

LAPORAN PRAKTIKUM
PEMROGRAMAN WEB LANJUT

Jobsheet-14

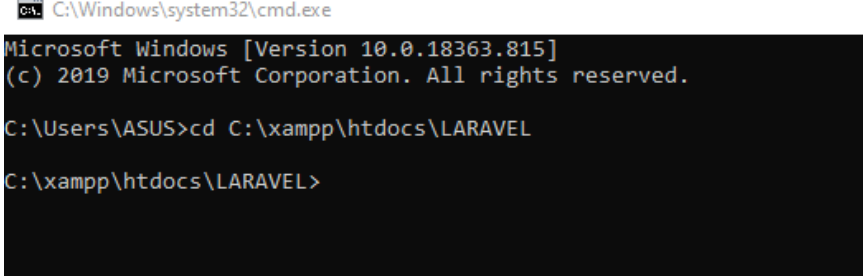
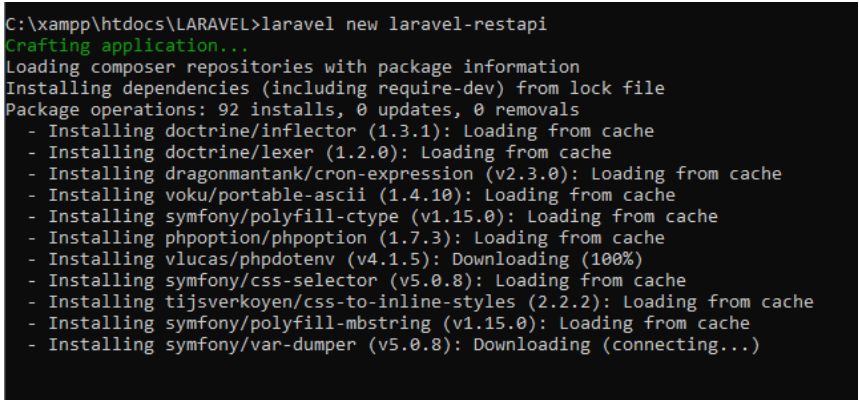
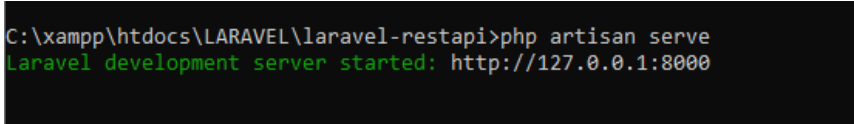
Membuat RestFull API Laravel

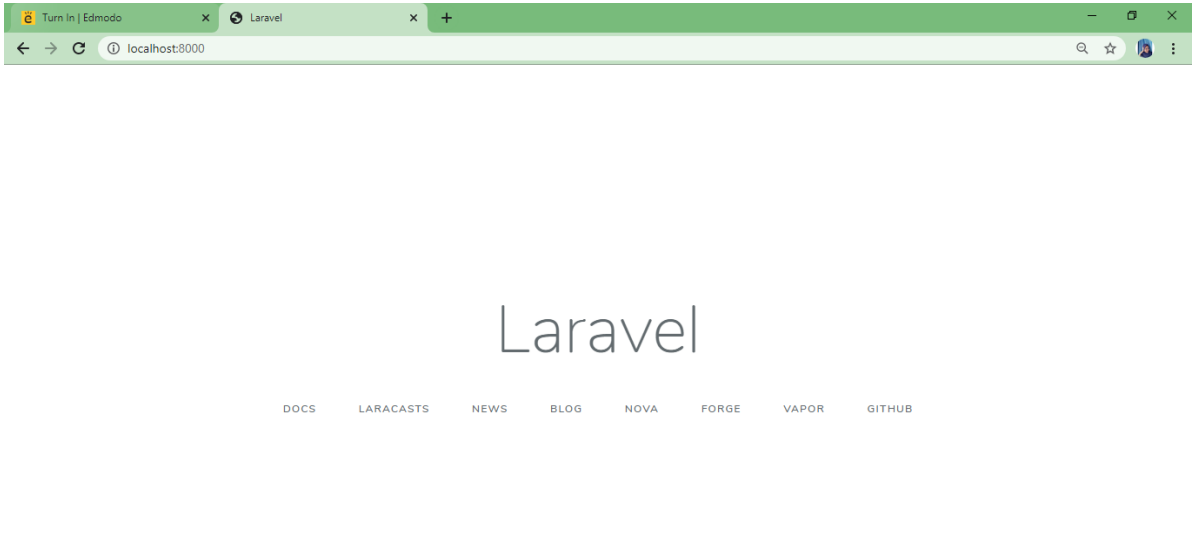
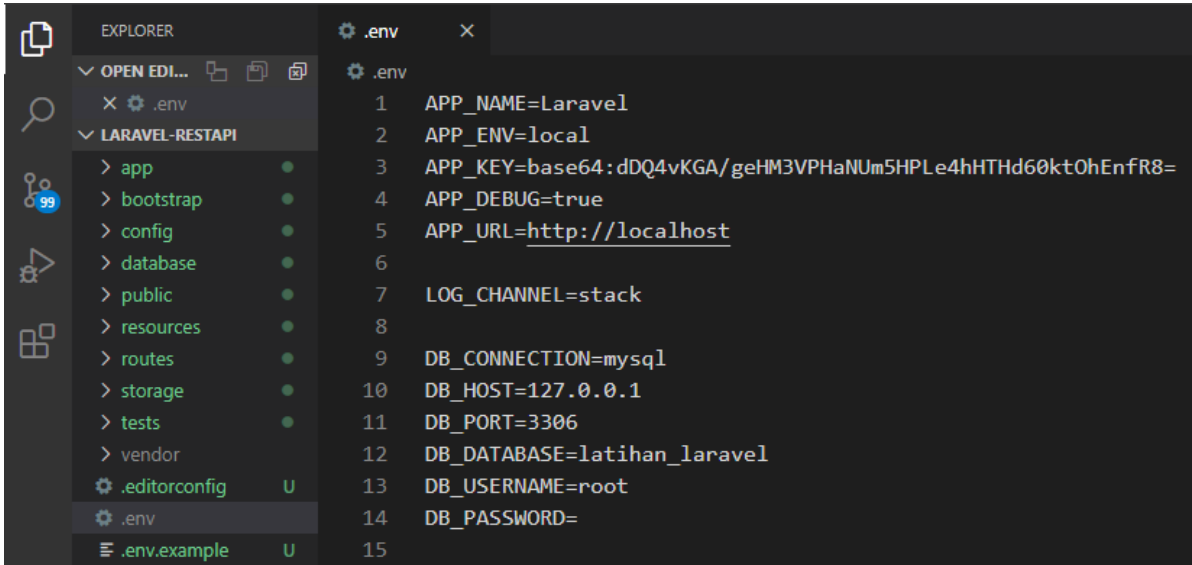


NAMA : DINA RISKY ALIN SAPUTRI
NIM : 1841720016
KELAS : 2B

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2020

Praktikum : Membuat RestFull 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</p>  <p>laravel new laravel-restapi</p>  <p>C:\xampp\htdocs\LARAVEL>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 - Installing doctrine/lexer (1.2.0): Loading from cache - Installing dragonmantank/cron-expression (v2.3.0): Loading from cache - Installing voku/portable-ascii (1.4.10): Loading from cache - Installing symfony/polyfill-ctype (v1.15.0): Loading from cache - Installing phpoption/phpooption (1.7.3): Loading from cache - Installing vlucas/phpdotenv (v4.1.5): Downloading (100%) - Installing symfony/css-selector (v5.0.8): Loading from cache - Installing tijsverkoyen/css-to-inline-styles (2.2.2): Loading from cache - Installing symfony/polyfill-mbstring (v1.15.0): Loading from cache - Installing symfony/var-dumper (v5.0.8): Downloading (connecting...)</p>
2	<p>Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut.</p> <p>cd C:\xampp\htdocs\LARAVEL\laravel-restapi</p>  <p>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> 
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>  <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> 
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. Pertama, kita akan mengubah fungsi index agar saat fungsi index dipanggil, maka aplikasi akan menampilkan seluruh data dari tabel mahasiswa.</p>

```

EXPLORER
└─ OPEN EDITORS
   └─ MahasiswaController.php
      └─ MahasiswaController.php
         └─ index
            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     //fungsi index digunakan untuk menampilkan semua data mahasiswa
            11     public function index()
            12     {
            13         $data = Mahasiswa::all();
            14
            15         //cek data tidak kosong
            16         if (count($data) > 0) {
            17             $res['message'] = "Success!";
            18             $res['values'] = $data;
            19             return response($res);
            20         }
            21         //jika data kosong
            22         else {
            23             $res['message'] = "Kosong!";
            24             return response($res);
            25         }
            26     }
            27 }
            28

```

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).

EXPLORER

OPEN EDITORS

api.php routes

U

LARAVEL-RESTAPI

Middleware

Kernel.php

Providers

Mahasiswa.php

User.php

bootstrap

config

database

public

resources

routes

api.php

channels.php

console.php

web.php

storage

tests

vendor

.editorconfig

.env

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

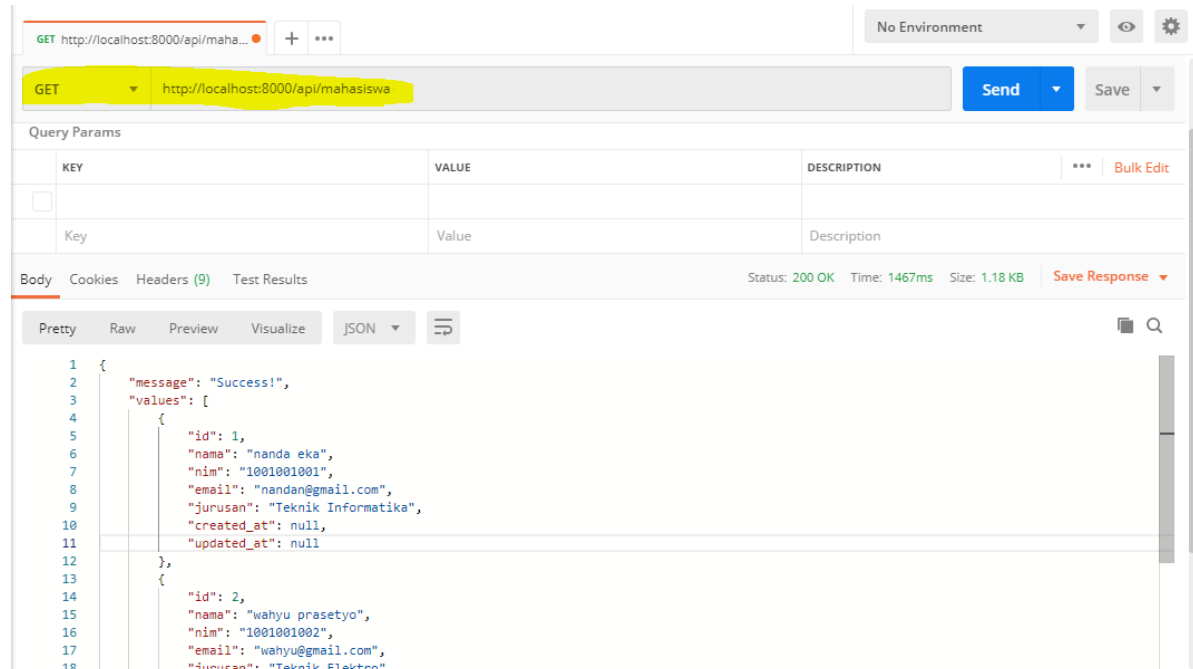
21

Route::get('mahasiswa', 'MahasiswaController@index');

22

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

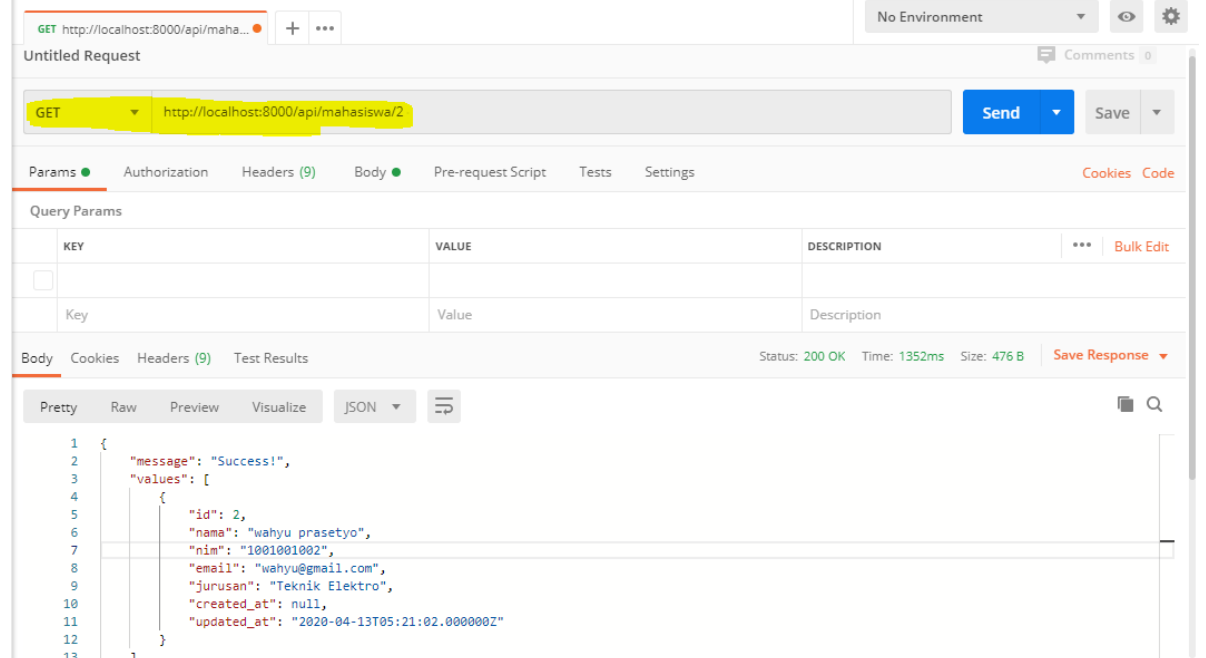
- 8
- Ketikkan perintah **php artisan serve** pada command prompt. Lalu kita coba menguji fungsi untuk menampilkan data menggunakan aplikasi **Postman**. Gunakan perintah **GET**, isikan url : **http://localhost:8000/api/mahasiswa** Berikut adalah tampilan dari aplikasi Postman.



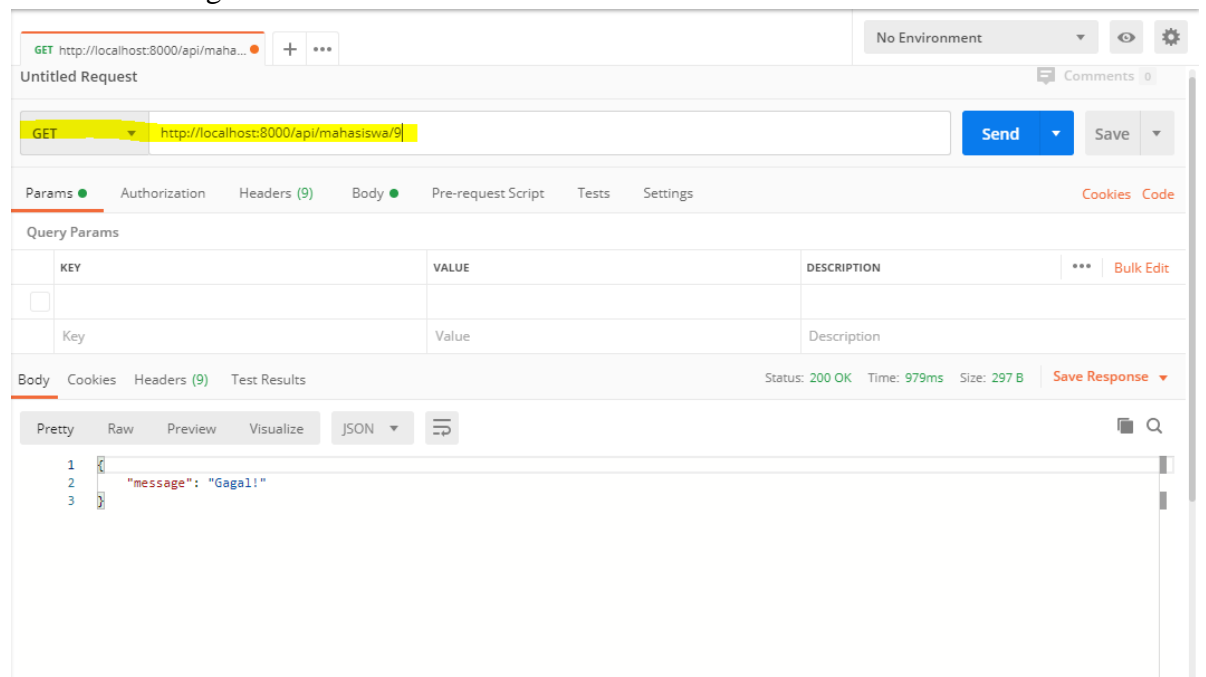
Semua data pada tabel mahasiswa akan tampil, ditampilkan juga pesan sukses.

- 9
- Selanjutnya kita akan menambahkan fungsi untuk melihat suatu data ketika dipilih ID tertentu. Buat fungsi baru yaitu **getId** pada **MahasiswaController.php**.

| | |
|----|--|
| |  <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 | <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> |

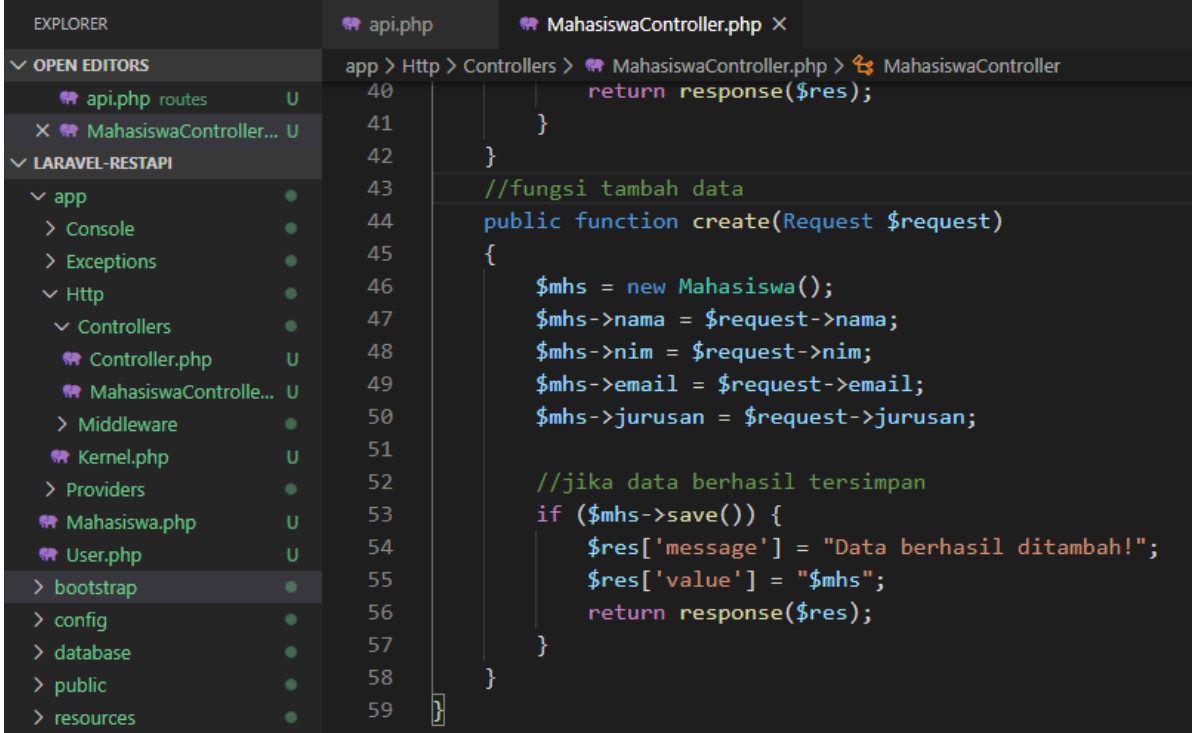
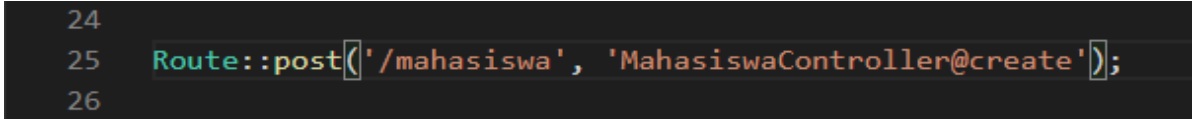


Ketika mencoba menampilkan ID=9 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut.



12

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

| | |
|----|--|
| |  <p>Keterangan:</p> <ul style="list-style-type: none"> • Fungsi create menerima parameter Request yang menampung isian data mahasiswa yang akan ditambahkan ke database. • Line 55-59 : <code>\$mhs->save()</code> digunakan untuk menyimpan data ke database, apabila <code>save()</code> berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang ditambah. |
| 13 | <p>Tambahkan route untuk memanggil fungsi create pada routes/api.php</p>  <p>Karena kita ingin menambah data, maka perintah yang dipakai adalah 'post'.</p> |
| 14 | <p>Kita coba untuk menambahkan data melalui Postman.</p> |

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

Send Save

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

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

| KEY | VALUE | DESCRIPTION |
|---------|--------------------|-------------|
| nama | Eka | |
| nim | 1001001011 | |
| email | eka@gmail.com | |
| jurusan | Teknik Informatika | |
| Key | Value | Description |

Body Cookies Headers (9) Test Results Status: 200 OK Time: 1032ms Size: 533 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "Data berhasil ditambah!",
3   "value": "{\n  \"nama\": \"Eka\",\n  \"nim\": \"1001001011\",\n  \"email\": \"eka@gmail.com\",\n  \"jurusan\": \"Teknik Informatika\",\n  \"updated_at\": \"2020-05-04T10:10:11.000000Z\"\n}"
4 }

```

- 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

Send Save

Body Cookies Headers (9) Test Results Status: 200 OK Time: 1014ms Size: 1.36 KB Save Response

Pretty Raw Preview Visualize JSON

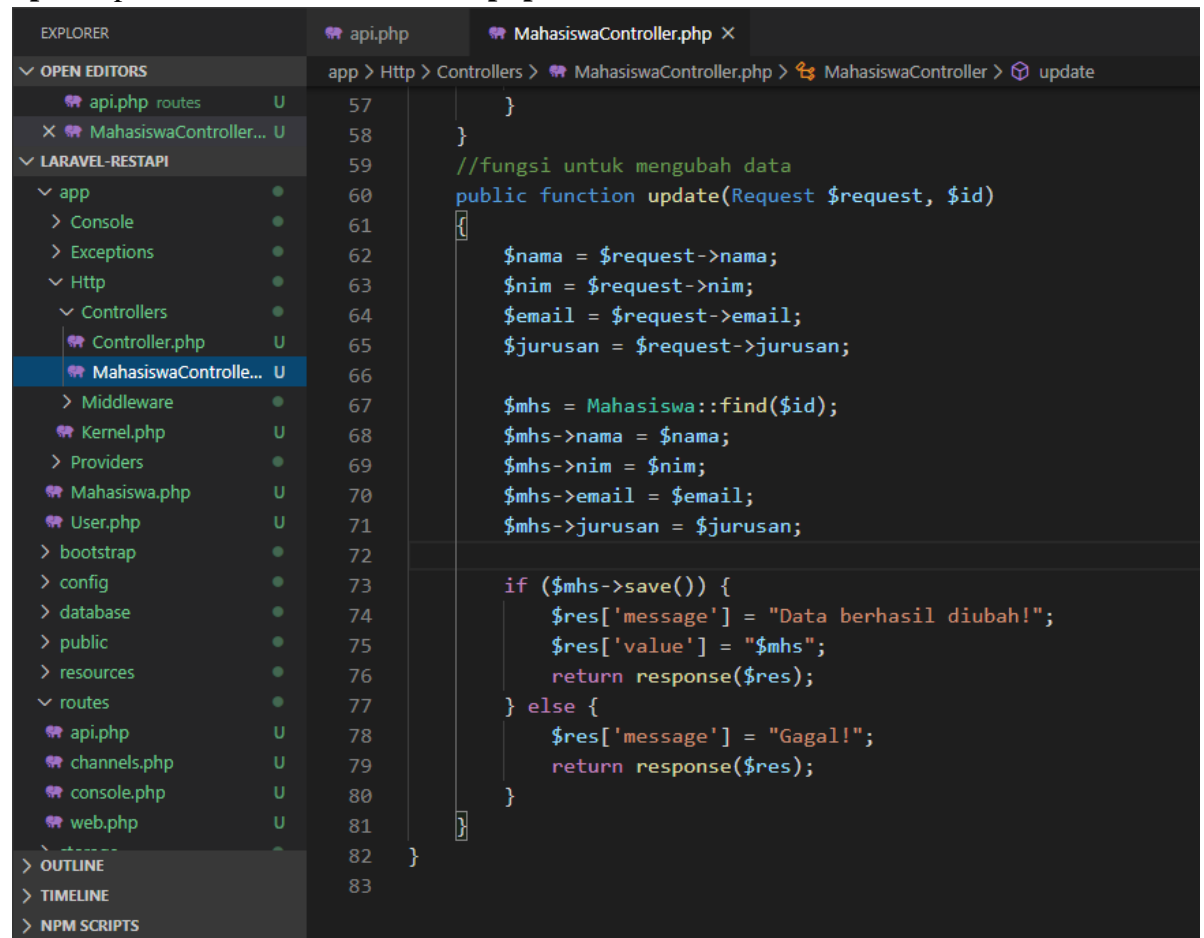
```

43 {
44   "nim": "1001001005",
45   "email": "agitariirawan44@gmail.com",
46   "jurusan": "Teknik Informatika",
47   "created_at": null,
48   "updated_at": null
49 },
50 {
51   "id": 11,
52   "nama": "Agit Ari Irawan",
53   "nim": "1001001005",
54   "email": "agitariirawan44@gmail.com",
55   "jurusan": "Teknik Informatika",
56   "created_at": null,
57   "updated_at": null
58 },
59 {
60   "id": 12,
61   "nama": "Eka",
62   "nim": "1001001011",
63   "email": "eka@gmail.com",
64   "jurusan": "Teknik Informatika",
65   "created_at": "2020-05-04T01:51:55.000000Z",
66   "updated_at": "2020-05-04T01:51:55.000000Z"
67 }

```

15

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



```

EXPLORER
├── OPEN EDITORS
│   ├── api.php routes U
│   └── MahasiswaController... U
├── LARAVEL-RESTAPI
│   ├── app
│   │   ├── Console
│   │   ├── Exceptions
│   │   └── Http
│   │       ├── Controllers
│   │       │   ├── Controller.php U
│   │       │   └── MahasiswaControlle... U
│   │       ├── Middleware
│   │       ├── Kernel.php U
│   │       ├── Providers
│   │       ├── Mahasiswa.php U
│   │       ├── User.php U
│   │       ├── bootstrap
│   │       ├── config
│   │       ├── database
│   │       ├── public
│   │       ├── resources
│   │       └── routes
│   │           ├── api.php U
│   │           ├── channels.php U
│   │           ├── console.php U
│   │           └── web.php U
│   ├── OUTLINE
│   ├── TIMELINE
│   └── NPM SCRIPTS
└── api.php
    └── app > Http > Controllers > MahasiswaController.php > MahasiswaController > update
        57     }
        58     }
        59     //fungsi untuk mengubah data
        60     public function update(Request $request, $id)
        61     {
        62         $nama = $request->nama;
        63         $nim = $request->nim;
        64         $email = $request->email;
        65         $jurusan = $request->jurusan;
        66
        67         $mhs = Mahasiswa::find($id);
        68         $mhs->nama = $nama;
        69         $mhs->nim = $nim;
        70         $mhs->email = $email;
        71         $mhs->jurusan = $jurusan;
        72
        73         if ($mhs->save()) {
        74             $res['message'] = "Data berhasil diubah!";
        75             $res['value'] = "$mhs";
        76             return response($res);
        77         } else {
        78             $res['message'] = "Gagal!";
        79             return response($res);
        80         }
        81     }
        82 }
        83

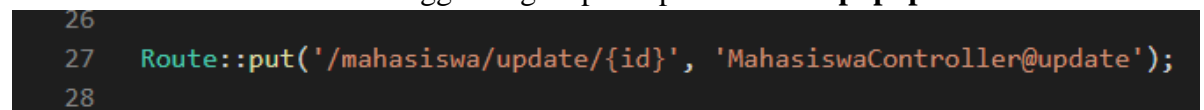
```

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**



```

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

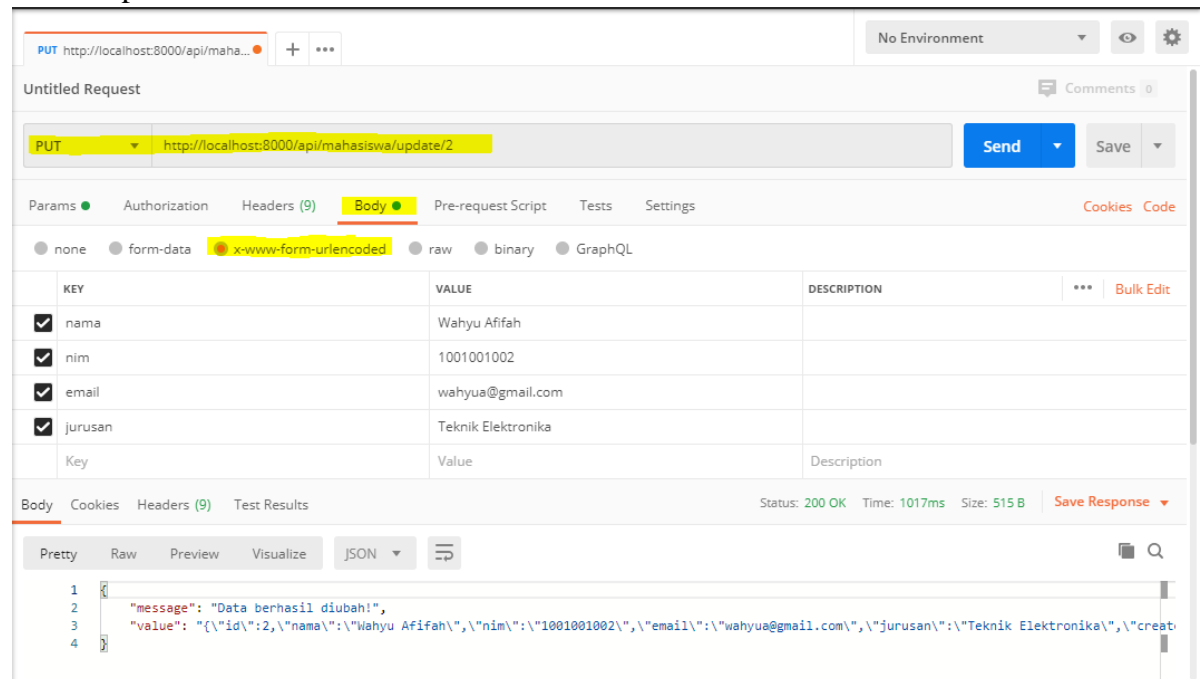
```

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

17

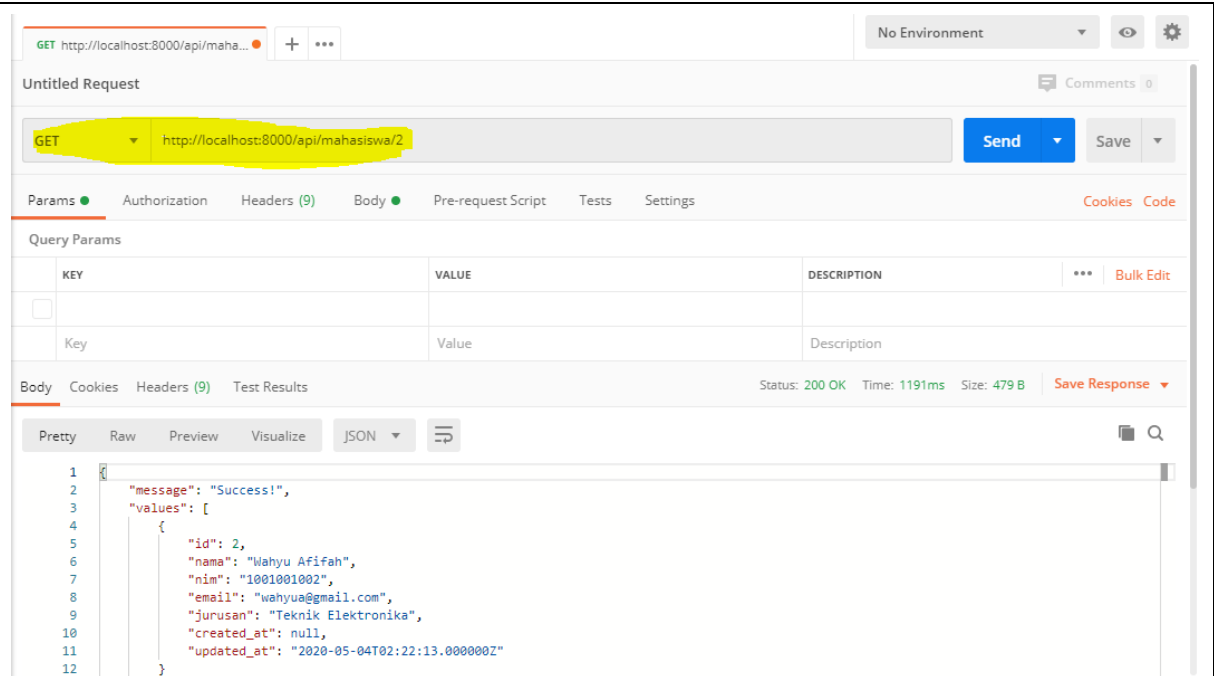
Sekarang kita coba untuk mengubah data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah **PUT** untuk mengubah data.

Berikut adalah contoh untuk mengubah data dengan ID=2, maka url diisi :. Pilih ***http://localhost:8000/api/mahasiswa/update/2*** 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**.



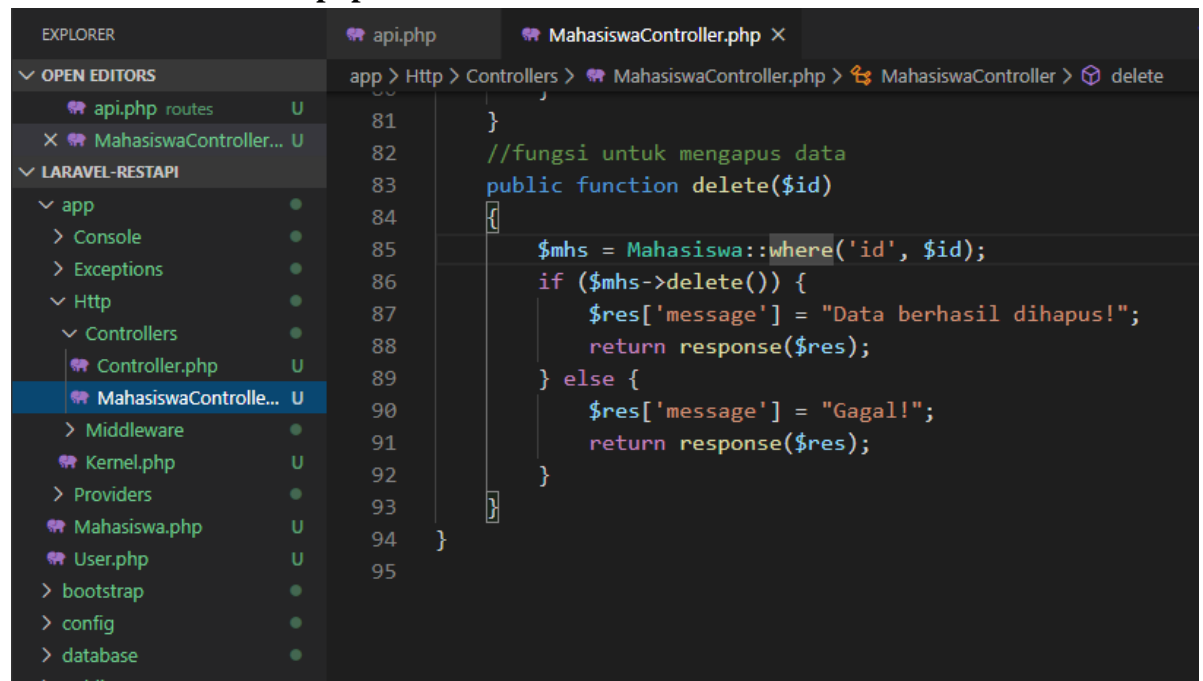
Akan muncul pesan berhasil serta perubahan data dari ID=2.

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**.



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**

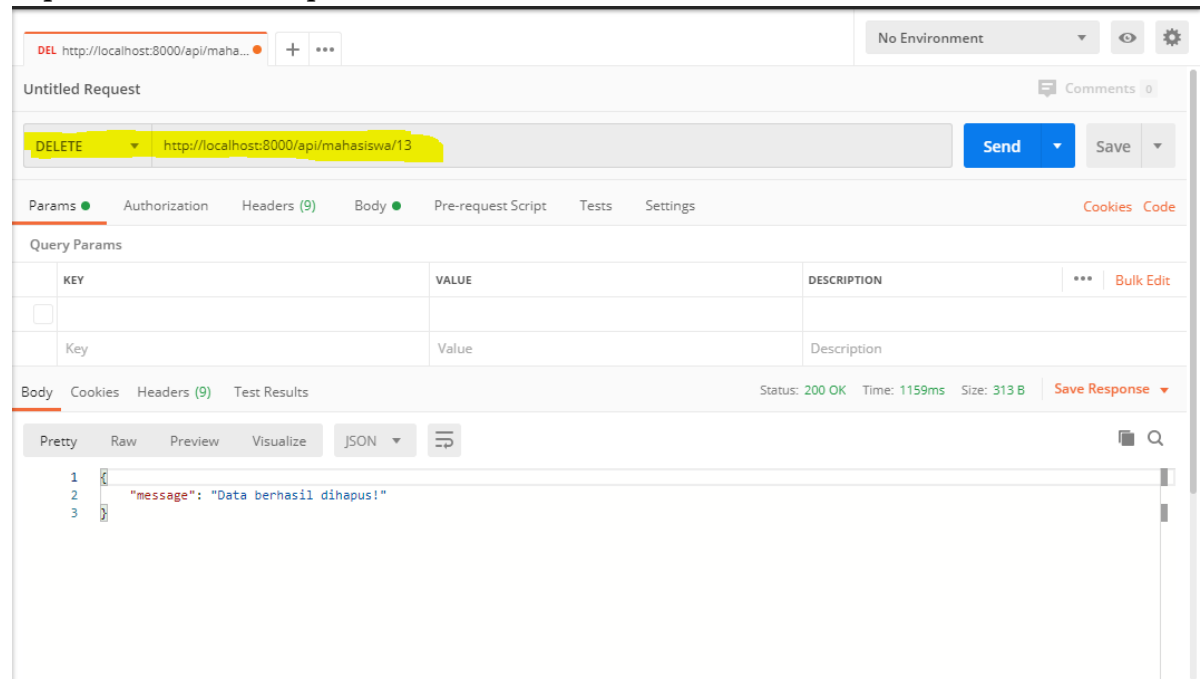
```
28
29 Route::delete('/mahasiswa/{id}', 'MahasiswaController@delete');
30
```

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.