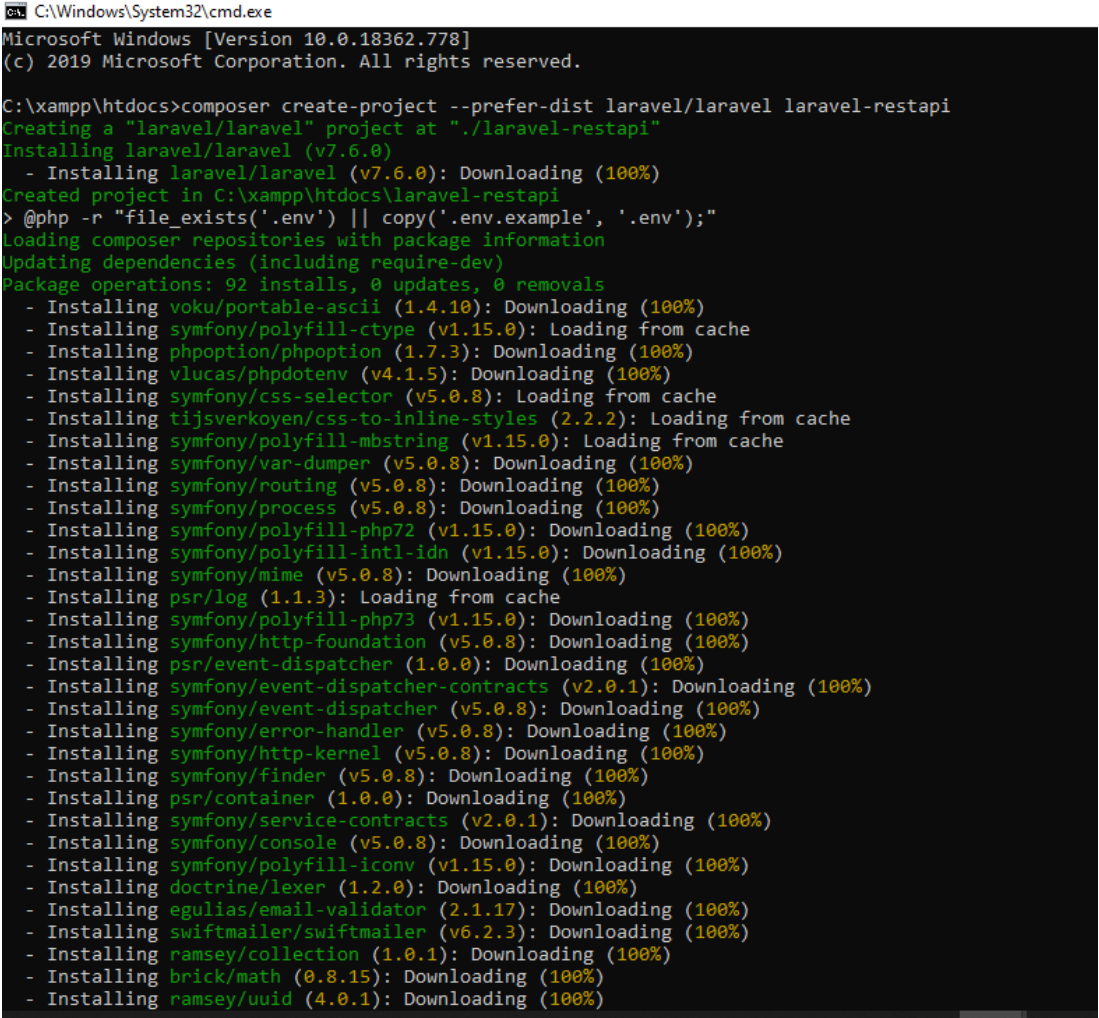




NAMA : HUSNUL HOTIMAH
NIM : 1841720014
KELAS : TI-2A

Praktikum: Membuat RESTful API di Laravel

Langkah	Keterangan
1	<p>Buat project baru dengan nama "laravel-restapi". Buka command prompt, tuliskan perintah berikut.</p> <pre>C:\xampp\htdocs>composer create-project --prefer-dist laravel/laravel laravel-restapi</pre> 

2

Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut.

```
cd C:\laravel-restapi
php artisan serve
```

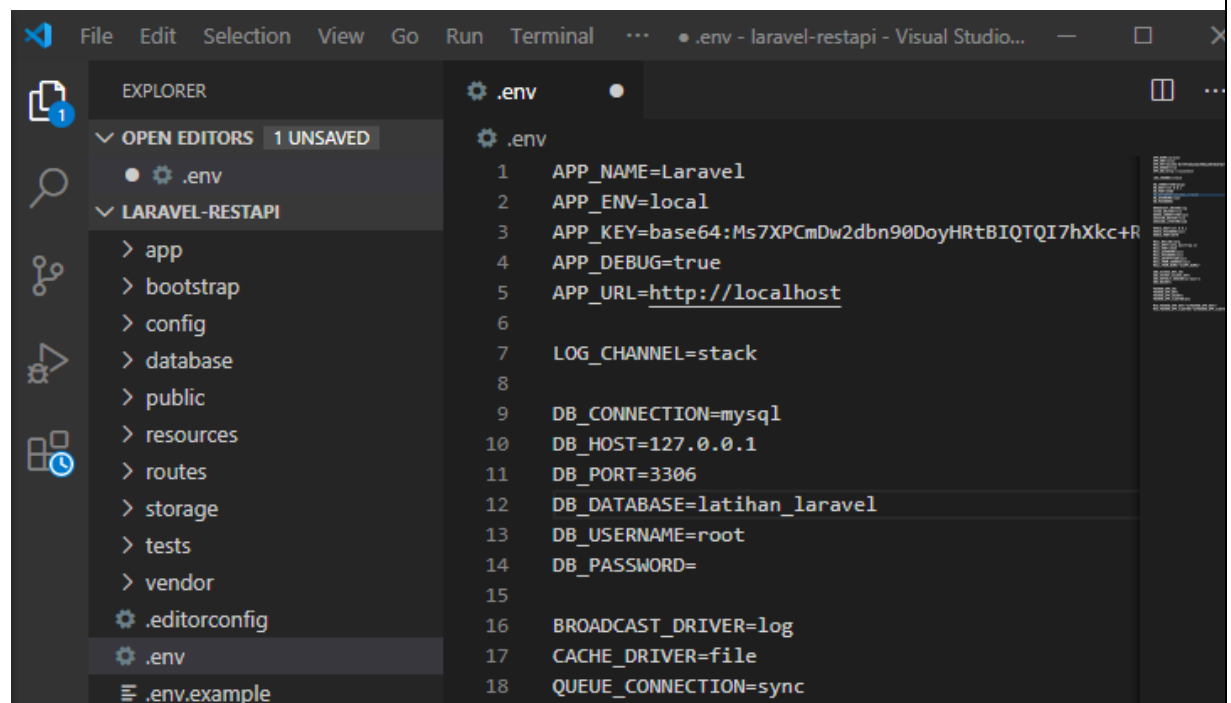
```
C:\xampp\htdocs>cd laravel-restapi
C:\xampp\htdocs\laravel-restapi>php artisan serve
Laravel development server started: http://127.0.0.1:8000
```

Akan tampil halaman default Laravel seperti di bawah ini.



3

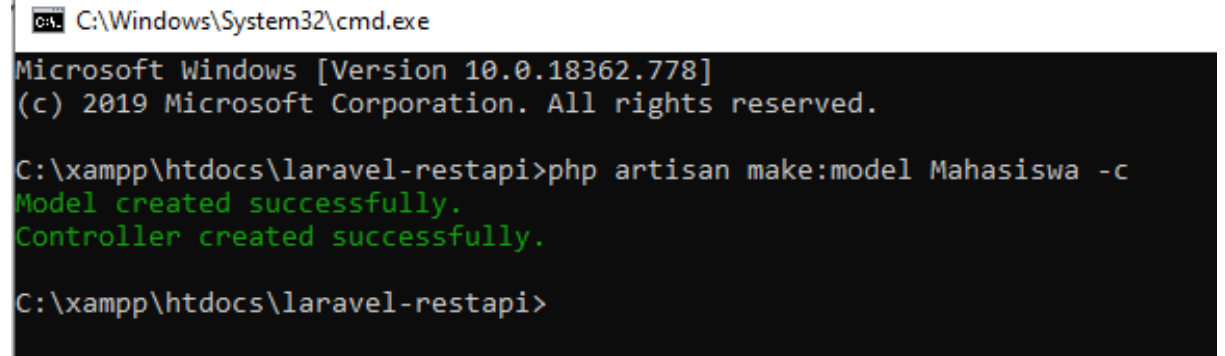
Kemudian lakukan konfigurasi *database* pada file **.env**. Isikan nama database, username, dan password yang akan digunakan. Pada project ini, kita gunakan database dari latihan di minggu-minggu sebelumnya yaitu **"latihan_laravel"**



4

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)

php artisan make:model Mahasiswa -c



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.18362.778]
(c) 2019 Microsoft Corporation. All rights reserved.

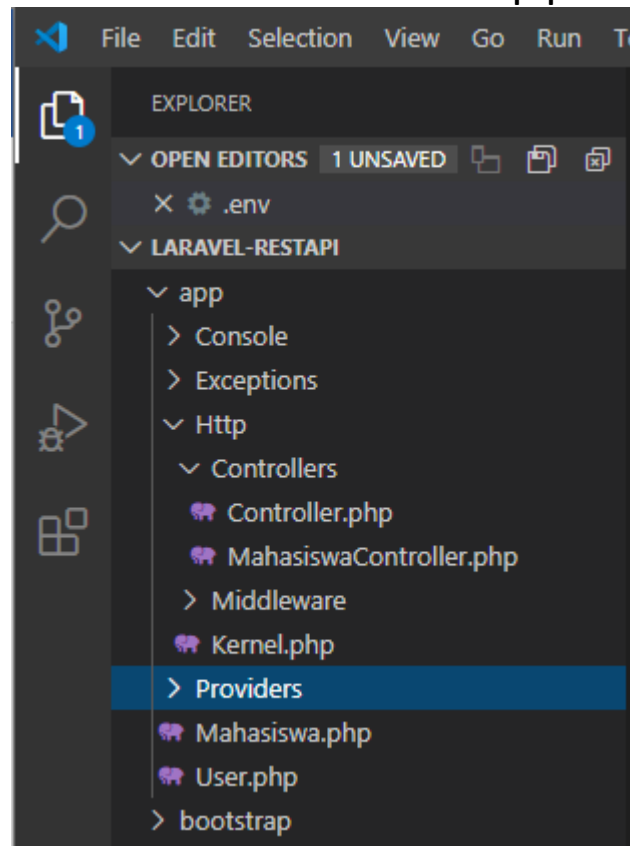
C:\xampp\htdocs\laravel-restapi>php artisan make:model Mahasiswa -c
Model created successfully.
Controller created successfully.

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.

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.

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).

```
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
```

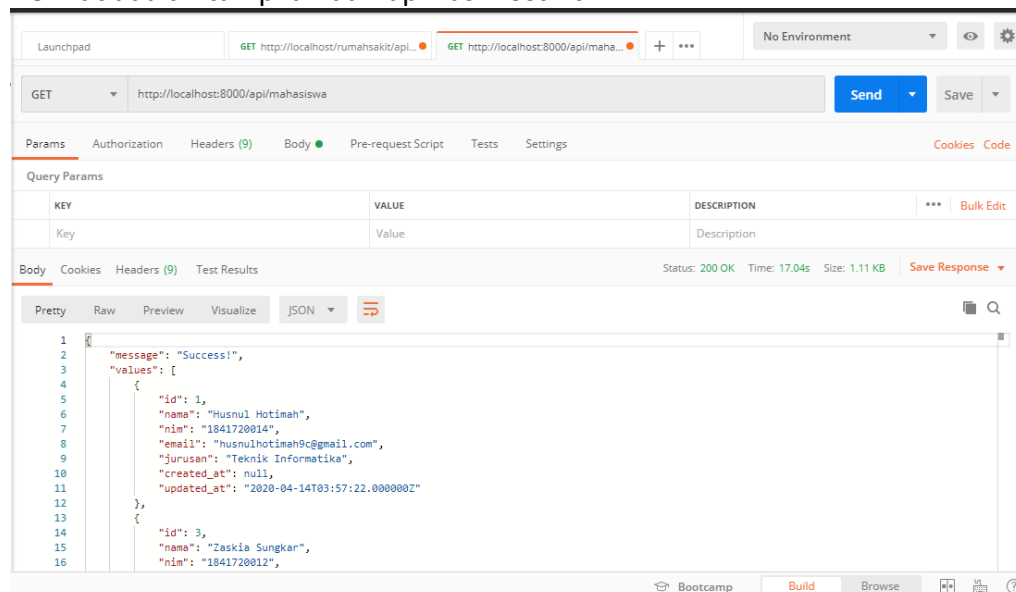
Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'.

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.



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**.

```

27 //fungsi untuk menampilkan data dari sebuah ID
28 public function getId($id){
29     $data = Mahasiswa::where('id',$id)->get();
30
31     //cek jika data ditemukan
32     if(count($data)>0){
33         $res['message'] = "Success!";
34         $res['values'] = $data;
35         return response($res);
36     }
37     //jika data tidak ditemukan
38     else{
39         $res['message'] = "Gagal!";
40         return response($res);
41     }
42 }
43

```

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

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=7, maka url diisi :
http://localhost:8000/api/mahasiswa/7

GET http://localhost:8000/api/mahasiswa/7

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 1205ms Size: 497 B Save Response

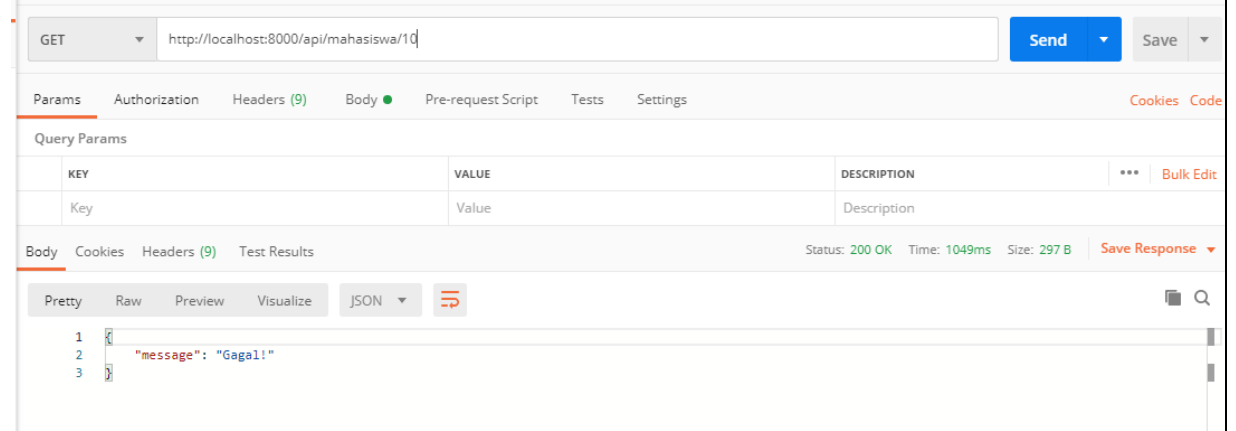
Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "Success!",
3   "values": [
4     {
5       "id": 7,
6       "nama": "Muhammad Amar",
7       "nim": "1841720013",
8       "email": "amar@gmail.com",
9       "jurusan": "Teknik Mesin",
10      "created_at": "2020-04-14T03:26:03.000000Z",
11      "updated_at": "2020-04-14T03:26:03.000000Z"
12    }
13  ]
14 }

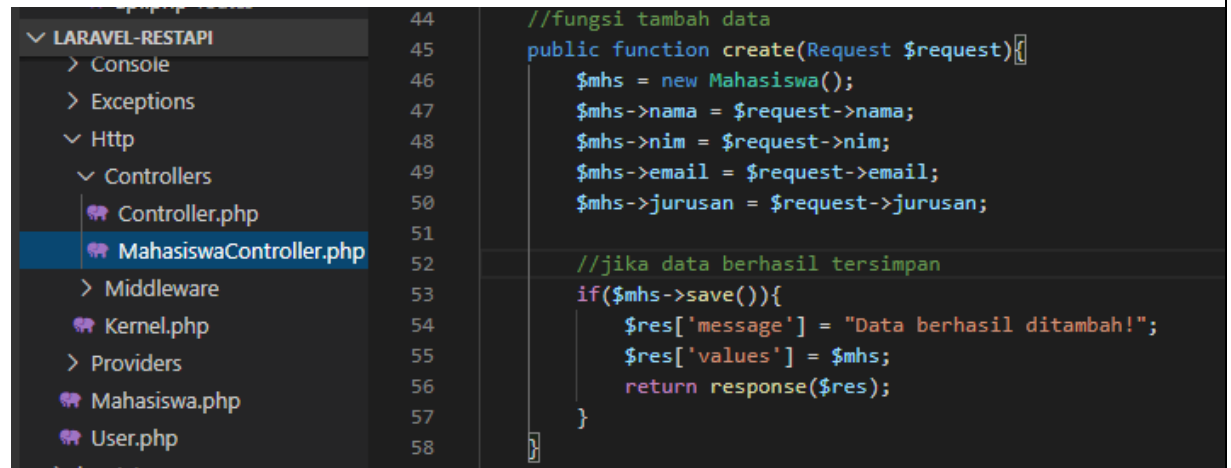
```

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**.



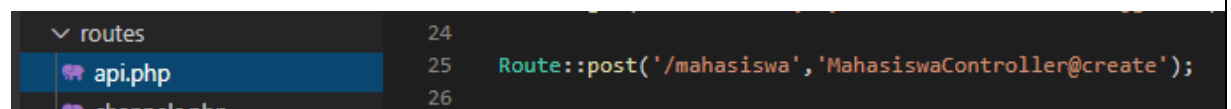
Keterangan:

- Fungsi **create** menerima parameter **Request** yang menampung isian data mahasiswa yang akan ditambahkan ke database.

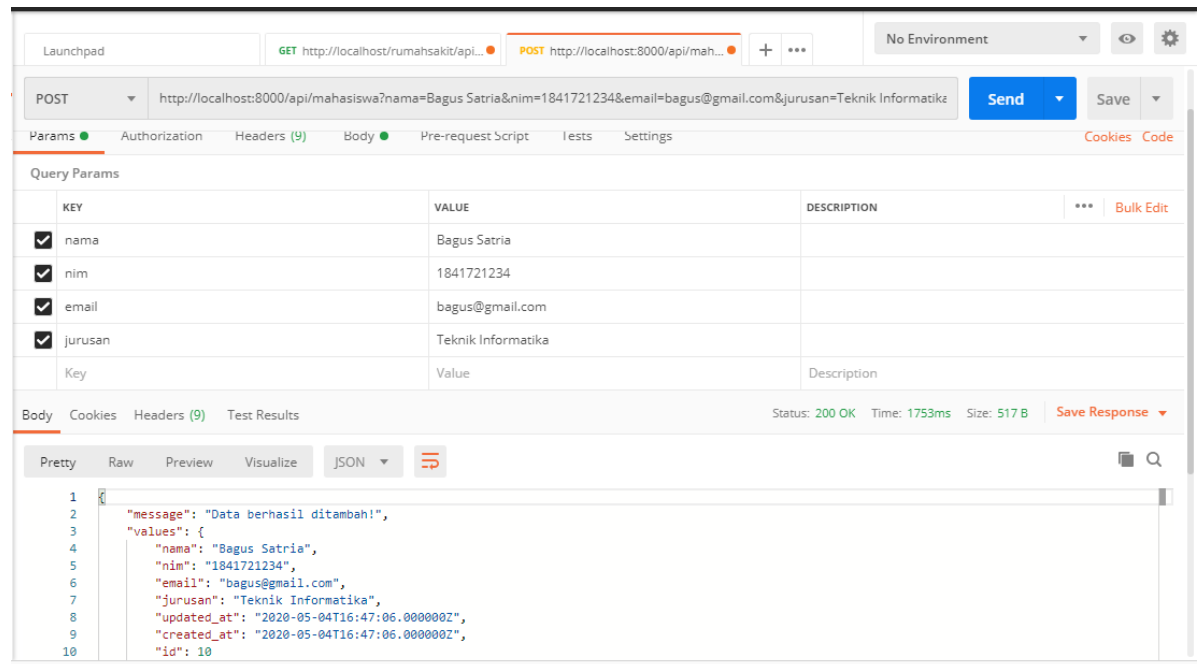
Line 55-59 : `$mhs->save()` digunakan untuk menyimpan data ke database, apabila `save()` berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang ditambahkan.

13

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

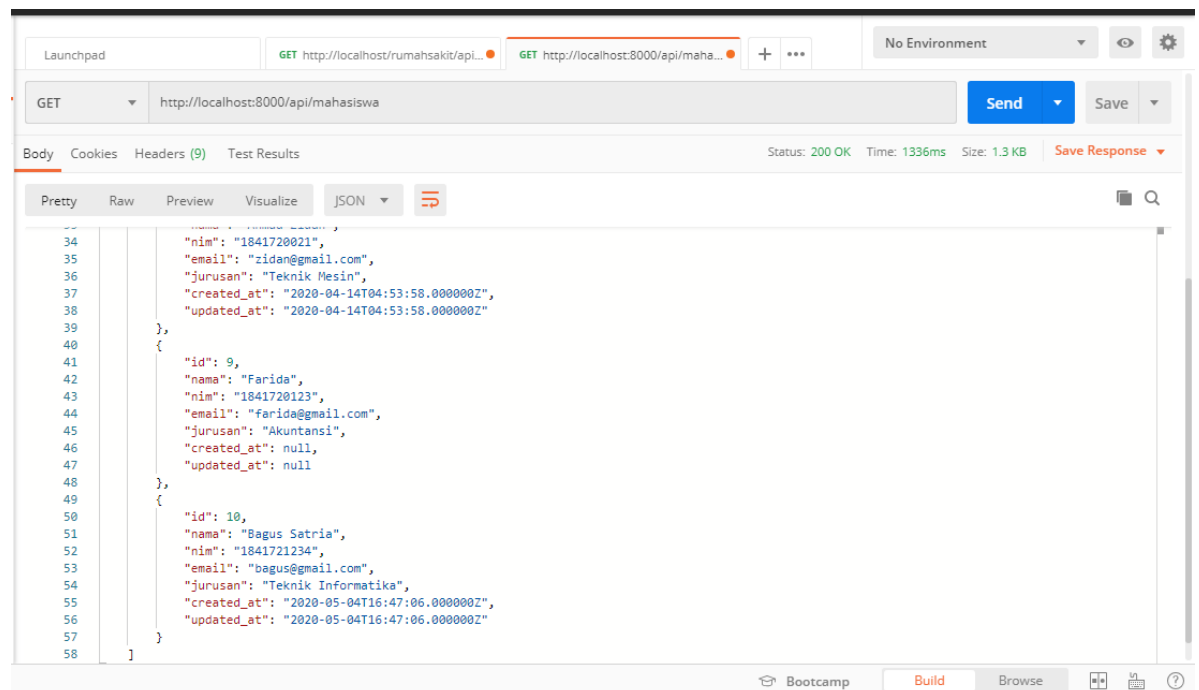


Karena kita ingin menambah data, maka perintah yang dipakai adalah 'post'



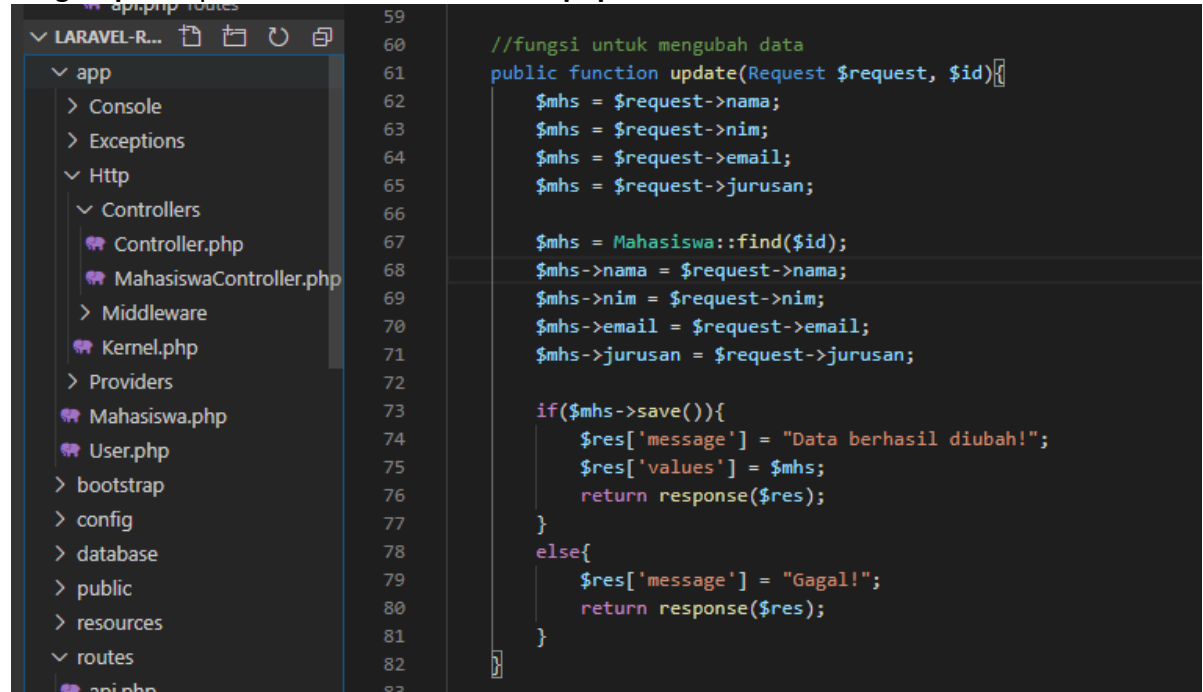
- Isikan url : **`http://localhost:8000/api/mahasiswa`**. Karena kita ingin mengirim data ke database, maka perintah yang dipakai adalah **'POST'**.
- Pilih tab **Body** dan pilih radio button **x-www-form-urlencoded**. Isikan nama kolom pada database pada **KEY**, untuk isian datanya tuliskan pada **VALUE**.

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**.



```

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

```

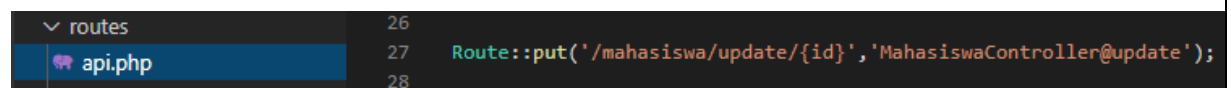
Keterangan:

- 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**



```

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.

Akan mengubah data berikut ini



```

49 {
50     "id": 10,
51     "nama": "Bagus Satria",
52     "nim": "1841721234",
53     "email": "bagus@gmail.com",
54     "jurusan": "Teknik Informatika",
55     "created_at": "2020-05-04T16:47:06.000000Z",
56     "updated_at": "2020-05-04T16:47:06.000000Z"
57 }
58 ]
59

```

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 a Postman interface with a PUT request to `http://localhost:8000/api/mahasiswa/update/10`. The 'Body' tab is selected, and the 'x-www-form-urlencoded' radio button is chosen. The body is represented as a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'.

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	Bagus Satria Bima	
<input checked="" type="checkbox"/> nim	1841720000	
<input checked="" type="checkbox"/> email	bagussattria@gmail.com	
<input checked="" type="checkbox"/> jurusan	Teknik Informatika	
Key	Value	Description

The status bar shows 'Status: 200 OK', 'Time: 1794ms', and 'Size: 527 B'. The response body is shown in JSON format:

```

1 {
2   "message": "Data berhasil diubah!",
3   "values": {
4     "id": 10,
5     "nama": "Bagus Satria Bima",
6     "nim": "1841720000",
7     "email": "bagussattria@gmail.com",
8     "jurusan": "Teknik Informatika",
  
```

Akan muncul pesan berhasil serta perubahan data dari ID=10.

Kemudian coba untuk menampilkan data dengan ID=10 untuk melihat apakah data sudah ter-*update*.

The screenshot shows a Postman interface with a GET request to `http://localhost:8000/api/mahasiswa/`. The 'Pretty' tab is selected, and the response is shown in JSON format:

```

34   "nim": "1841720021",
35   "email": "zidan@gmail.com",
36   "jurusan": "Teknik Mesin",
37   "created_at": "2020-04-14T04:53:58.000000Z",
38   "updated_at": "2020-04-14T04:53:58.000000Z"
39 },
40 {
41   "id": 9,
42   "nama": "Farida",
43   "nim": "1841720123",
44   "email": "farida@gmail.com",
45   "jurusan": "Akuntansi",
46   "created_at": null,
47   "updated_at": null
48 },
49 {
50   "id": 10,
51   "nama": "Bagus Satria Bima",
52   "nim": "1841720000",
53   "email": "bagussattria@gmail.com",
54   "jurusan": "Teknik Informatika",
55   "created_at": "2020-05-04T16:47:06.000000Z",
56   "updated_at": "2020-05-04T16:56:52.000000Z"
57 }
58 ]
59 }
  
```

18

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

```

83
84 //fungsi untuk menghapus data
85 public function delete($id){
86     $mhs = Mahasiswa::where('id',$id);
87
88     if($mhs->delete()){
89         $res['message'] = "Data berhasil dihapus!";
90         return response($res);
91     }
92     else{
93         $res['message'] = "Gagal!";
94         return response($res);
95     }
96 }
97
98
99 }

```

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**

```

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

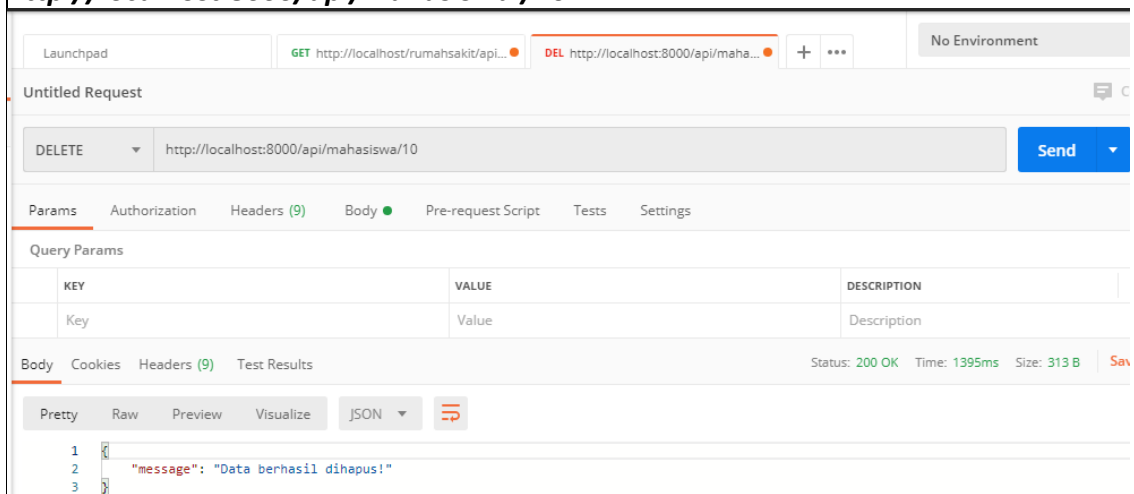
```

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=10, maka url diisi :
http://localhost:8000/api/mahasiswa /10



Muncul pesan berhasil ketika data terhapus dari database.