



LAPORAN PRAKTIKUM PEMROGRAMAN DASAR WEB

NAMA : NUR HANIFAH
NIM : 1841720062
Kelas : TI 2B
Isi Laporan : Jobsheet14 – Membuat RESTful API 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. cd C:\xampp\htdocs laravel new laravel-restapi</p> <pre>Microsoft Windows [Version 10.0.18363.778] (c) 2019 Microsoft Corporation. All rights reserved. C:\Users\ASUS A405U>cd C:\xampp\htdocs C:\xampp\htdocs>composer create-project --prefer-dist laravel/laravel laravel-restapi Installing laravel/laravel (v7.6.0) - Installing laravel/laravel (v7.6.0): Downloading (100%) Created project in laravel-restapi > @php -r "file_exists('.env') copy('.env.example', '.env');" Loading composer repositories with package information Updating dependencies (including require-dev)</pre> <p>Lalu kita cek di dalam C:xampp->htdocs->laravel-restapi</p> <ul style="list-style-type: none">laravel-crud-kedualaravel-restapilatihan web
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>

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

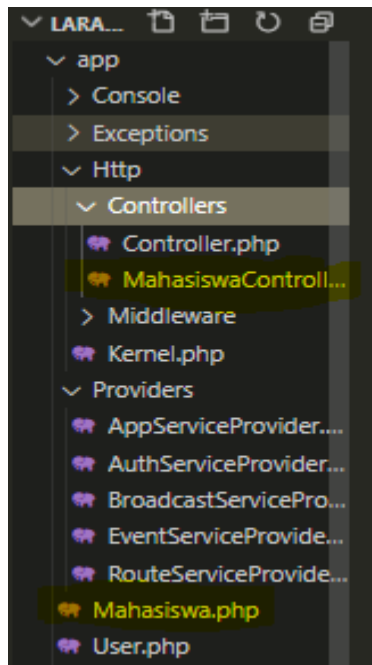
```
Microsoft Windows [Version 10.0.18363.778]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\ASUS A405U>cd C:\xampp\htdocs

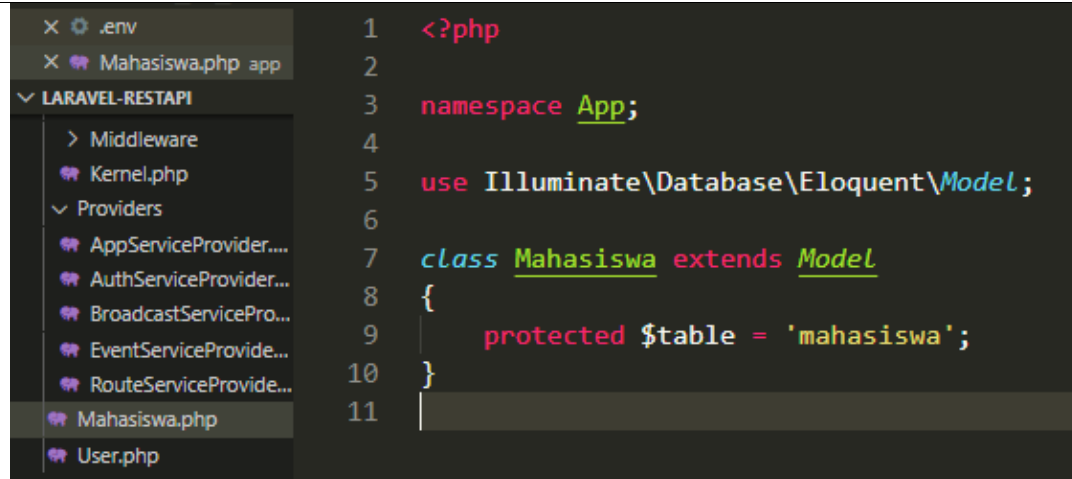
C:\xampp\htdocs>cd laravel-restapi

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

C:\xampp\htdocs\laravel-restapi>
```



5. Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.

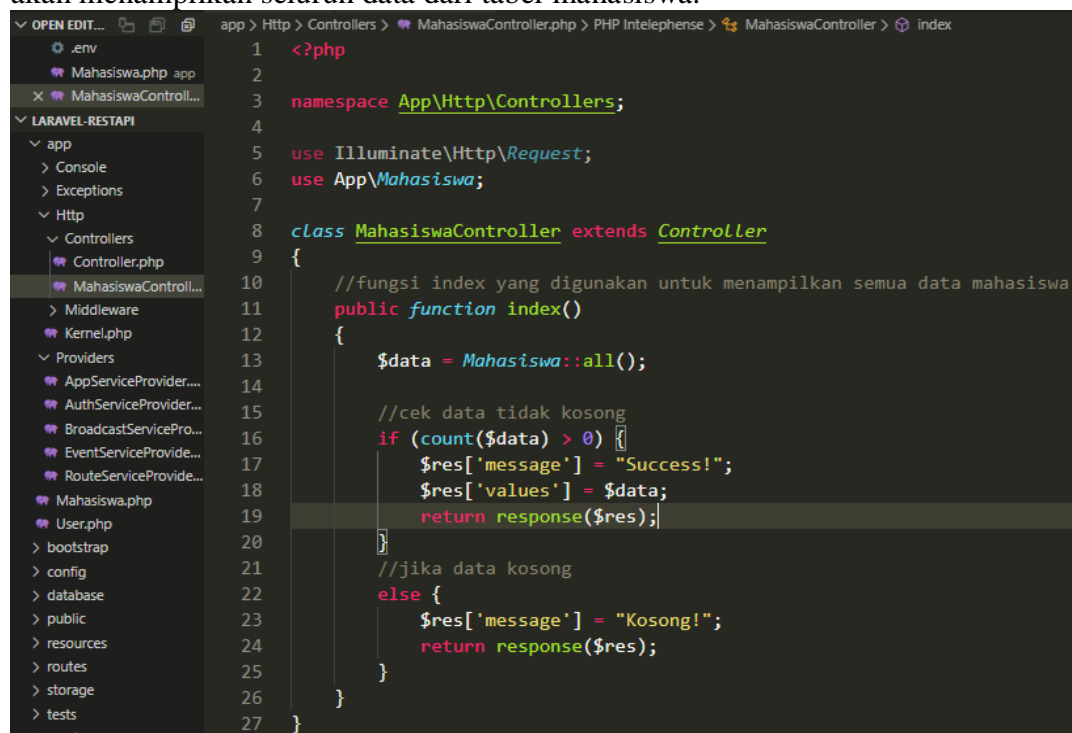


```

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

```

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.



```

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 yang 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 }

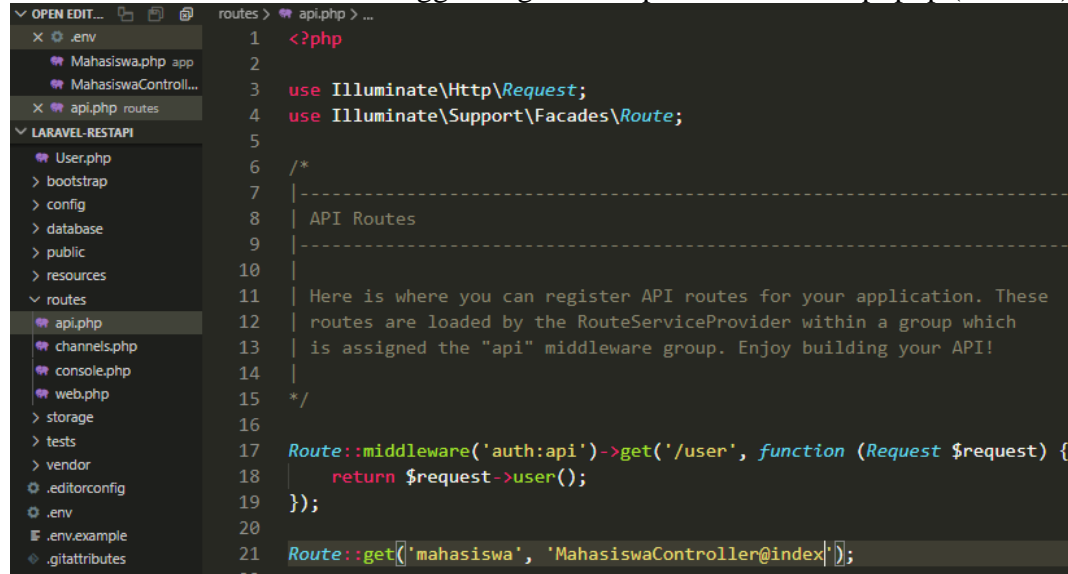
```

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

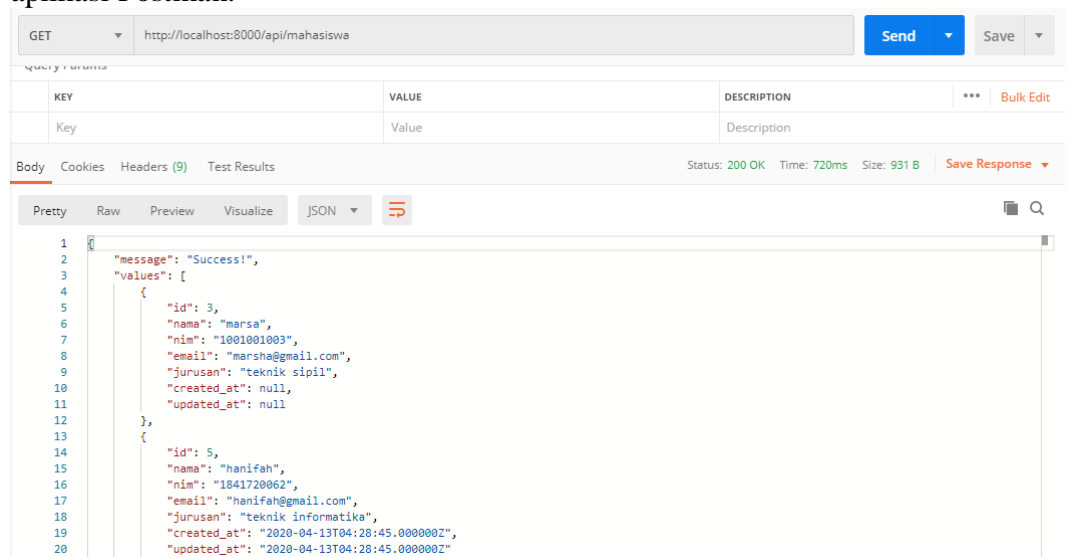
7. Tambahkan route untuk memanggil fungsi index pada file routes/api.php (Line 21).



```
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');
```

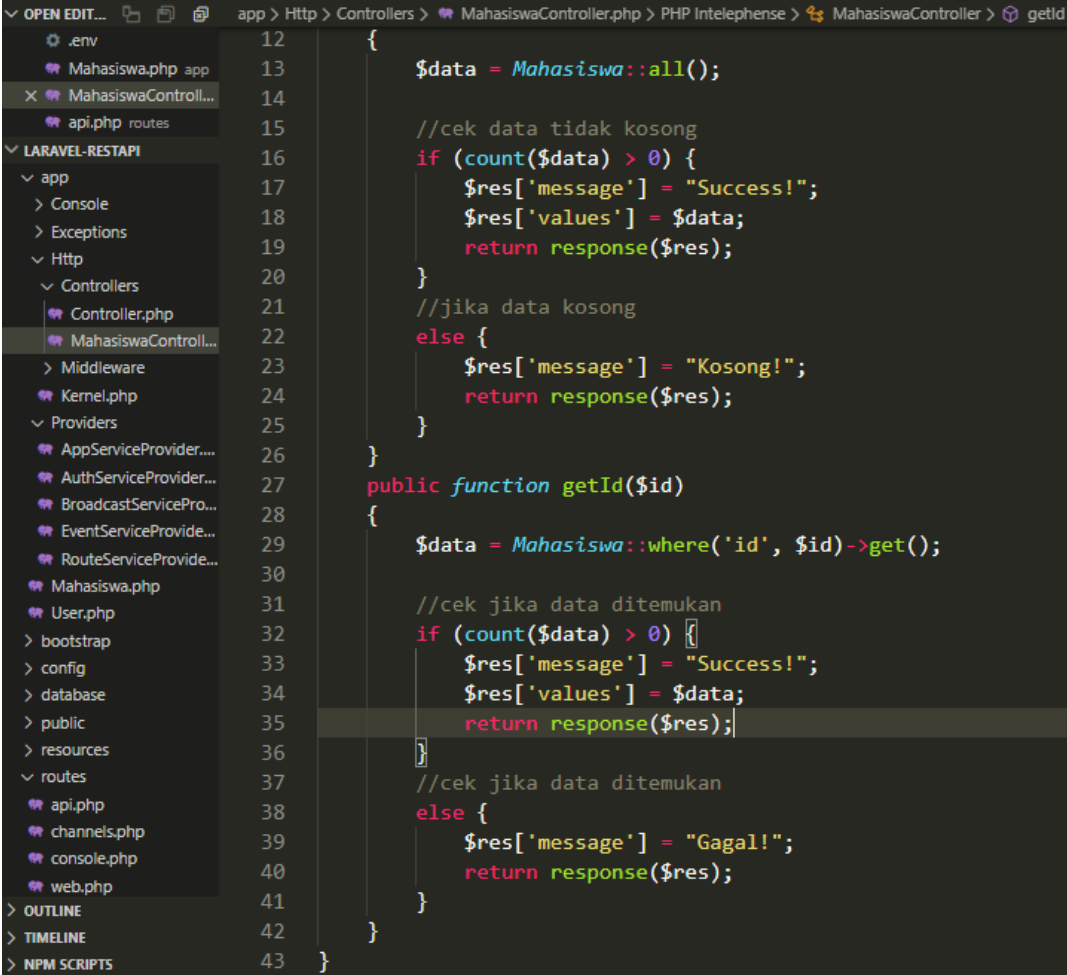
Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'.

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.



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.

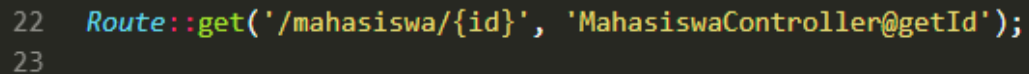


```
12 {
13     $data = Mahasiswa::all();
14
15     //cek data tidak kosong
16     if (count($data) > 0) {
17         $res['message'] = "Success!";
18         $res['values'] = $data;
19         return response($res);
20     }
21     //jika data kosong
22     else {
23         $res['message'] = "Kosong!";
24         return response($res);
25     }
26 }
27 public function getId($id)
28 {
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     //cek jika data 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

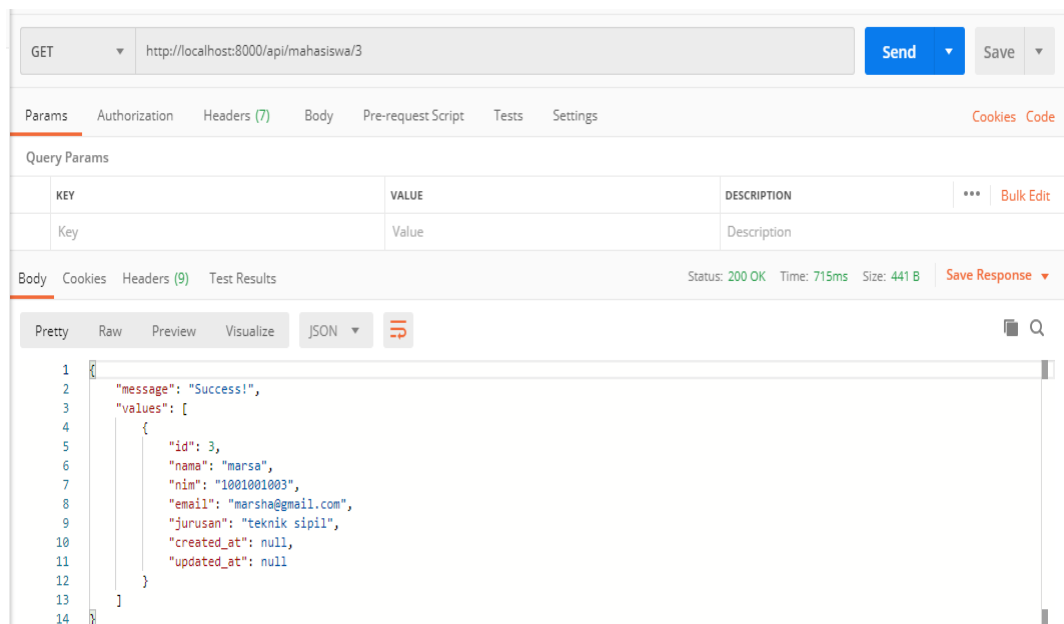


```
22 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId');
23
```

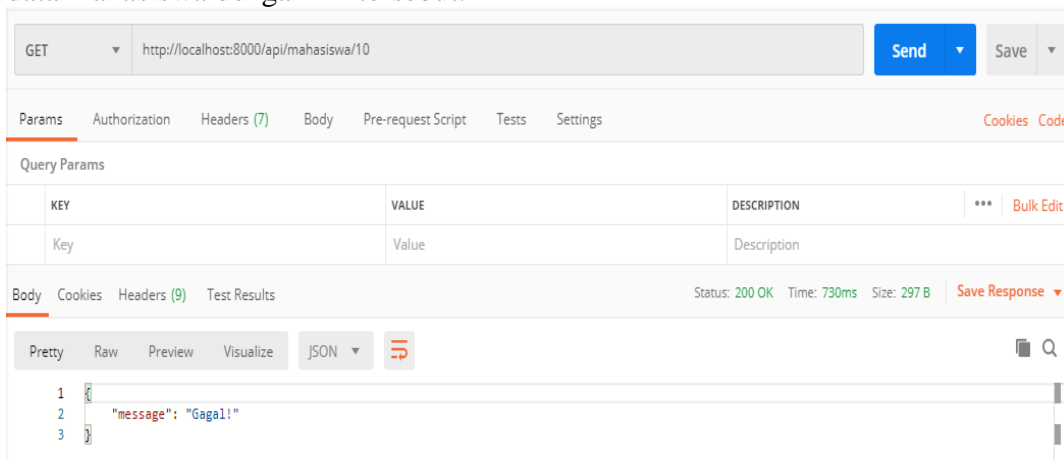
Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'

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=3, maka url diisi :
<http://localhost:8000/api/mahasiswa/3>



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.

```

//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 disimpan
    if ($mhs->save()) {
        $res['message'] = "Data berhasil Ditambah!";
        $res['value'] = "$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

```

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

```

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

14. Kita coba untuk menambahkan data melalui Postman.

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

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
<input checked="" type="checkbox"/> id	7	
<input checked="" type="checkbox"/> nama	Ning Hanifah99	
<input checked="" type="checkbox"/> nim	10010099	
<input checked="" type="checkbox"/> email	hanifah99@gmail.com	
<input checked="" type="checkbox"/> jurusan	teknik informatika	
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 757ms Size: 548 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "Data berhasil Ditambah!",
3   "value": "{\n\"nama\": \"Ning Hanifah99\", \"nim\": \"10010099\", \"email\": \"hanifah99@gmail.com\", \"jurusan\": \"teknik informatika\",
4     \"updated_at\": \"2020-05-03T22:39:37.000000Z\", \"created_at\": \"2020-05-03T22:39:37.000000Z\", \"id\": 13}"

```

- 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.

```

25     "nim": "1841720062",
26     "email": "hanifah11235@gmail.com",
27     "jurusan": "teknik informatika",
28     "created_at": null,
29     "updated_at": null
30   },
31   {
32     "id": 12,
33     "nama": "NingHanifah2",
34     "nim": "1841720062",
35     "email": "hanifah987@gmail.com",
36     "jurusan": "teknik informatika",
37     "created_at": null,
38     "updated_at": null
39   },
40   {
41     "id": 13,
42     "nama": "Ning Hanifah99",
43     "nim": "18010099",
44     "email": "hanifah99@gmail.com",
45     "jurusan": "teknik informatika",
46     "created_at": "2020-05-03T22:39:37.000000Z",
47     "updated_at": "2020-05-03T22:39:37.000000Z"
48   }
49 ]
50

```

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

```

59 //fungsi 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

```

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

```
24 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');
25
```

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=13, maka url diisi : <http://localhost:8000/api/mahasiswa/update/13>. 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/13>. The request body is x-www-form-urlencoded with the following data:

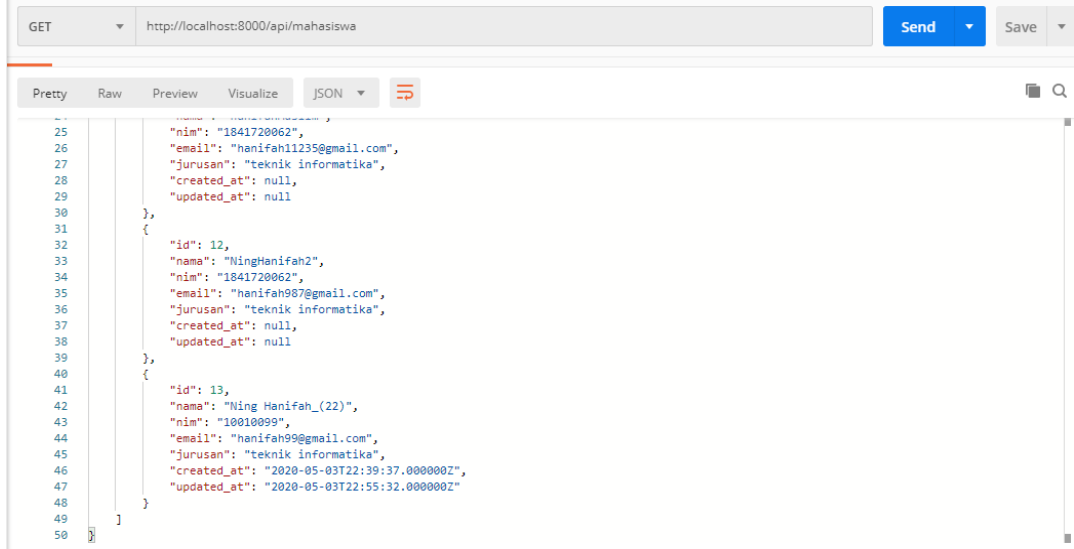

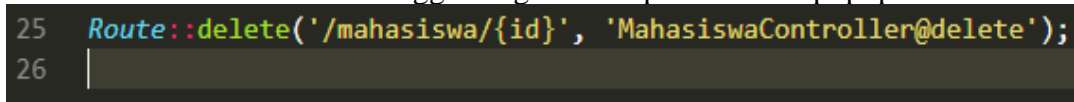
KEY	VALUE
id	7
nama	Ning Hanifah_(22)
nim	10010099
email	hanifah99@gmail.com
jurusan	teknik informatika
Key	Value

The response is a JSON object with the following content:

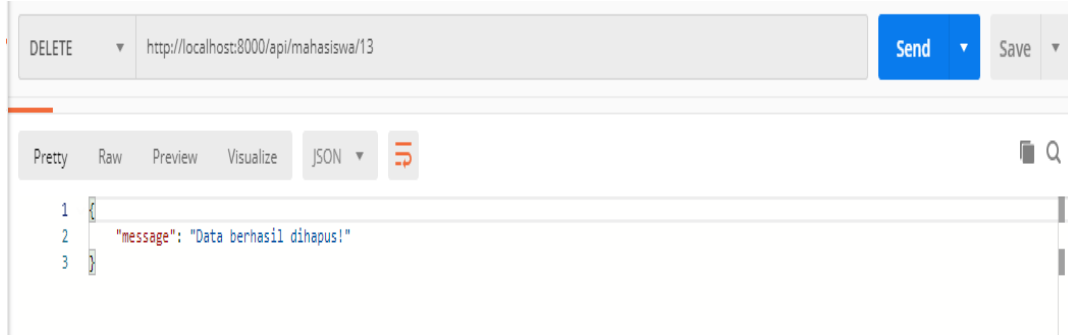
```
{
  "message": "Data berhasil Diubah!",
  "value": {
    "id": "13",
    "nama": "Ning Hanifah_(22)",
    "nim": "10010099",
    "email": "hanifah99@gmail.com",
    "jurusan": "teknik informatika",
    "created_at": "2020-05-03T22:39:37.000000Z",
    "updated_at": "2020-05-03T22:55:32.000000Z"
  }
}
```

Akan muncul pesan berhasil serta perubahan data dari ID=13.(Merubah nama:Ning Hanifah Menjadi Ning Hanifah_(22))

Kemudian coba untuk menampilkan data dengan ID=13 untuk melihat apakah data sudah ter-update.

	
18.	<p>Terakhir kita akan membuat fungsi untuk menghapus data dengan nama delete di MahasiswaController.php.</p>  <p>Keterangan:</p> <ul style="list-style-type: none"> • 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.	<p>Tambahkan route untuk memanggil fungsi delete pada routes/api.php</p>  <p>Karena kita ingin menghapus data, maka perintah yang dipakai adalah 'delete'.</p>
20.	<p>Buka Postman untuk mencoba menghapus data dari database. Gunakan perintah DELETE untuk mengubah data.</p> <p>Berikut adalah contoh untuk menghapus data dengan ID=13, maka url diisi :</p>

http://localhost:8000/api/mahasiswa /13



Muncul pesan berhasil ketika data terhapus dari database.

-- Selamat Mengerjakan --