

Jobsheet 14 : Membuat Restful API Laravel

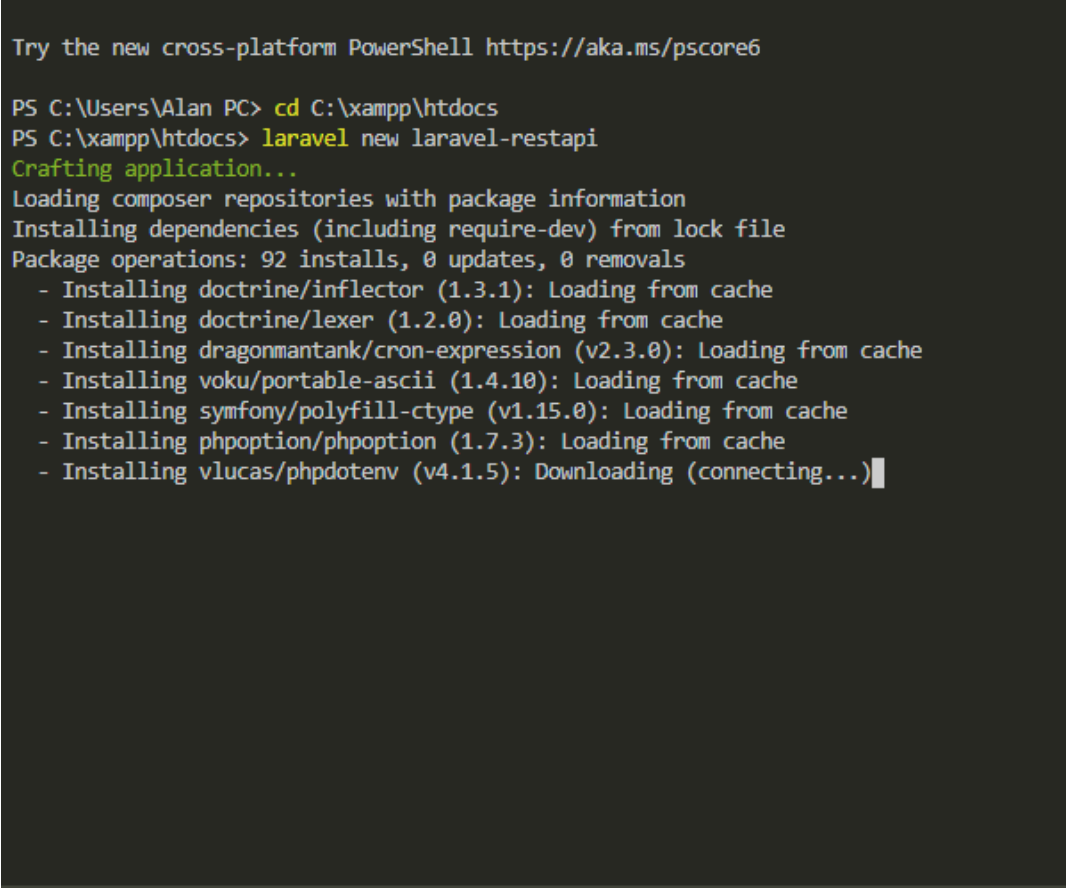
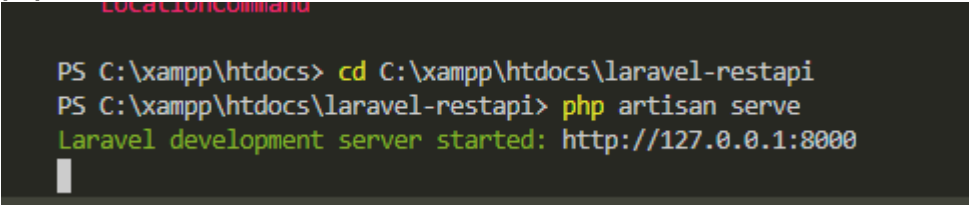
Mata Kuliah Pemrograman Web Lanjut

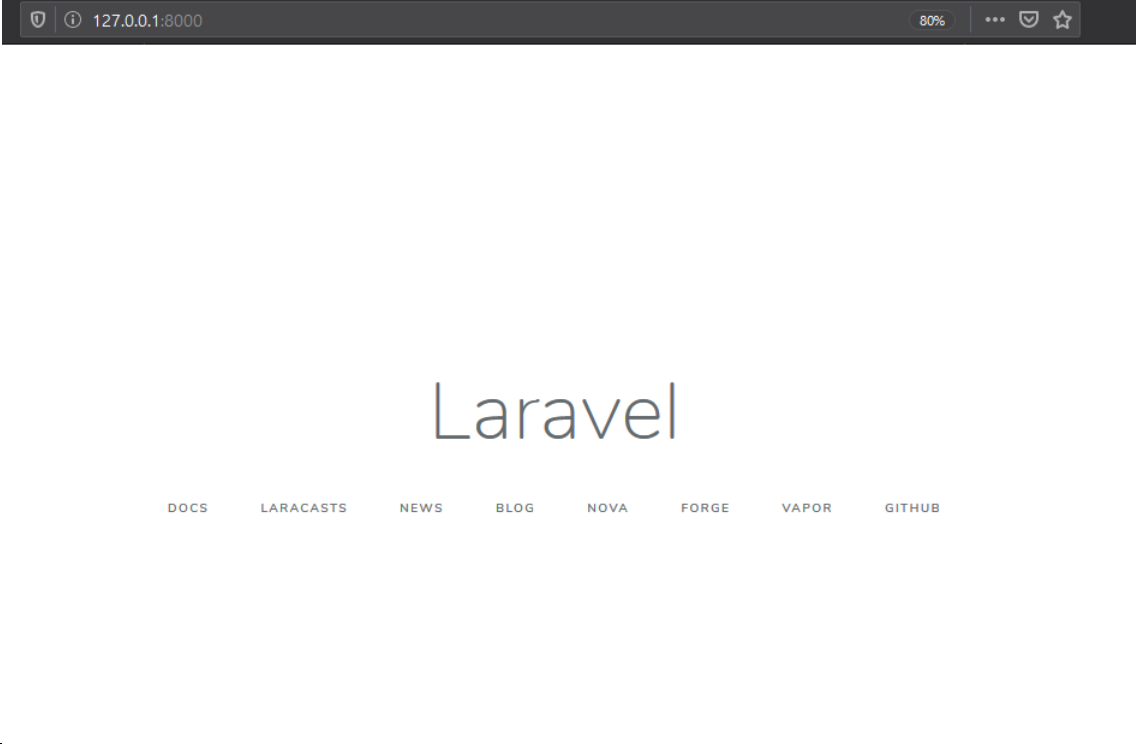


Oleh :

**M AQSAL ZILDJIAN / NIM : 1841720191
KELAS TI 2A**

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

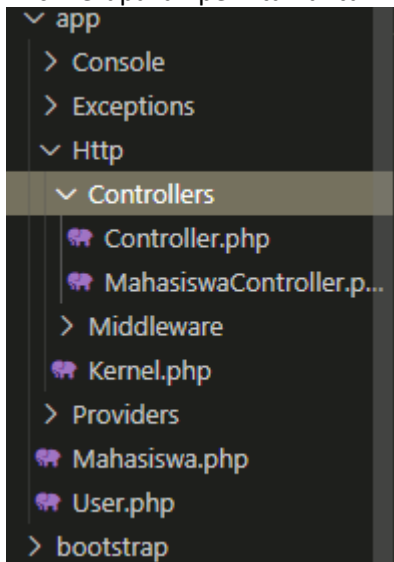
1	<p>Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut.</p> <pre>cd C:\xampp\htdocs laravel new laravel-restapi</pre> 
2	<p>Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut.</p> <pre>cd C:\laravel-restapi php artisan serve</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 data-bbox="427 1041 1256 1654"> 1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:9ti1NjggplAtG7Ieubo2zP3fhR 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 </pre>
4	<p>Buat model dengan nama Mahasiswa, buat juga controllernya. Untuk membuat model dan <i>controllernya</i> 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>

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

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.

```
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.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Mahasiswa;

class MahasiswaController extends Controller
{
    //fungsi index untuk tampilkan semua data mahasiswa
    public function index()
    {
        $data = Mahasiswa::all();

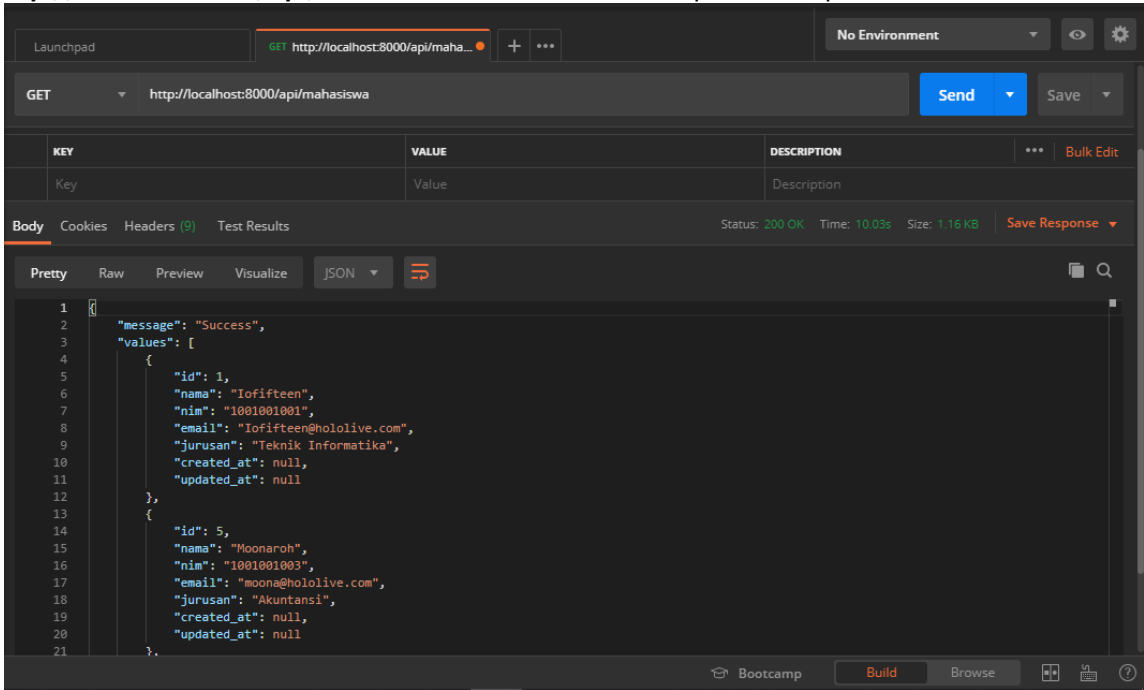
        // cek data tidak kosong
        if(count($data) > 0){
            $res['message'] = 'Success';
            $res['values'] = $data;
            return response($res);
        }
        //jika data kosong
        else{
            $res['message'] = 'kosong!';
            return response($res);
        }
    }
}
```

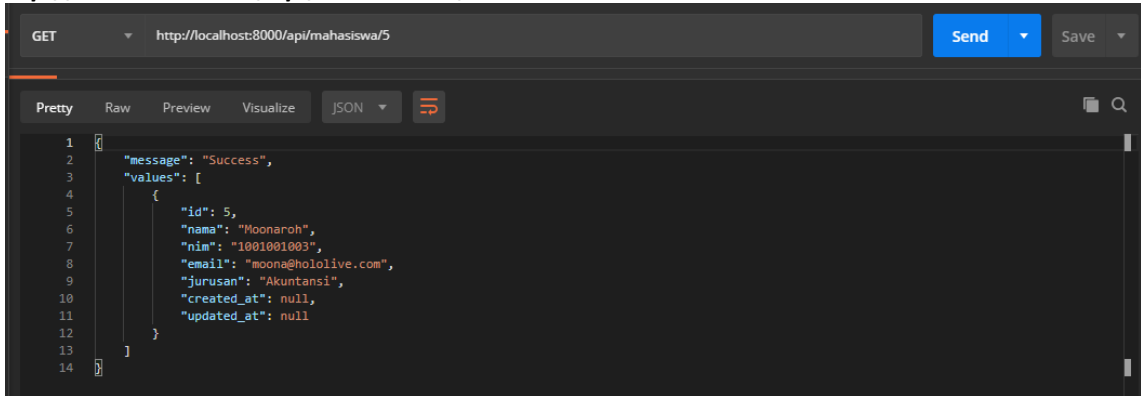
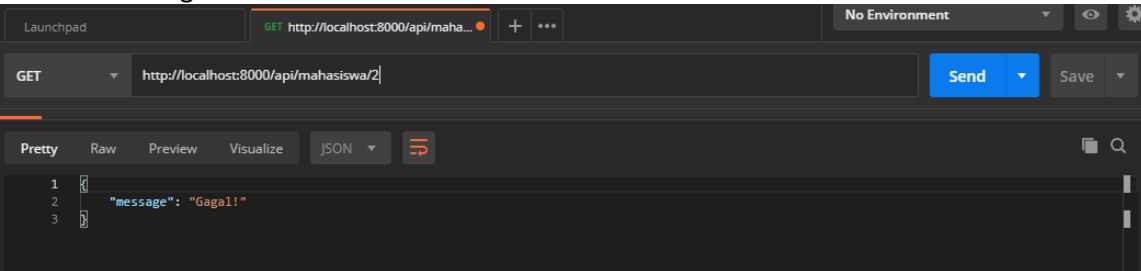
Keterangan:

- 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).

```
16
17 Route::middleware('auth:api')->get('/user', function (Request $r
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	<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 : <i>http://localhost:8000/api/mahasiswa</i> Berikut adalah tampilan dari aplikasi Postman</p>  <p>Semua data pada tabel mahasiswa akan tampil, ditampilkan juga pesan sukses.</p>
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> //fungsi untuk tampil data dari sebuah ID public function getId(\$id) { \$data = Mahasiswa::where('id',\$id)->get(); // cek data 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 <i>route</i> untuk memanggil fungsi getId pada routes/api.php</p> <pre>Route::get('mahasiswa', 'MahasiswaController@index'); 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. Di bawah ini adalah contoh untuk menampilkan data dengan ID=5, maka url diisi :</p> <p><i>http://localhost:8000/api/mahasiswa/5</i></p>  <pre>1 { 2 "message": "Success", 3 "values": [4 { 5 "id": 5, 6 "nama": "Moonaroh", 7 "nim": "1001001003", 8 "email": "moona@hololive.com", 9 "jurusan": "Akuntansi", 10 "created_at": null, 11 "updated_at": null 12 } 13] 14 }</pre> <p>Ketika mencoba menampilkan ID=2 akan muncul pesan "Gagal", karena tidak ada data mahasiswa dengan ID tersebut.</p>  <pre>1 { 2 "message": "Gagal!" 3 }</pre>
12	<p>Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php.</p>

```

    }
    //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 di tambahkan
        if($mhs->save()){
            $res['message']="Data berhasil di tambahkan !";
            $res['value']= "$mhs";
            return response($res);
        }
    }
}

```

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**

```

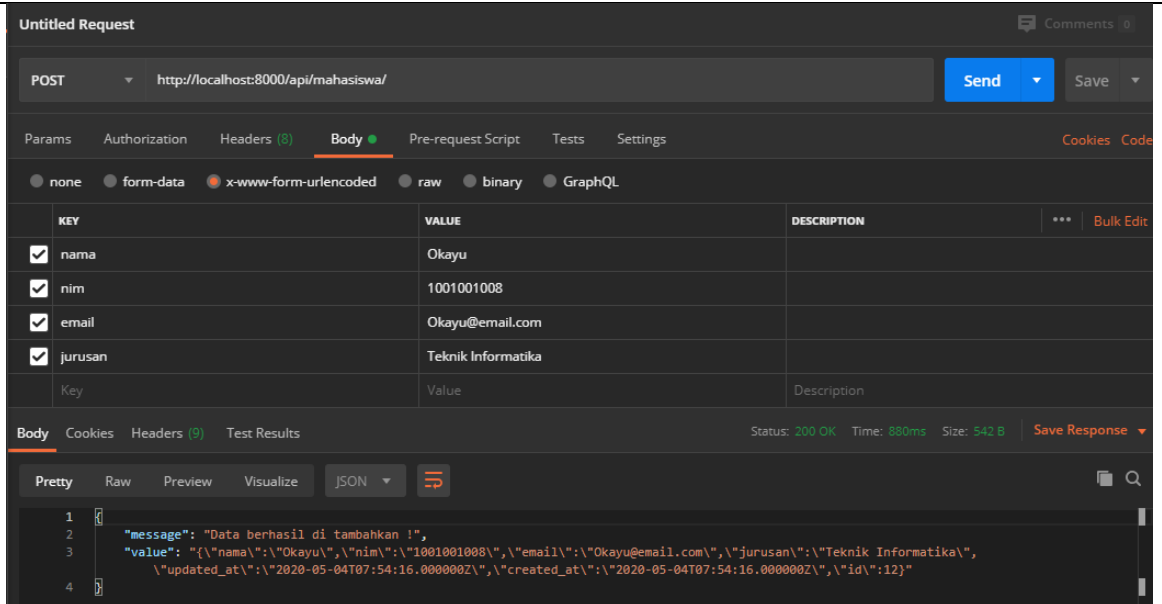
22 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId');
23 Route::post('/mahasiswa', 'MahasiswaController@create');

```

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.

```

57 },
58 {
59   "id": 12,
60   "nama": "Okayu",
61   "nim": "1001001008",
62   "email": "Okayu@email.com",
63   "jurusan": "Teknik Informatika",
64   "created_at": "2020-05-04T07:54:16.000000Z",
65   "updated_at": "2020-05-04T07:54:16.000000Z"
66 }
67 ]
68 }

```

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

```

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 terupdate!";
        $res['value'] = "$mhs";
        return response($res);
    }
    else{
        $res['message']="data tidak bisa diupdate!";
        return response($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**

```

23 Route::post('/mahasiswa', 'MahasiswaController@create');
24 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');

```

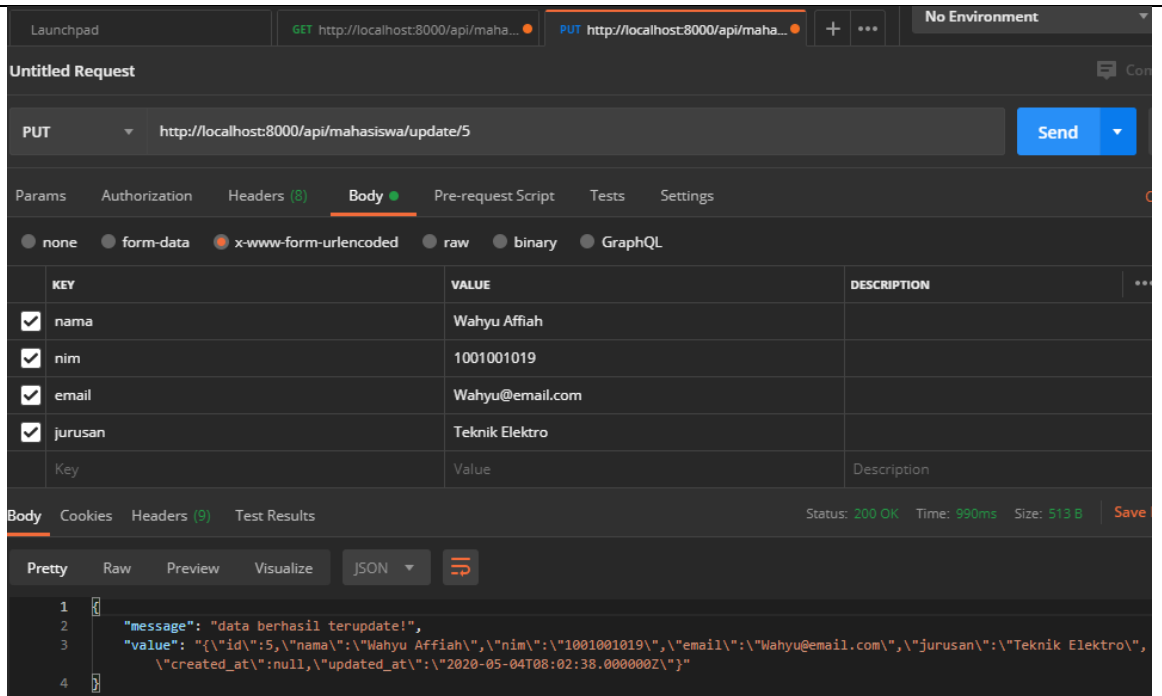
Karena kita ingin memasukkan perubahan data, maka perintah yang dipakai adalah 'put'

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=5, maka url diisi :

<http://localhost:8000/api/mahasiswa/update/5>. Pilih tab **Body** dan pilih radio button **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=5.



Kemudian coba untuk menampilkan data dengan ID=5 untuk melihat apakah data sudah ter-update.

```

{
  "id": 5,
  "nama": "Wahyu Affiah",
  "nim": "1001001019",
  "email": "Wahyu@email.com",
  "jurusan": "Teknik Elektro",
  "created_at": null,
  "updated_at": "2020-05-04T08:02:38.000000Z"
},

```

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

```

82     public function delete($id)
83     {
84
85         $mhs = Mahasiswa::where('id', $id);
86
87         if ($mhs->delete()) {
88             $res['message'] = "Data Berhasil dihapus";
89             return response($res);
90         } else {
91             $res['message'] = "data gagal dihapus!";
92             return response($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 Tambahkan *route* untuk memanggil fungsi delete pada **routes/api.php**

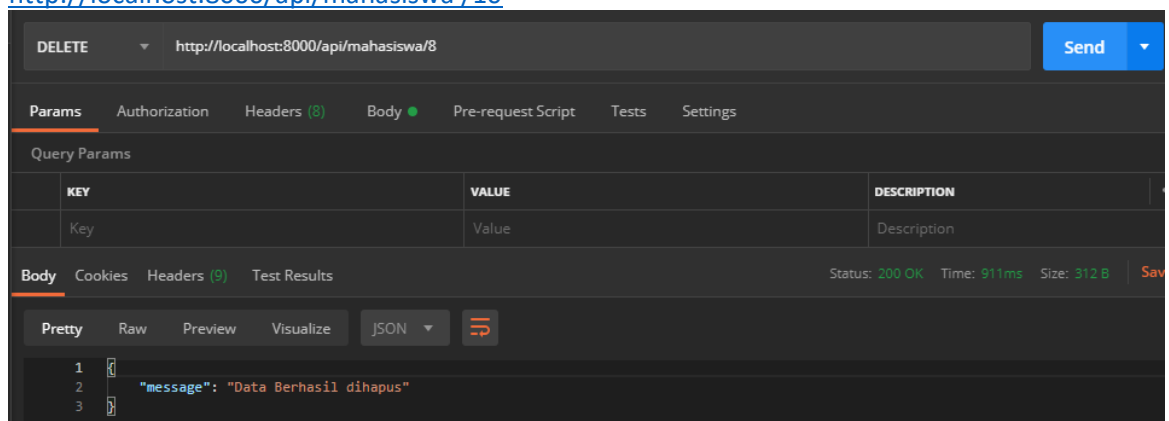
```
Route::delete('/mahasiswa/{id}', 'MahasiswaController@delete');
```

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=8, maka url diisi :

<http://localhost:8000/api/mahasiswa/10>



Muncul pesan berhasil ketika data terhapus dari database