

LAPORAN

PEMROGRAMAN WEB LANJUT

“RESTFUL API LARAVEL”



Oleh:

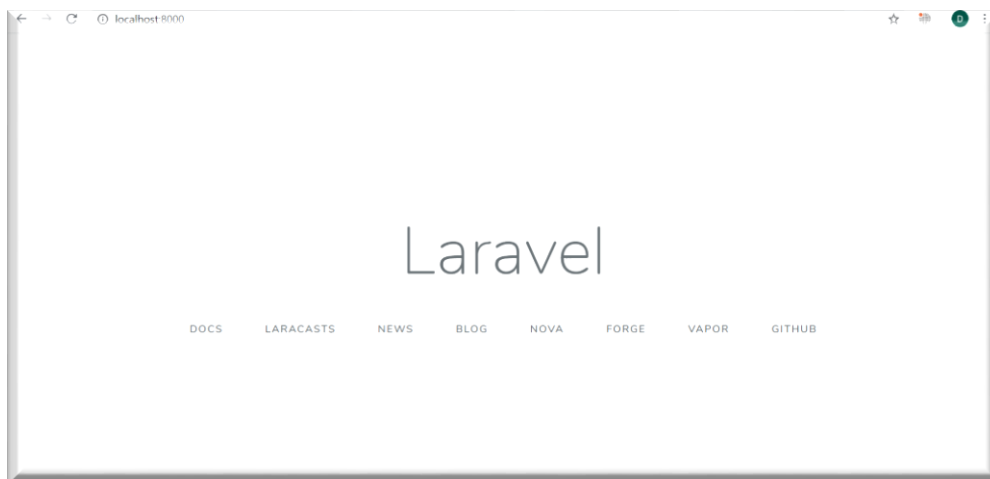
Rio Febriandistra Adi

NIM 1841720199

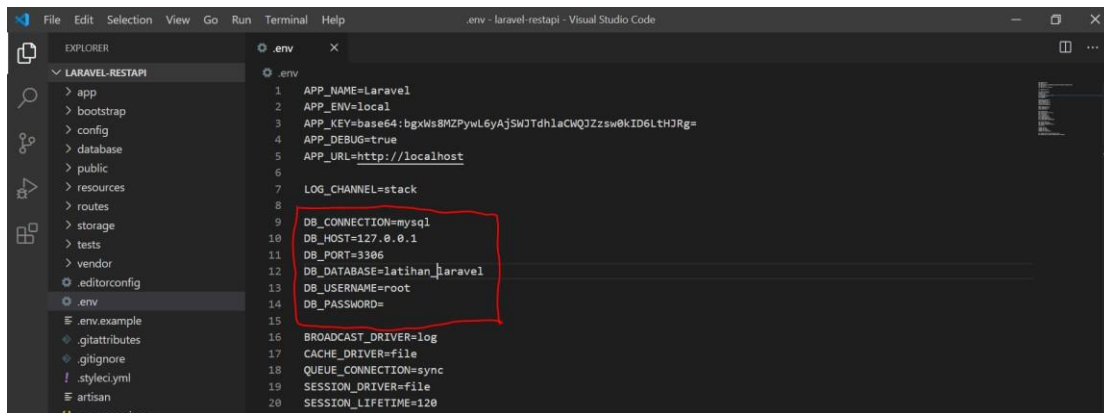
TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

No	LANGKAH LANGKAH
----	-----------------

1.	<p>Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut. cd C:\xampp\htdocs laravel new laravel-restapi</p> <pre> 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 - 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/phpoption (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 (100%) - Installing symfony/routing (v5.0.8): Downloading (100%) - Installing symfony/process (v5.0.8): Downloading (100%) Use the `composer fund` command to find out more! > @php -r "file_exists('.env') copy('.env.example', '.env');" > @php artisan key:generate --ansi Application key set successfully. > Illuminate\Foundation\ComposerScripts::postAutoloadDump > @php artisan package:discover --ansi Discovered Package: facade/ignition Discovered Package: fideloper/proxy Discovered Package: fruitcake/laravel-cors Discovered Package: laravel/tinker Discovered Package: nesbot/carbon Discovered Package: nunomaduro/collision Package manifest generated successfully. Application ready! Build something amazing. </pre>
2.	<p>Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut. cd C:\laravel-restapi php artisan serve Akan tampil halaman default Laravel seperti di bawah ini.</p>



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 **controller**-nya. 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)

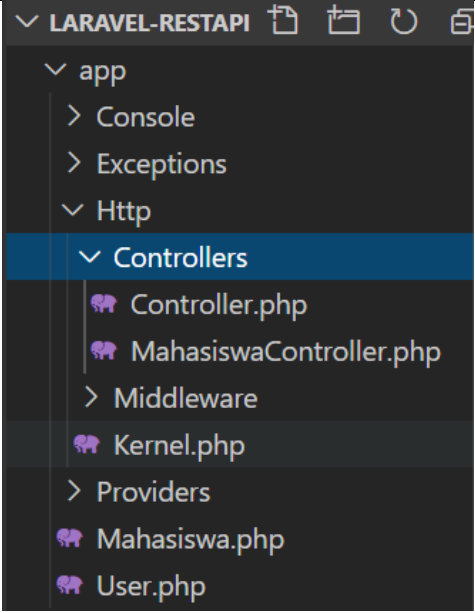
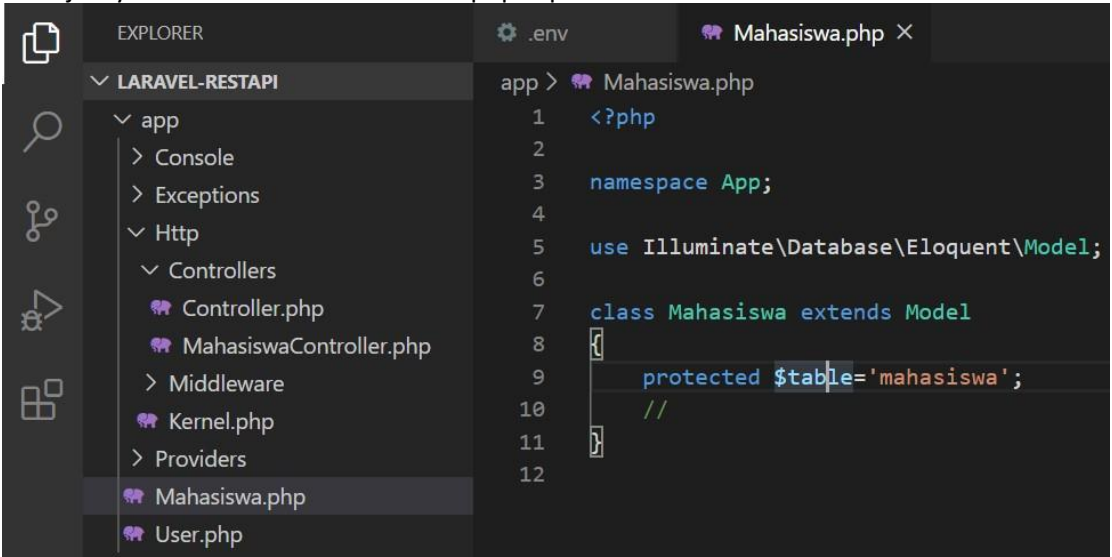
php artisan make:model Mahasiswa -c

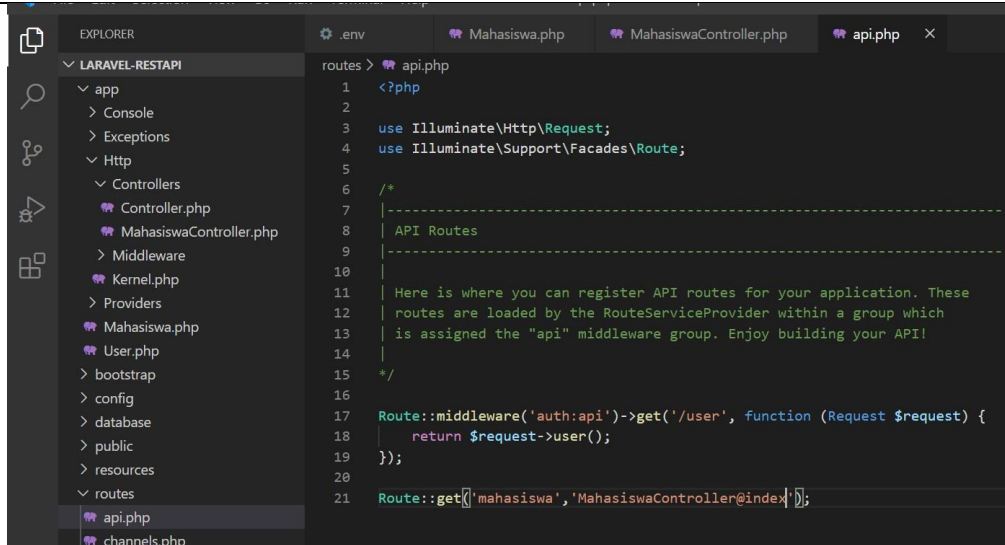
Keterangan : • -c merupakan perintah untuk menyertakan pembuatan controller

Sehingga pada project laravel-restapi akan bertambah dua file yaitu model Mahasiswa.php serta controller MahasiswaController.php.

```
C:\xampp\htdocs\laravel-restapi>php artisan serve
Laravel development server started: http://127.0.0.1:8000
[Mon May 4 11:29:07 2020] 127.0.0.1:5526 [200]: /favicon.ico
^C
C:\xampp\htdocs\laravel-restapi>php artisan make:model Mahasiswa -c
Model created successfully.
Controller created successfully.
```

Controller telah dibuat.

	
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>

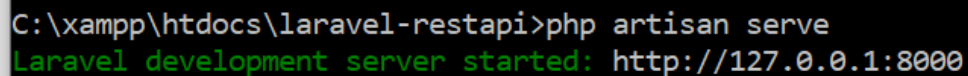


The screenshot shows an IDE with the Explorer panel on the left displaying the project structure for 'LARAVEL-RESTAPI'. The 'routes' folder is expanded, showing 'api.php' and 'channels.php'. The main editor displays the content of 'routes/api.php', which includes PHP code for defining API routes. The code starts with a PHP opening tag, uses 'Illuminate\Http\Request' and 'Illuminate\Support\Facades\Route', and defines a GET route for '/user' that returns the user data. It also includes a comment about registering API routes and a note about the 'api' middleware group.

```
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');
```

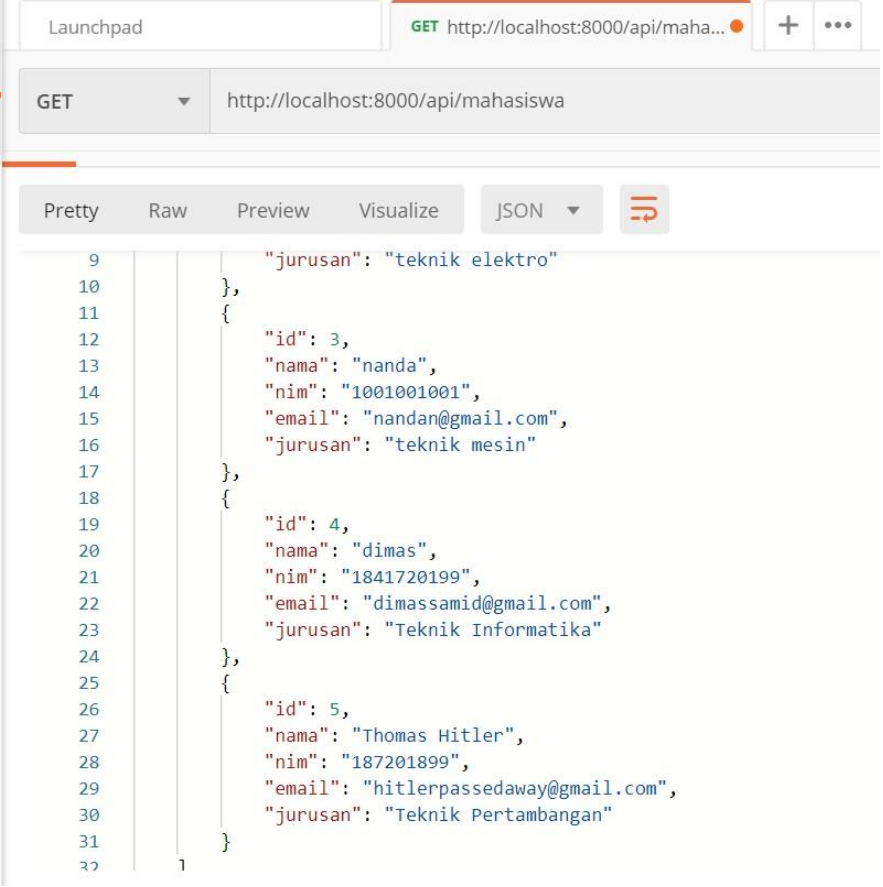
Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'.

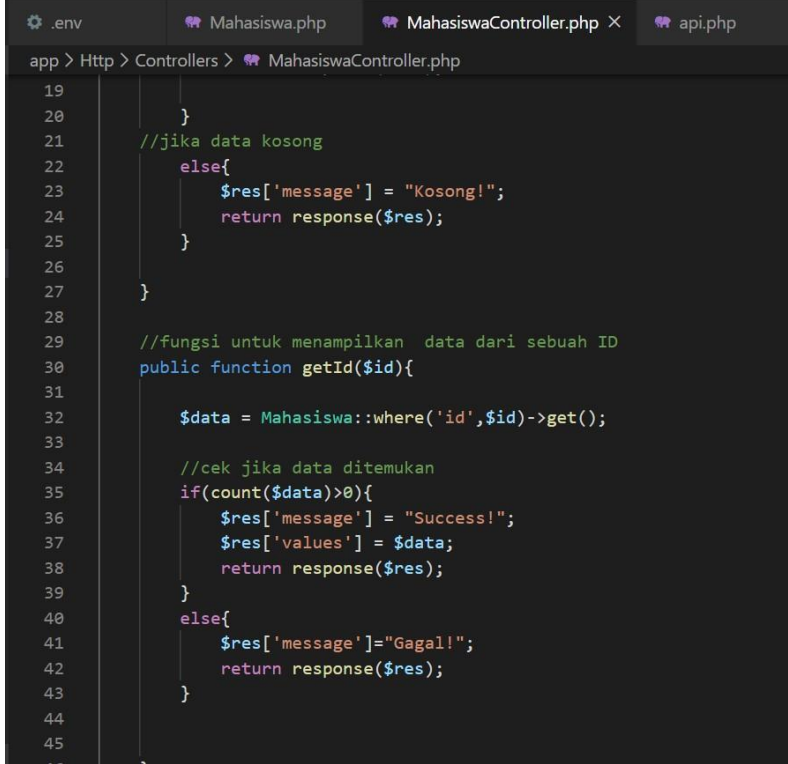
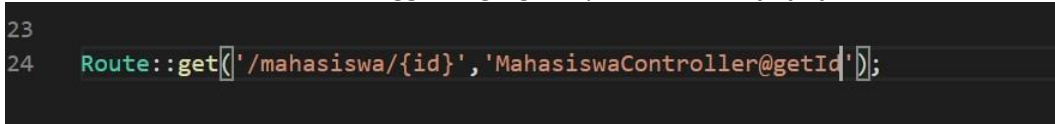
8. Ketikkan perintah **php artisan serve** pada command prompt. Lalu kita coba menguji fungsi untuk menampilkan data menggunakan aplikasi **Postman**. Gunakan perintah **GET**, isikan url : **http://localhost:8000/api/mahasiswa** Berikut adalah tampilan dari aplikasi Postman

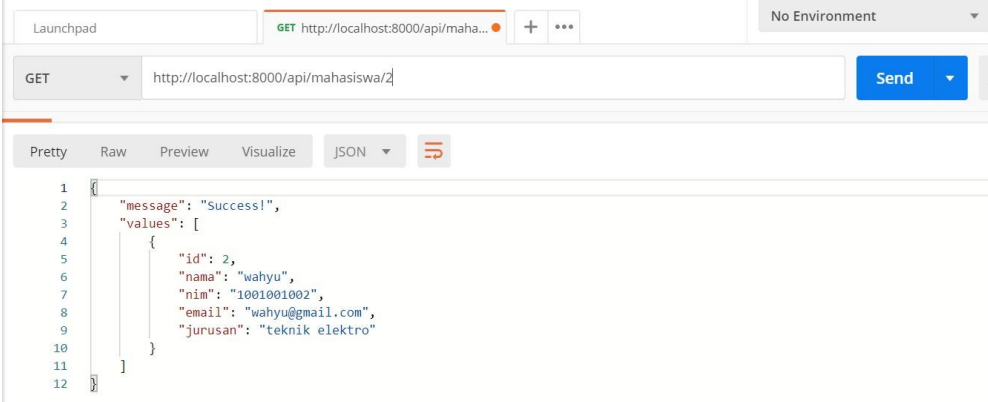
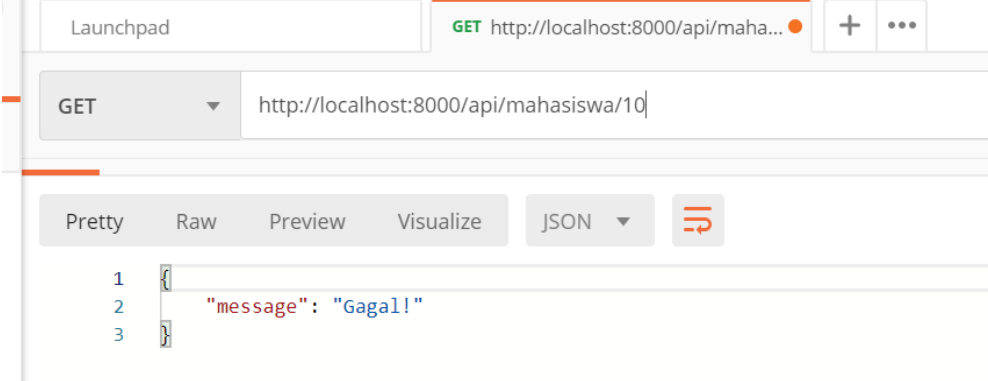


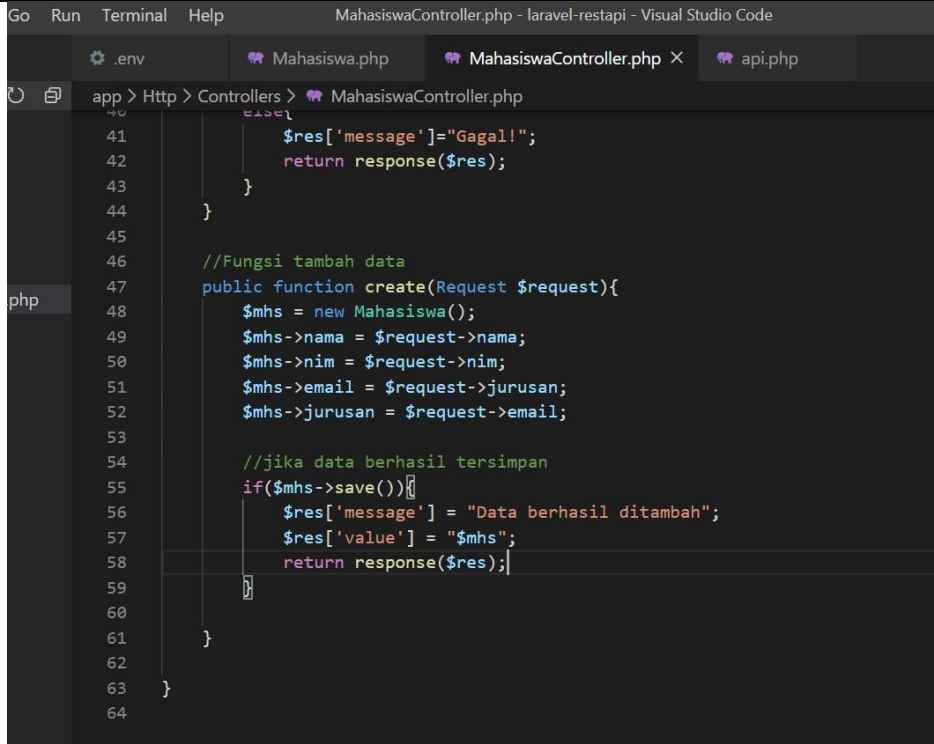
The screenshot shows a command prompt window with the following text:

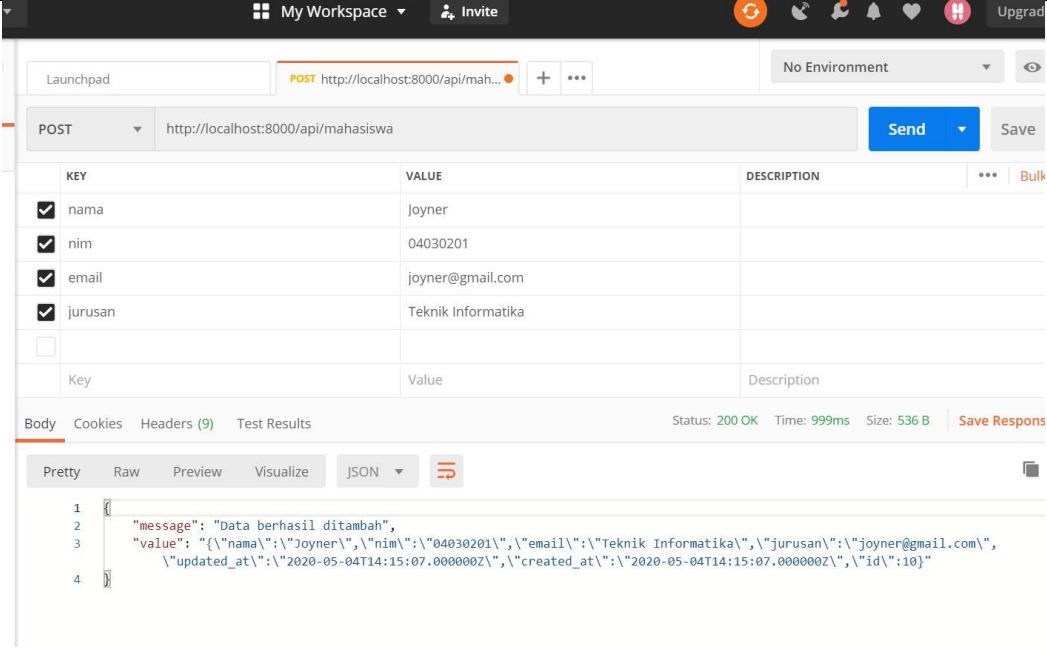
```
C:\xampp\htdocs\laravel-restapi>php artisan serve
Laravel development server started: http://127.0.0.1:8000
```

	 <pre> 9 "jurusan": "teknik elektro" 10 }, 11 { 12 "id": 3, 13 "nama": "nanda", 14 "nim": "1001001001", 15 "email": "nandan@gmail.com", 16 "jurusan": "teknik mesin" 17 }, 18 { 19 "id": 4, 20 "nama": "dimas", 21 "nim": "1841720199", 22 "email": "dimassamid@gmail.com", 23 "jurusan": "Teknik Informatika" 24 }, 25 { 26 "id": 5, 27 "nama": "Thomas Hitler", 28 "nim": "187201899", 29 "email": "hitlerpassedaway@gmail.com", 30 "jurusan": "Teknik Pertambangan" 31 } 32] </pre> <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>

	<div data-bbox="269 195 1052 955">  <pre> 19 } 20 } 21 //jika data kosong 22 else{ 23 \$res['message'] = "Kosong!"; 24 return response(\$res); 25 } 26 } 27 28 29 //fungsi untuk menampilkan data dari sebuah ID 30 public function getId(\$id){ 31 32 \$data = Mahasiswa::where('id',\$id)->get(); 33 34 //cek jika data ditemukan 35 if(count(\$data)>0){ 36 \$res['message'] = "Success!"; 37 \$res['values'] = \$data; 38 return response(\$res); 39 } 40 else{ 41 \$res['message']="Gagal!"; 42 return response(\$res); 43 } 44 } 45 </pre> </div> <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> <div data-bbox="269 1140 1317 1262">  <pre> 23 24 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId'); </pre> </div> <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 : http://localhost:8000/api/mahasiswa/2</p> <p>HASIL:</p>

	<div data-bbox="272 205 1253 604">  <pre> 1 { 2 "message": "Success!", 3 "values": [4 { 5 "id": 2, 6 "nama": "wahyu", 7 "nim": "1001001002", 8 "email": "wahyu@gmail.com", 9 "jurusan": "teknik elektro" 10 } 11] 12 }</pre> </div> <p data-bbox="272 619 1312 688">Ketika mencoba menampilkan ID=10 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut</p> <div data-bbox="272 699 1253 1077">  <pre> 1 { 2 "message": "Gagal!" 3 }</pre> </div>
12	<p>Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php.</p>

	<div data-bbox="267 191 1195 930">  <pre> 41 \$res['message'] = "Gagal!"; 42 return response(\$res); 43 } 44 } 45 46 //Fungsi tambah data 47 public function create(Request \$request){ 48 \$mhs = new Mahasiswa(); 49 \$mhs->nama = \$request->nama; 50 \$mhs->nim = \$request->nim; 51 \$mhs->email = \$request->jurusan; 52 \$mhs->jurusan = \$request->email; 53 54 //jika data berhasil tersimpan 55 if(\$mhs->save()){ 56 \$res['message'] = "Data berhasil ditambah"; 57 \$res['value'] = "\$mhs"; 58 return response(\$res); 59 } 60 61 } 62 63 } 64 </pre> </div> <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 : \$mhs->save() digunakan untuk menyimpan data ke database, apabila save() berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang ditambahkan.
13	<div data-bbox="267 1192 1203 1295"> <pre> 26 27 Route::post('/mahasiswa', 'MahasiswaController@create'); </pre> </div> <p>Karena kita ingin menambah data, maka perintah yang dipakai adalah 'post'</p>
14	<p>Kita coba untuk menambahkan data melalui Postman.</p>

	
<p>15</p>	<p>Selanjutnya kita akan menambahkan fungsi untuk mengubah data dari database. Buat fungsi update pada MahasiswaController.php.</p> <pre> 1 2 public function update(Request \$request, \$id){ 3 \$nama = \$request -> nama; 4 \$nim = \$request -> nim; 5 \$email = \$request -> email; 6 \$jurusan = \$request -> jurusan; 7 8 \$mhs = Mahasiswa::find(\$id); 9 \$mhs->nama = \$nama; 10 \$mhs->nim = \$nim; 11 \$mhs->email = \$email; 12 \$mhs->jurusan = \$jurusan; 13 14 if(\$mhs->save()){ 15 \$res['message'] = "Data berhasil ditambah!"; 16 \$res['value'] = "\$mhs"; 17 return response(\$res); 18 } 19 else{ 20 \$res['message']="Gagal!"; 21 return response(\$res); 22 } 23 } 24 } </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> • 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.
<p>16</p>	<p>Tambahkan route untuk memanggil fungsi update pada routes/api.php</p>

29

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

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. The URL is `http://localhost:8000/api/mahasiswa/update/2`. 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'. The data being updated is as follows:

KEY	VALUE	DESCRIPTION
nama	lukas yanto	
nim	04030201	
email	lukasmli@gmail.com	
jurusan	Teknik Informatika	

Below the table, the raw JSON response is shown:

```

1 {
2   "message": "Data berhasil ditambah!",
3   "value": "{\n\"id\":2,\n\"nama\":\n\"lukas yanto\", \n\"nim\":\n\"04030201\", \n\"email\":\n\"lukasmli@gmail.com\", \n\"jurusan\":\n\"Teknik Informatika\", \n\"created_at\":null, \n\"updated_at\":\n\"2020-05-04T14:33:10.000000Z\"}"
}
```

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.

```

},
{
  "id": 2,
  "nama": "lukas yanto",
  "nim": "04030201",
  "email": "lukasmli@gmail.com",
  "jurusan": "Teknik Informatika",
  "created_at": null,
  "updated_at": "2020-05-04T14:33:10.000000Z"
},
{
  "id": 3,
  "nama": "adrian",
  "nim": "04030202",
  "email": "adrian@gmail.com",
  "jurusan": "Teknik Informatika",
  "created_at": null,
  "updated_at": "2020-05-04T14:33:10.000000Z"
}
```

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

```

public function delete($id){
    $mhs = Mahasiswa::where('id',$id);

    if($mhs->delete()){
        $res['message'] = "Data berhasil dihapus!";
        return response($res);
    }else{
        $res['message'] = "Gagal!";
        return response($res);
    }
}

```

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

```

32
33 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=2, maka url diisi :

<http://localhost:8000/api/mahasiswa/2>

