahasiswa yang dipilih
- Line 30 merupakan pemanggilan model untuk membaca data berdasarkan ID

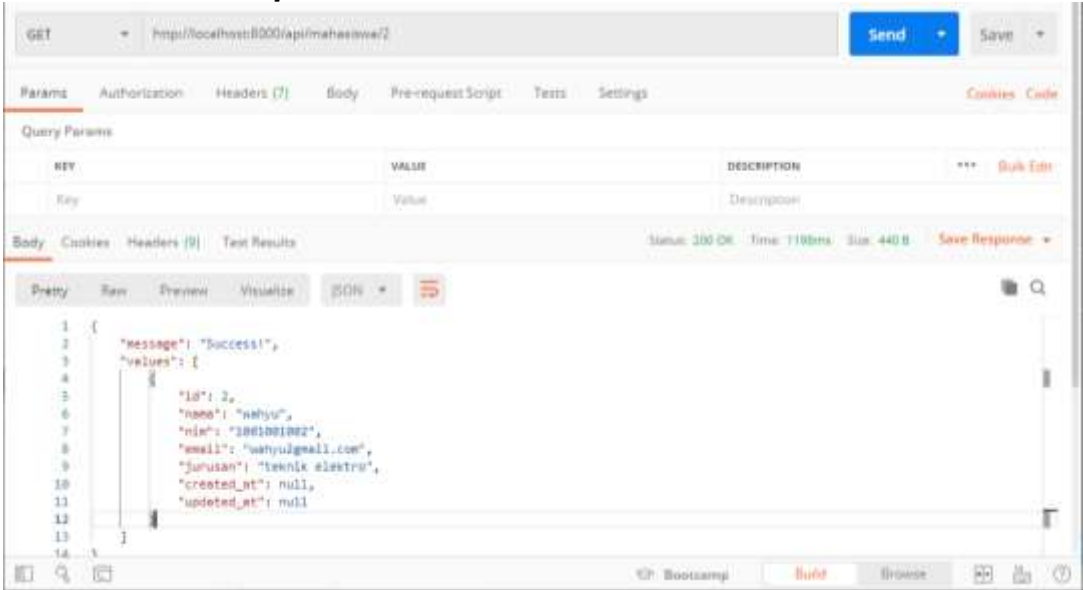
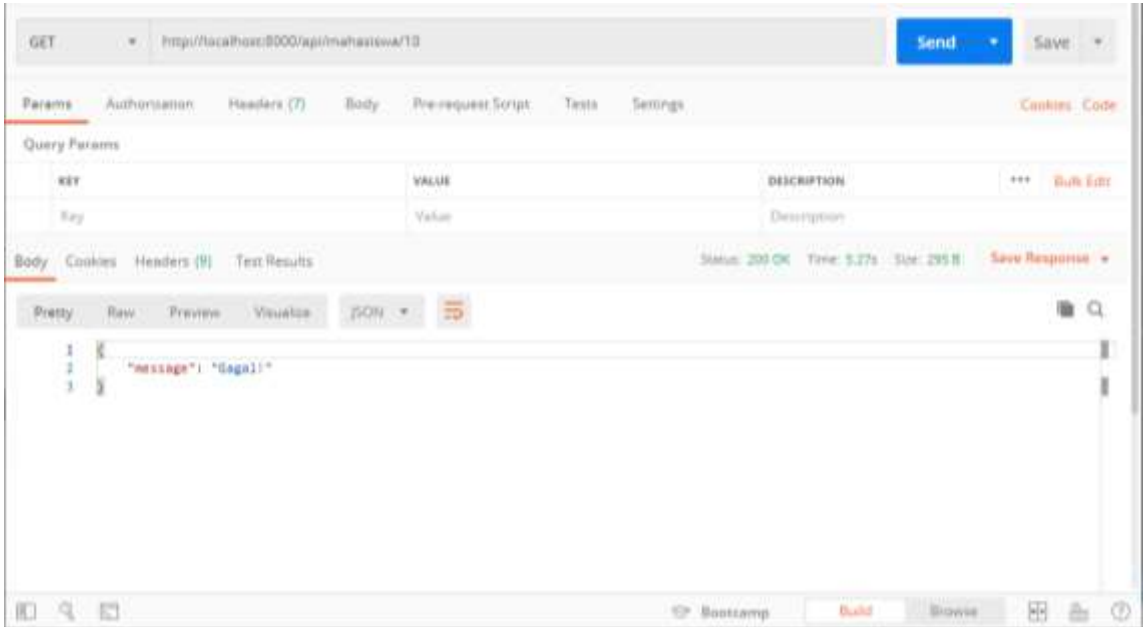
10

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

```

22
23     Route::get('/mahasiswa/{id}', 'MahasiswaController@getId');
24

```

	Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'
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 : <i>http://localhost:8000/api/mahasiswa/2</i></p>  <pre> 1 { 2 "message": "Success!", 3 "values": [4 { 5 "id": 2, 6 "name": "Nahyu", 7 "nim": "1001001002", 8 "email": "nahyu@gmail.com", 9 "jurusan": "teknik elektro", 10 "created_at": null, 11 "updated_at": null 12 } 13] 14 } </pre> <p>Ketika mencoba menampilkan ID=10 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut.</p>  <pre> 1 { 2 "message": "Gagal!" 3 } </pre>
12	Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php .


```

37
38     public function create(Request $request) {
39         $mhs = new Mahasiswa();
40         $mhs->nama = $request->nama;
41         $mhs->nim = $request->nim;
42         $mhs->email = $request->email;
43         $mhs->jurusan = $request->jurusan;
44
45         if($mhs->save()){
46             $res['message'] = "Data berhasil ditambah!";
47             $res['value'] = "$mhs";
48             return response($res);
49         }
50     }
51 }
52

```

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**

```

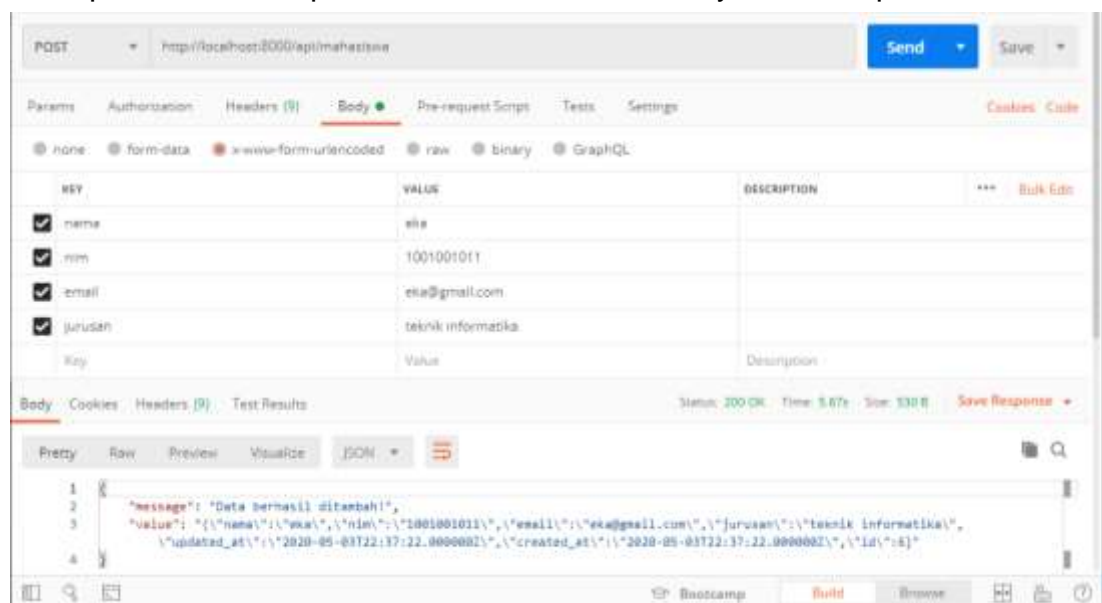
24
25     Route::post('/mahasiswa', 'MahasiswaController@create');
26

```

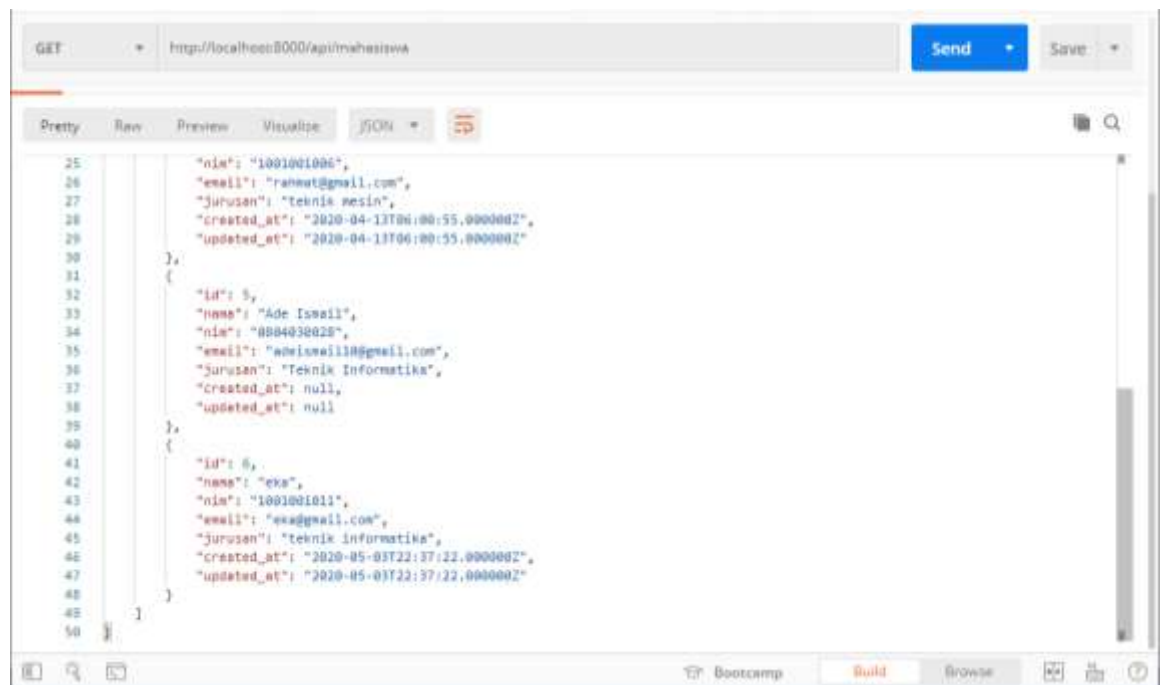
Karena kita ingin menambah data, maka perintah yang dipakai adalah 'post'.

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.



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

Pretty Raw Preview Visualize JSON

[
  {
    "id": 5,
    "name": "Ade Ismail",
    "nim": "8884038028",
    "email": "adeismail@gmail.com",
    "jurusan": "Teknik Mesin",
    "created_at": "2020-04-13T06:00:55.000000Z",
    "updated_at": "2020-04-13T06:00:55.000000Z"
  },
  {
    "id": 6,
    "name": "eka",
    "nim": "1001001011",
    "email": "eka@gmail.com",
    "jurusan": "Teknik Informatika",
    "created_at": "2020-05-03T22:37:22.000000Z",
    "updated_at": "2020-05-03T22:37:22.000000Z"
  },
  {
    "id": 7,
    "name": "eka",
    "nim": "1001001011",
    "email": "eka@gmail.com",
    "jurusan": "Teknik Informatika",
    "created_at": "2020-05-03T22:37:22.000000Z",
    "updated_at": "2020-05-03T22:37:22.000000Z"
  }
]
```

15

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

```
52 public function update(Request $request, $id) {
53     $nama = $request->nama;
54     $nim = $request->nim;
55     $email = $request->email;
56     $jurusan = $request->jurusan;
57
58     $mhs = Mahasiswa::find($id);
59     $mhs->nama = $nama;
60     $mhs->nim = $nim;
61     $mhs->email = $email;
62     $mhs->jurusan = $jurusan;
63
64     if($mhs->save()){
65         $res['message'] = "Data berhasil diubah!";
66         $res['value'] = "$mhs";
67         return response($res);
68     }else {
69         $res['message'] = "Gagal";
70         return response($res);
71     }
72 }
```

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**

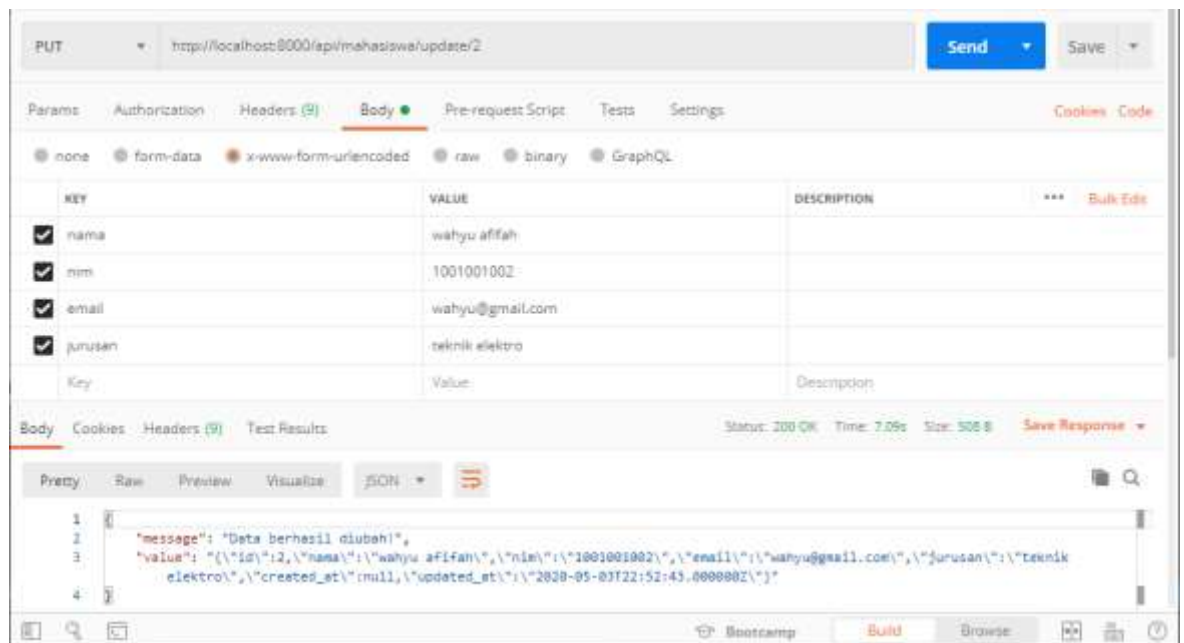
```
26
27 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');
28
```

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

17

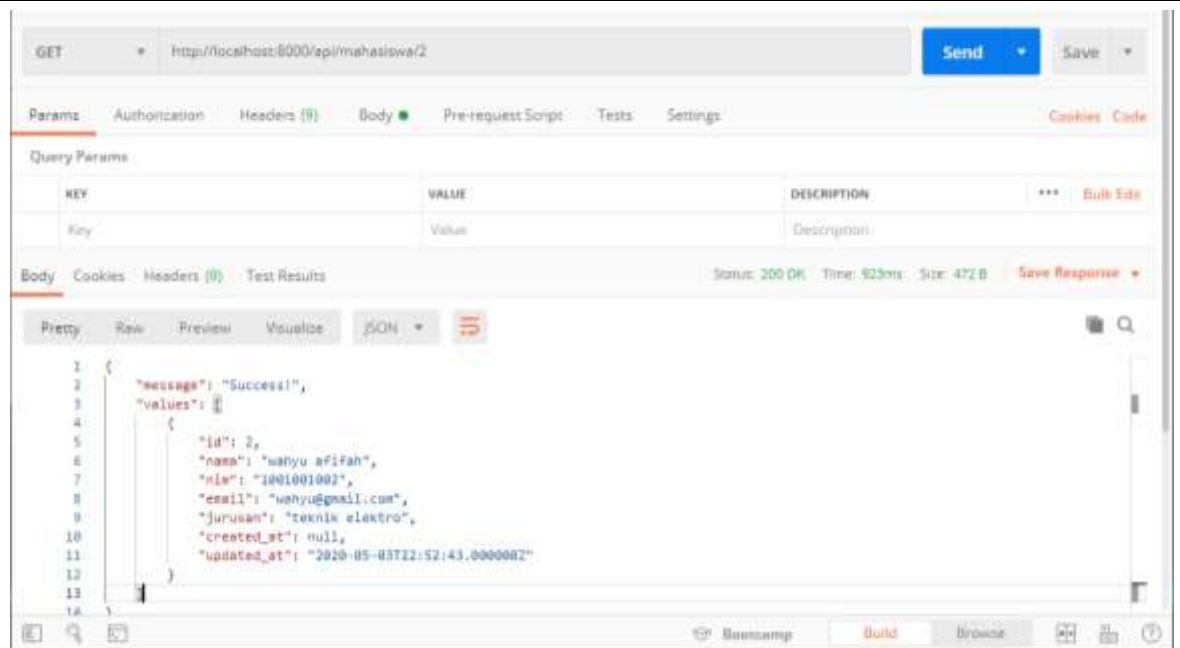
Sekarang kita coba untuk mengubah data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah **PUT** untuk mengubah data.

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**.

```
73
74     public function delete($id){
75
76         $mhs = Mahasiswa::where('id',$id);
77
78         if($mhs->delete()){
79             $res['message'] = "Data berhasil dihapus!";
80             return response($res);
81         }else {
82             $res['message'] = "Gagal!";
83             return response($res);
84         }
85     }
86 }
87
```

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

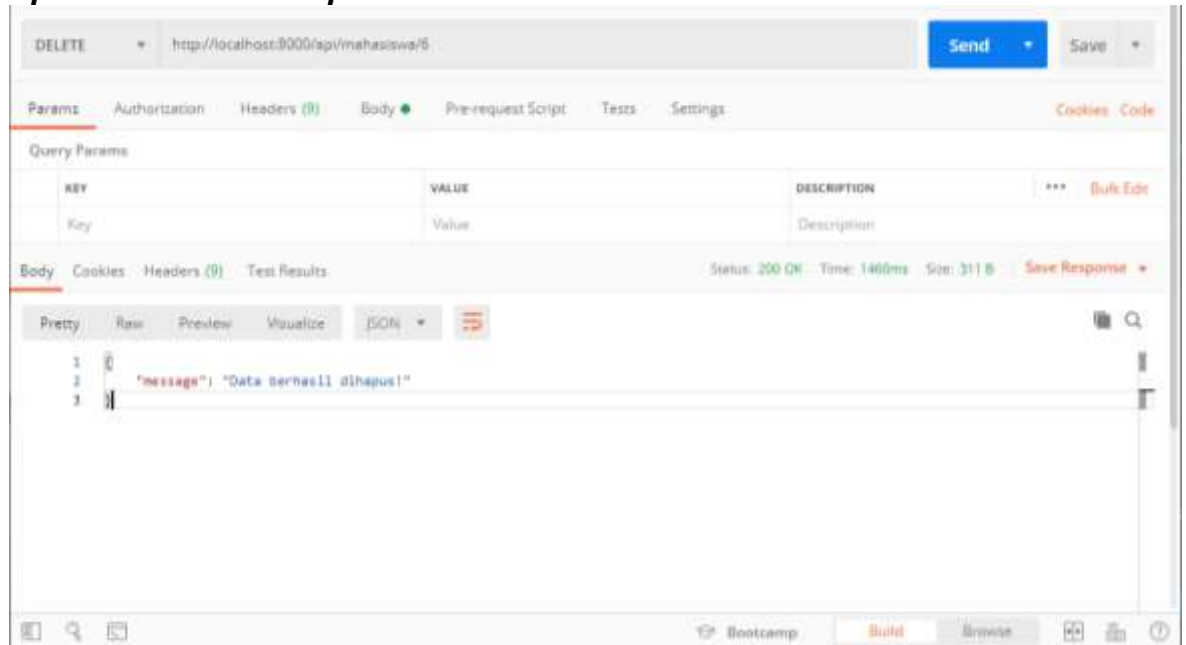
```
29 Route::delete('/mahasiswa/{id}', 'MahasiswaController@delete');
```

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=6, maka url diisi :
http://localhost:8000/api/mahasiswa/6



Muncul pesan berhasil ketika data terhapus dari database.

-- Selamat
Mengerjakan --