

LAPORAN PRAKTIKUM

Pemrograman Web Lanjut

Laravel Rest-API



Fidela Trisaktiyani
1841720211
TI-2B

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG



Topik

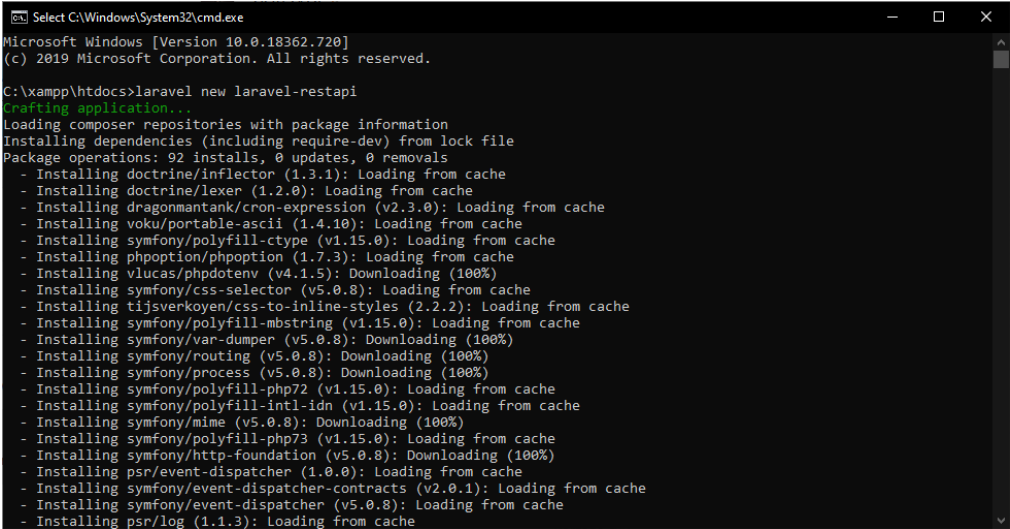
Membuat RESTful API dengan Framework Laravel

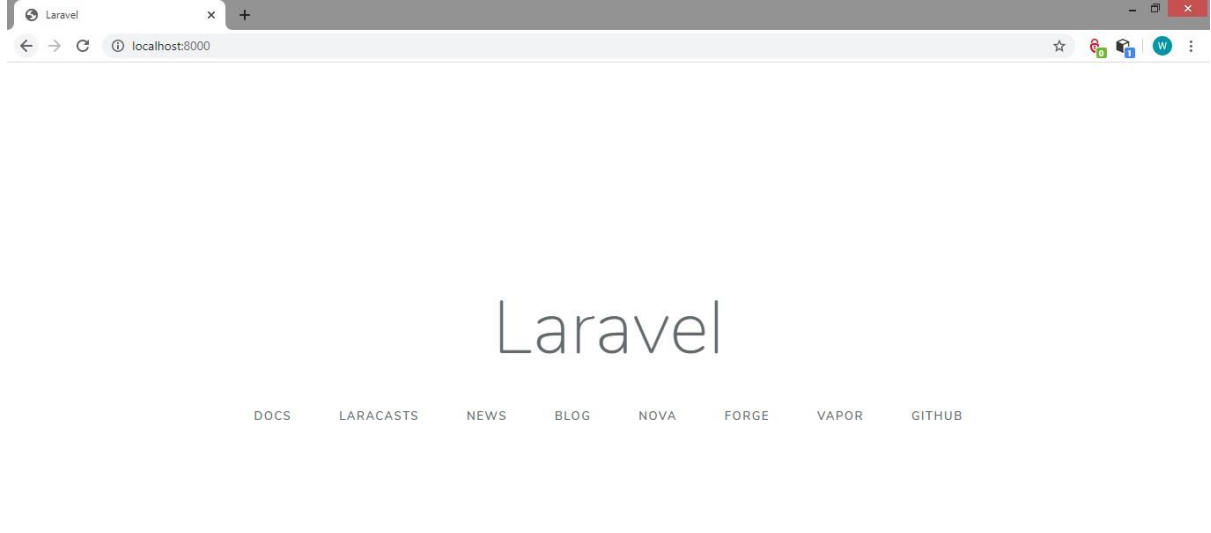
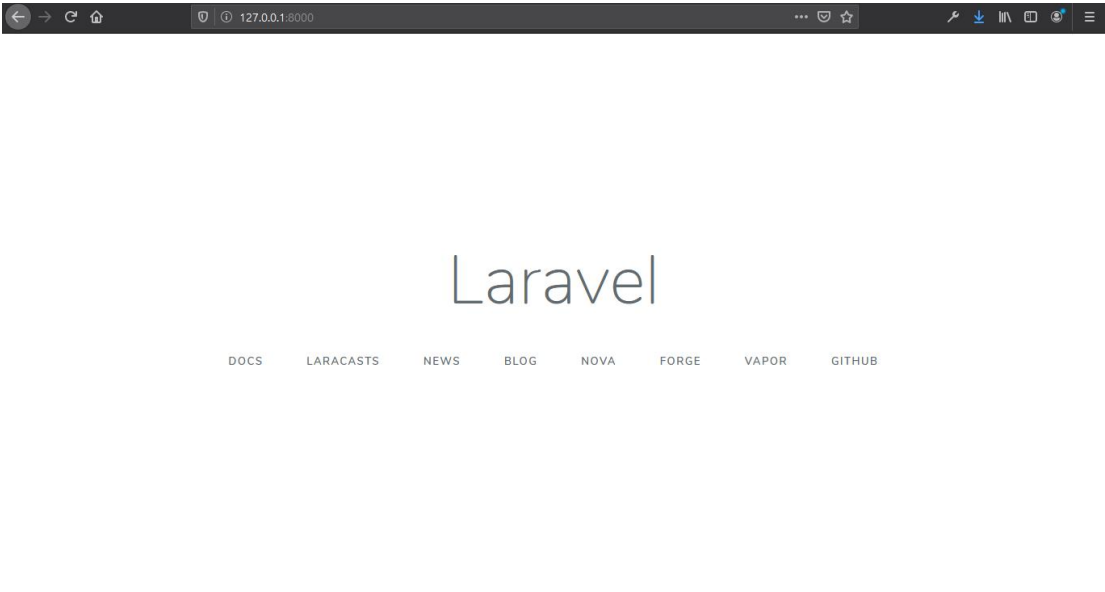
Tujuan

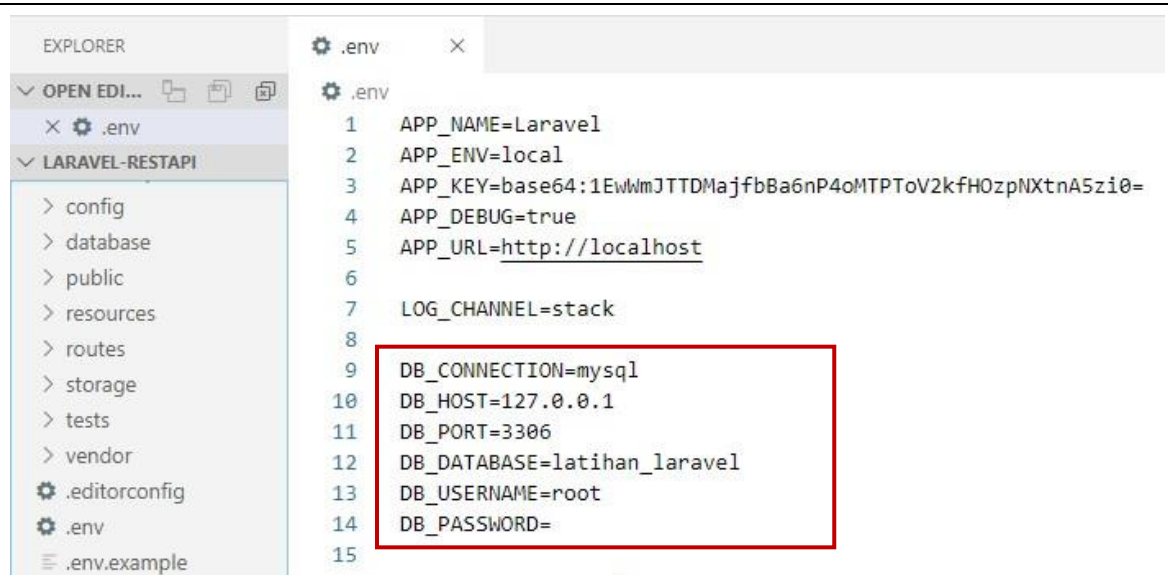
Mahasiswa diharapkan dapat:

1. Memahami bagaimana cara membuat RESTful API menggunakan Laravel

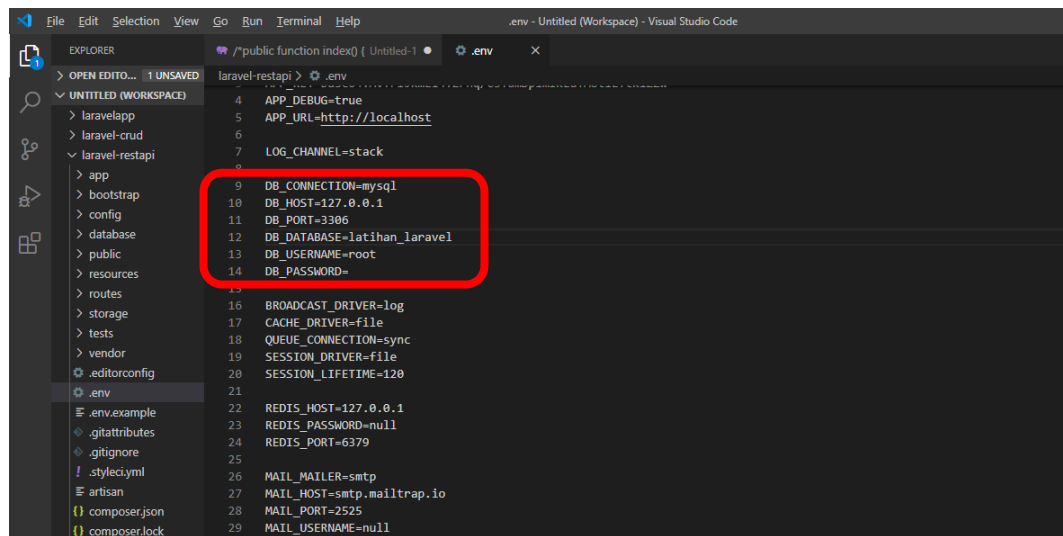
Praktikum: Membuat RESTful API di Laravel

Langkah	Keterangan
1	<p>Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut.</p> <pre>cd C:\xampp\htdocs laravel new laravel-restapi</pre> <p>Laporan:</p> 

2	<p>Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut.</p> <p>cd C:\laravel-restapi php artisan serve</p> <p>Akan tampil halaman default Laravel seperti di bawah ini.</p>  <p>Laporan:</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>



Laporan:



4

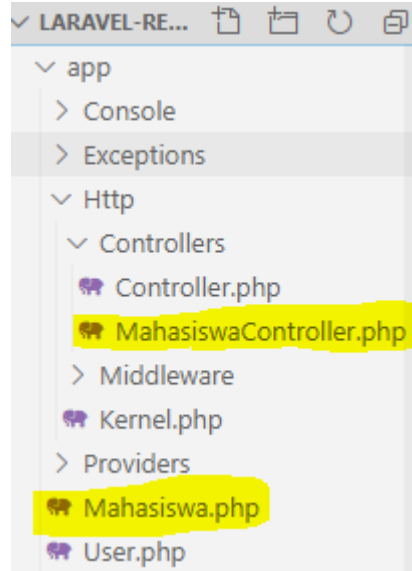
Buat **model** dengan nama **Mahasiswa**, buat juga **controllernya**. Untuk membuat model dan **controllernya** sekaligus tuliskan perintah berikut pada *command prompt* (terlebih dahulu keluar dari php artisan serve dengan mengetik ctrl+C pada keyboard)

php artisan make:model Mahasiswa -c

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**.

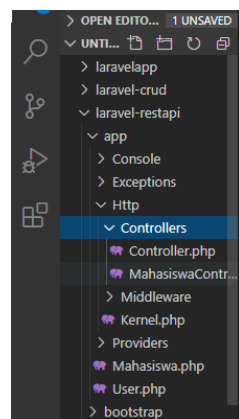


Laporan:

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

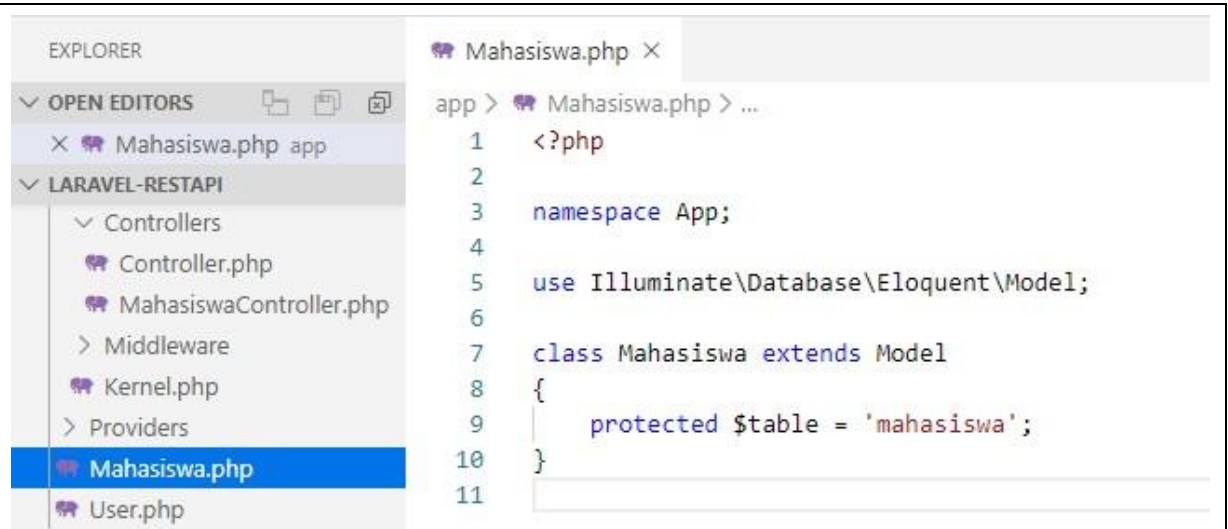
C:\xampp\htdocs\laravel-restapi>php artisan make:controller MahasiswaController
Controller created successfully.

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



5

Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.

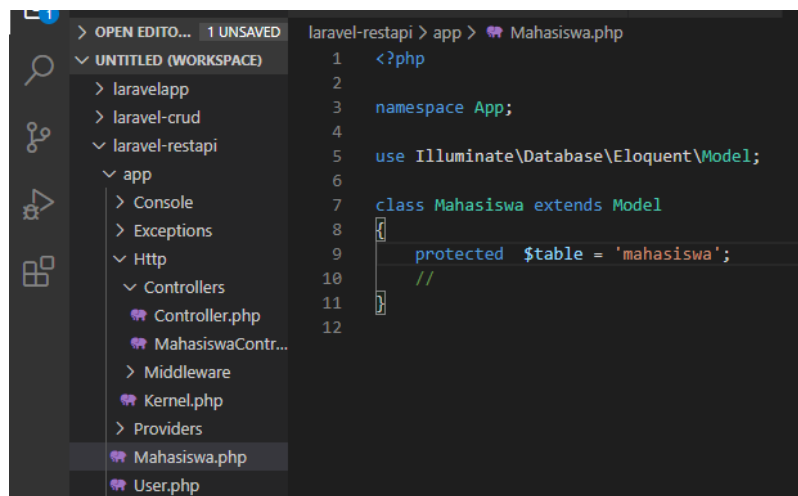


```
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Mahasiswa extends Model
8  {
9      protected $table = 'mahasiswa';
10 }
11
```

Keterangan:

- Model ini akan mengelola tabel “mahasiswa” yang terdapat pada database latihan_laravel

Laporan:

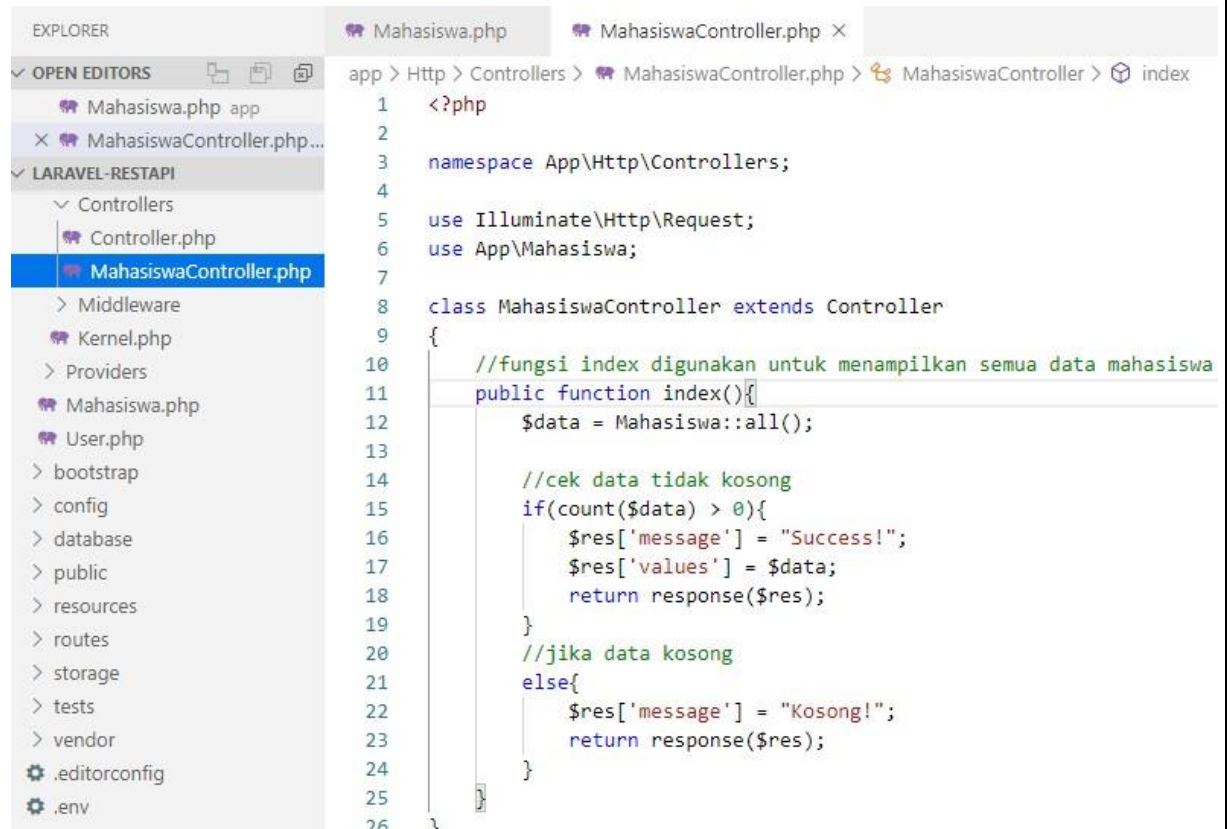


```
1  <?php
2
3  namespace App;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Mahasiswa extends Model
8  {
9      protected $table = 'mahasiswa';
10      //
11  }
12
```

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.

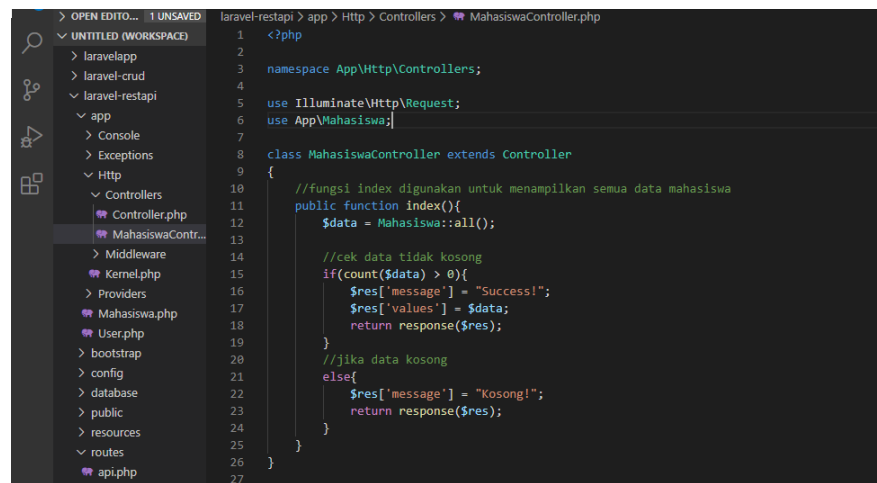


```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Mahasiswa;
7
8  class MahasiswaController extends Controller
9  {
10     //fungsi index digunakan untuk menampilkan semua data mahasiswa
11     public function index(){
12         $data = Mahasiswa::all();
13
14         //cek data tidak kosong
15         if(count($data) > 0){
16             $res['message'] = "Success!";
17             $res['values'] = $data;
18             return response($res);
19         }
20         //jika data kosong
21         else{
22             $res['message'] = "Kosong!";
23             return response($res);
24         }
25     }
26 }
```

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

Laporan:



```
<?php
namespace App\Http\Controllers;

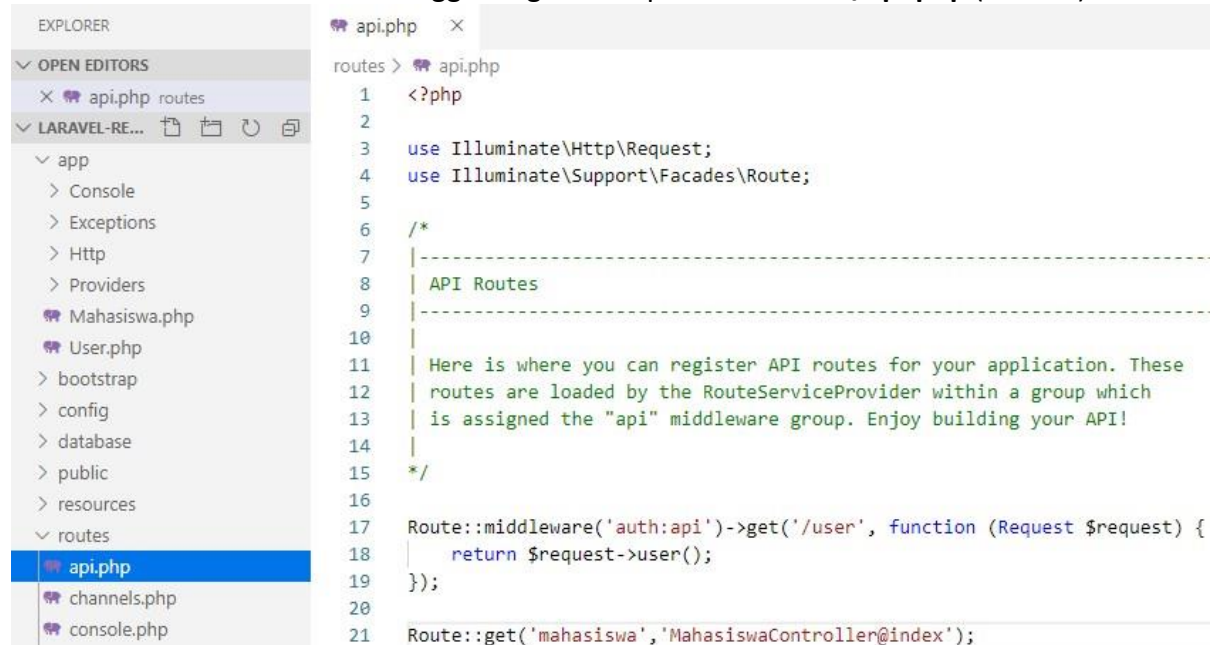
use Illuminate\Http\Request;
use App\Mahasiswa;

class MahasiswaController extends Controller
{
    //fungsi index digunakan untuk menampilkan 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);
        }
    }
}
```

7

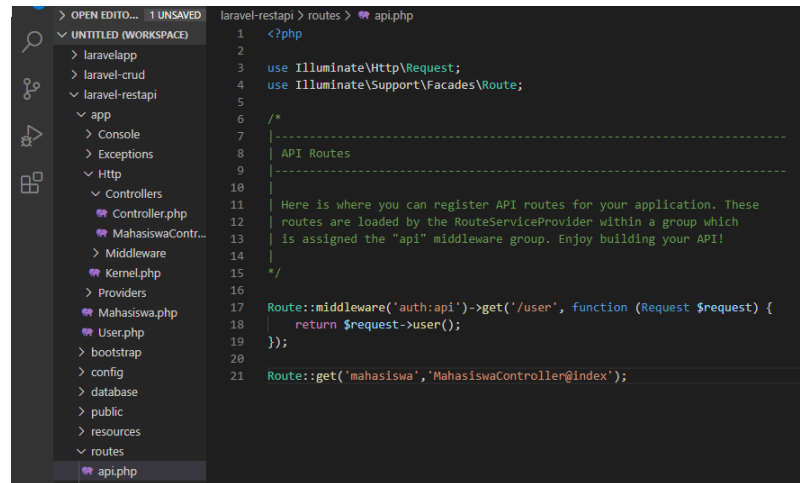
Tambahkan route untuk memanggil fungsi index pada file **routes/api.php** (Line 21).



```
api.php
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');
```

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

Laporan:



```
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');
```

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.

GET http://localhost:8000/api/maha... + ... No Environment

Untitled Request Comments 0

GET http://localhost:8000/api/mahasiswa Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (9) Test Results Status: 200 OK Time: 7.64s Size: 1.11 KB Save Response

Pretty Raw Preview Visualize JSON

```
5 {
6   "id": 1,
7   "nama": "nanda eka",
8   "nim": "1001001001",
9   "email": "nandan@gmail.com",
10  "jurusan": "teknik informatika",
11  "created_at": null,
12  "updated_at": null
13 },
14 {
15   "id": 2,
16   "nama": "wahyu",
17   "nim": "1001001002",
18   "email": "wwahyu@gmail.com",
19   "jurusan": "teknik elektro",
```

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

Laporan:

My Workspace Invite No Environment

Launchpad GET http://localhost:8000/api/maha... + ...

Untitled Request Comments 0

GET http://localhost:8000/api/mahasiswa Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (9) Test Results Status: 200 OK Time: 899ms Size: 615 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Success!",
3   "values": [
4     {
5       "id": 1,
6       "nama": "nanda",
7       "nim": "1001001001",
8       "email": "nandan@gmail.com",
9       "jurusan": "teknik informatika"
10    },
11    {
12      "id": 2,
13      "nama": "wahyu",
14      "nim": "1001001002",
15      "email": "wwahyu@gmail.com"
```

9

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

```

26
27 //fungsi untuk menampilkan data dari sebuah ID
28 public function getId($id)
29 {
30     $data = Mahasiswa::where('id',$id)->get();
31
32     //cek jika data ditemukan
33     if(count($data) > 0){
34         $res['message'] = "Success!";
35         $res['values'] = $data;
36         return response($res);
37     }
38     //jika data tidak ditemukan
39     else{
40         $res['message'] = "Gagal!";
41         return response($res);
42     }
43 }

```

Keterangan:

- Fungsi getId menerima parameter \$id yang menunjukkan ID mahasiswa yang dipilih
- Line 30 merupakan pemanggilan model untuk membaca data berdasarkan ID

Laporan:

```

15
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 //fungsi untuk menampilkan data dari sebuah ID
28 public function getId($id)
29 {
30     $data = Mahasiswa::where('id',$id)->get();
31
32     //cek jika data ditemukan
33     if(count($data) > 0){
34         $res['message'] = "Success!";
35         $res['values'] = $data;
36         return response($res);
37     }
38     //jika data tidak ditemukan
39     else{
40         $res['message'] = "Gagal!";
41         return response($res);
42     }
43 }

```

10

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

```

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

```

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

Laporan:

```
laravel-restapi > 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
23 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId');
```

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=2, maka url diisi :
http://localhost:8000/api/mahasiswa/2

GET http://localhost:8000/api/mahasiswa/2

Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 890ms Size: 467 B Save

Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "Success!",
3   "values": [
4     {
5       "id": 2,
6       "nama": "wahyu",
7       "nim": "1001001002",
8       "email": "wwahyu@gmail.com",
9       "jurusan": "teknik elektro",
10      "created_at": null,
11      "updated_at": "2020-04-12T13:57:56.000000Z"
12    }
13  ]
14 }

```

Ketika mencoba menampilkan ID=10 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut.

GET http://localhost:8000/api/mahasiswa/10

Send Save

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies Code

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 799ms Size: 296 B Save Response

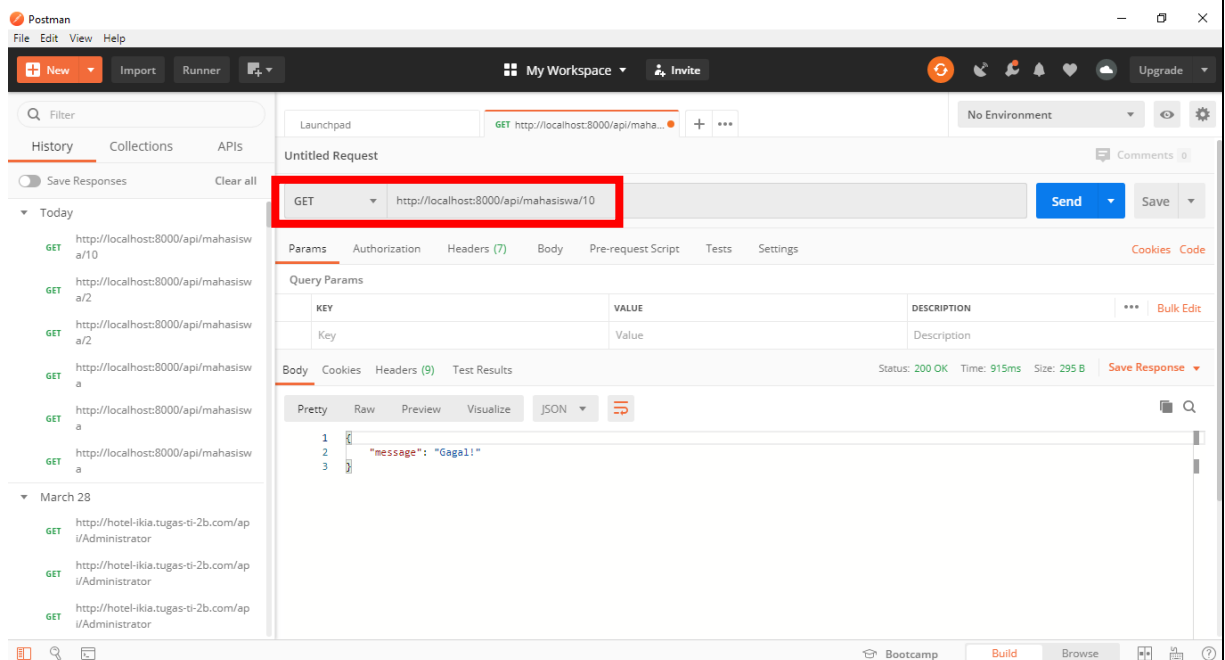
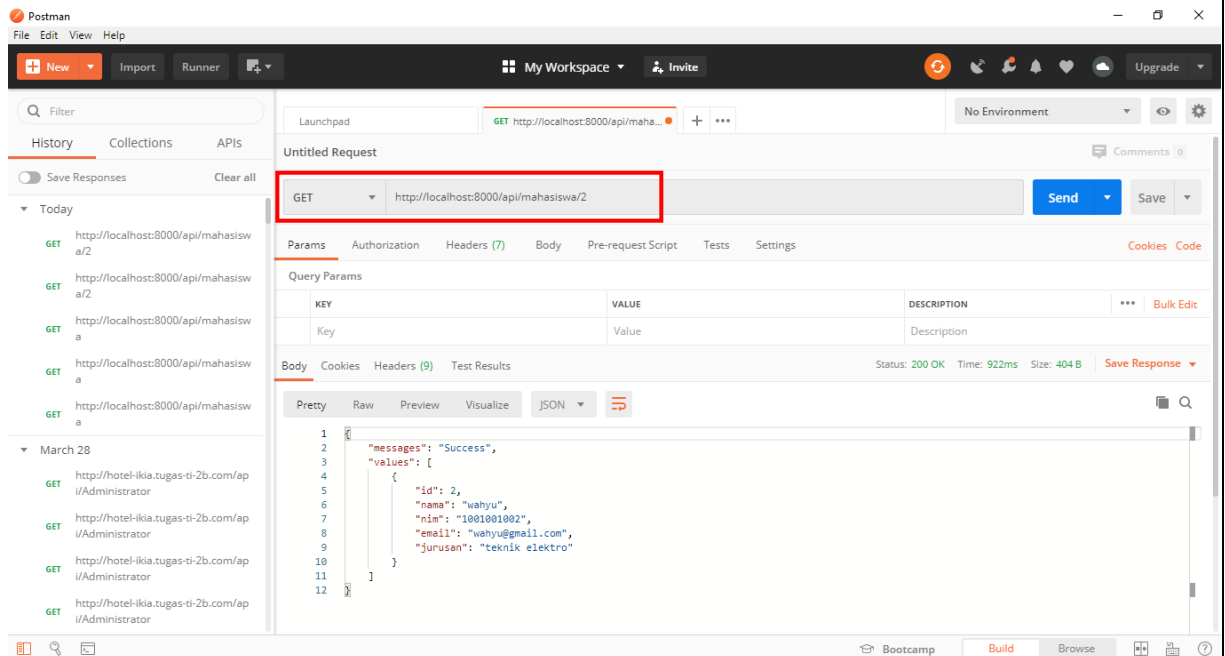
Pretty Raw Preview Visualize JSON

```

1 {
2   "message": "Gagal!"
3 }

```

Laporan:



12

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

```

44
45 //fungsi tambah data
46 public function create(Request $request)
47 {
48     $mhs = new Mahasiswa();
49     $mhs->nama = $request->nama;
50     $mhs->nim = $request->nim;
51     $mhs->email = $request->email;
52     $mhs->jurusan = $request->jurusan;
53
54     //jika data berhasil tersimpan
55     if($mhs->save()){
56         $res['message'] = "Data berhasil ditambah!";
57         $res['value'] = "$mhs";
58         return response($res);
59     }
60 }

```

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.

Laporan:

```

43 }
44
45 //fungsi tambah data
46 public function create(Request $request)
47 {
48     $mhs = new Mahasiswa();
49     $mhs->nama = $request->nama;
50     $mhs->nim = $request->nim;
51     $mhs->email = $request->email;
52     $mhs->jurusan = $request->jurusan;
53
54     //jika data berhasil tersimpan
55     if($mhs->save()){
56         $res['message'] = "Data berhasil ditambah!";
57         $res['value'] = "$mhs";
58         return response($res);
59     }
60 }
61
62 }

```

13

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

```

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

```

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

Laporan:

```

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

```

14

Kita coba untuk menambahkan data melalui Postman.

The screenshot shows the Postman interface with a POST request to `http://localhost:8000/api/mahasiswa`. The request body is in `x-www-form-urlencoded` format with the following data:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	eka	
<input checked="" type="checkbox"/> nim	1001001011	
<input checked="" type="checkbox"/> email	eka@gmail.com	
<input checked="" type="checkbox"/> jurusan	teknik informatika	
Key	Value	Description

The response status is 200 OK. The response body in JSON format is:

```
1 {
2   "message": "Data berhasil ditambah!",
3   "value": "{\n  \"nama\": \"eka\", \"nim\": \"1001001011\", \"email\": \"eka@gmail.com\", \"jurusan\": \"teknik informatika\",
4     \"updated_at\": \"2020-05-03T06:20:26.000000Z\", \"created_at\": \"2020-05-03T06:20:26.000000Z\", \"id\": 11}"
```

Laporan:

This screenshot shows the full Postman application window. On the left, the 'History' tab shows a list of recent POST requests to `http://localhost:8000/api/mahasiswa`. The main panel displays the same POST request and 200 OK response as the first screenshot, with the response body in JSON format:

```
1 {
2   "message": "Data berhasil ditambah!",
3   "value": "{\n  \"nama\": \"eka\", \"nim\": \"1001001011\", \"email\": \"eka@gmail.com\", \"jurusan\": \"teknik informatika\",
4     \"updated_at\": \"2020-05-03T23:41:23.000000Z\", \"created_at\": \"2020-05-03T23:41:23.000000Z\", \"id\": 7}"
```


- 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.

```

40  {
41    "id": 8,
42    "nama": "Farah",
43    "nim": "1001001008",
44    "email": "farah@gmail.com",
45    "jurusan": "teknik sipil",
46    "created_at": null,
47    "updated_at": null
48  },
49  {
50    "id": 9,
51    "nama": "Farhan",
52    "nim": "1001001010",
53    "email": "farhan@gmail.com",
54    "jurusan": "teknik sipil",
55    "created_at": null,
56    "updated_at": null
57  },
58  {
59    "id": 10,
60    "nama": "Eka",
61    "nim": "1001001011",
62    "email": "eka@gmail.com",
63    "jurusan": "teknik informatika",
64    "created_at": "2020-05-02T06:19:58.000000Z",
65    "updated_at": "2020-05-02T06:19:58.000000Z"
66  }
  
```

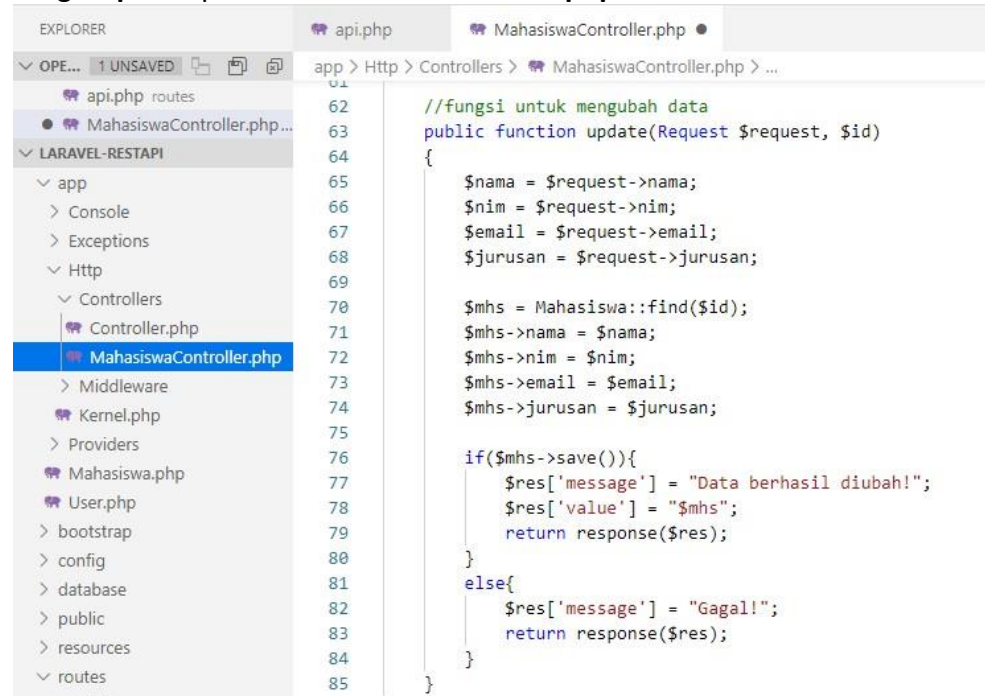
Laporan:

```

24    "nama": "rahmat",
25    "nim": "1001001006",
26    "email": "rahmat@gmail.com",
27    "jurusan": "teknik mesin",
28    "created_at": "2020-04-13T06:00:55.000000Z",
29    "updated_at": "2020-04-13T06:00:55.000000Z"
30  },
31  {
32    "id": 5,
33    "nama": "Ade Ismail",
34    "nim": "0804030028",
35    "email": "adeismail10@gmail.com",
36    "jurusan": "teknik informatika",
37    "created_at": null,
38    "updated_at": null
39  },
40  {
41    "id": 7,
42    "nama": "Eka",
43    "nim": "1001001011",
44    "email": "eka@gmail.com",
45    "jurusan": "teknik informatika",
46    "created_at": "2020-05-03T23:41:23.000000Z",
47    "updated_at": "2020-05-03T23:41:23.000000Z"
48  }
  
```

15

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



The screenshot shows an IDE with the Explorer panel on the left and the code editor on the right. The Explorer panel shows the project structure with the file **MahasiswaController.php** selected under the **Controllers** directory. The code editor displays the **update** function in **MahasiswaController.php**, which is designed to update a student's information in the database.

```
62 //fungsi untuk mengubah data
63 public function update(Request $request, $id)
64 {
65     $nama = $request->nama;
66     $nim = $request->nim;
67     $email = $request->email;
68     $jurusan = $request->jurusan;
69
70     $mhs = Mahasiswa::find($id);
71     $mhs->nama = $nama;
72     $mhs->nim = $nim;
73     $mhs->email = $email;
74     $mhs->jurusan = $jurusan;
75
76     if($mhs->save()){
77         $res['message'] = "Data berhasil diubah!";
78         $res['value'] = "$mhs";
79         return response($res);
80     }
81     else{
82         $res['message'] = "Gagal!";
83         return response($res);
84     }
85 }
```

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.

Laporan:

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

16

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

26

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

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

Laporan:

```
laravel-restapi > routes > api.php
26
27 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');
28
```

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 an 'Untitled Request'. The method is 'PUT' and 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. A table lists the data to be updated:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	wahyu afifah	
<input checked="" type="checkbox"/> nim	1001001002	
<input checked="" type="checkbox"/> email	wahyua@gmail.com	
<input checked="" type="checkbox"/> jurusan	teknik elektro	
Key	Value	Description

Below the table, the 'Body' tab shows the JSON response:

```

1 {
2   "message": "Data berhasil diubah!",
3   "value": "{\n\"id\":2,\n\"nama\": \"wahyu afifah\", \n\"nim\": \"1001001002\", \n\"email\": \"wahyua@gmail.com\", \n\"jurusan\": \"teknik elektro\", \n\"created_at\": null, \n\"updated_at\": \"2020-05-02T23:23:20.000000Z\"}"
4 }

```

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*.

GET http://localhost:8000/api/maha... + ... No Environment

Untitled Request Comments 0

GET http://localhost:8000/api/mahasiswa/2 Send Save

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 804 ms Size: 474 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Success!",
3   "values": [
4     {
5       "id": 2,
6       "nama": "wahyu afifah",
7       "nim": "1001001002",
8       "email": "wahyua@gmail.com",
9       "jurusan": "teknik elektro",
10      "created_at": null,
11      "updated_at": "2020-05-02T23:23:20.000000Z"
12    }
13  ]
14 }
```

Laporan:

Postman File Edit View Help

New Import Runner My Workspace Invite Upgrade

Filter History Collections APIs Save Responses Clear all

Today

- PUT http://localhost:8000/api/mahasiswa/update/2
- PUT http://localhost:8000/api/mahasiswa/update/2
- PUT http://localhost:8000/api/mahasiswa/update/2
- GET http://localhost:8000/api/mahasiswa
- POST http://localhost:8000/api/mahasiswa
- POST http://localhost:8000/api/mahasiswa
- POST http://localhost:8000/api/mahasiswa
- POST http://localhost:8000/api/mahasiswa
- POST http://localhost:8000/api/mahasiswa

Launched PUT http://localhost:8000/api/maha... + ... No Environment

Untitled Request Comments 0

PUT http://localhost:8000/api/mahasiswa/update/2 Send Save

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

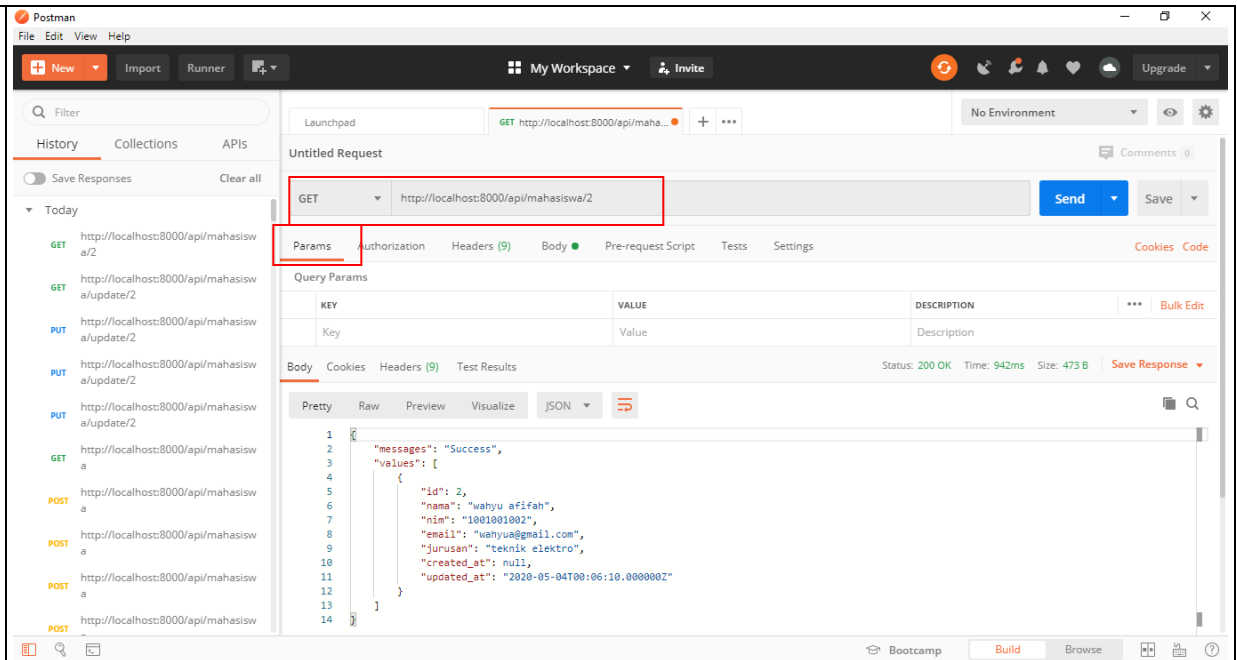
none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	wahyu afifah	
<input checked="" type="checkbox"/> nim	1001001002	
<input checked="" type="checkbox"/> email	wahyua@gmail.com	
<input checked="" type="checkbox"/> jurusan	teknik elektro	
Key	Value	Description

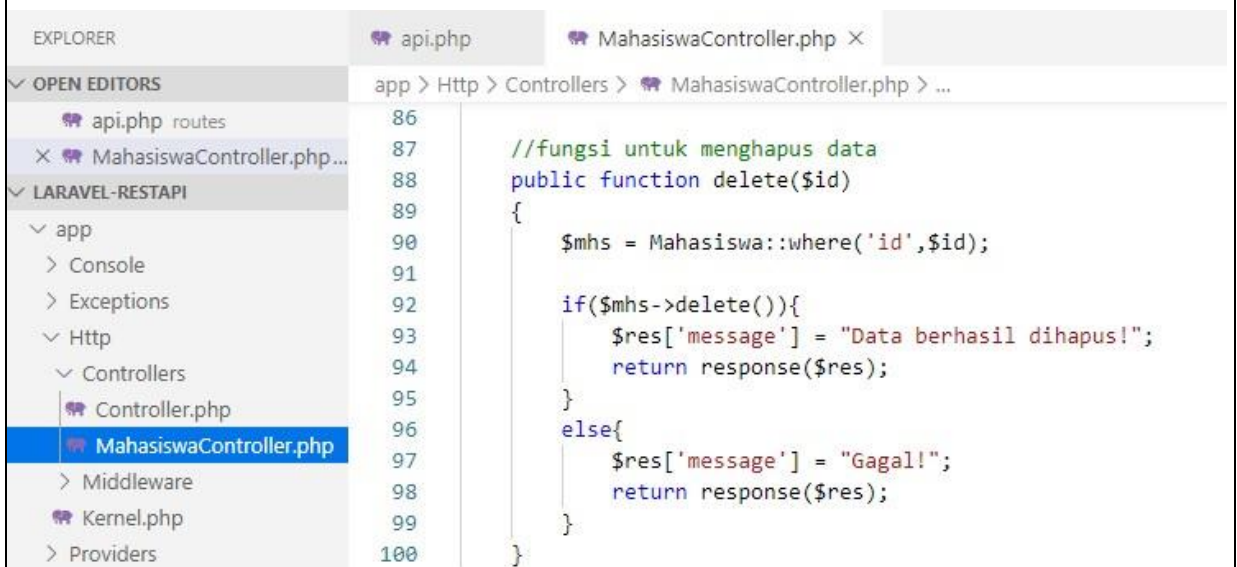
Body Cookies Headers (9) Test Results Status: 200 OK Time: 988ms Size: 509 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Data berhasil diubah!",
3   "value": "{\n  \"id\":2,\n  \"nama\": \"wahyu afifah\",\n  \"nim\": \"1001001002\",\n  \"email\": \"wahyua@gmail.com\",\n  \"jurusan\": \"teknik elektro\",\n  \"created_at\": null,\n  \"updated_at\": \"2020-05-04T00:06:10.000000Z\"}"
4 }
```



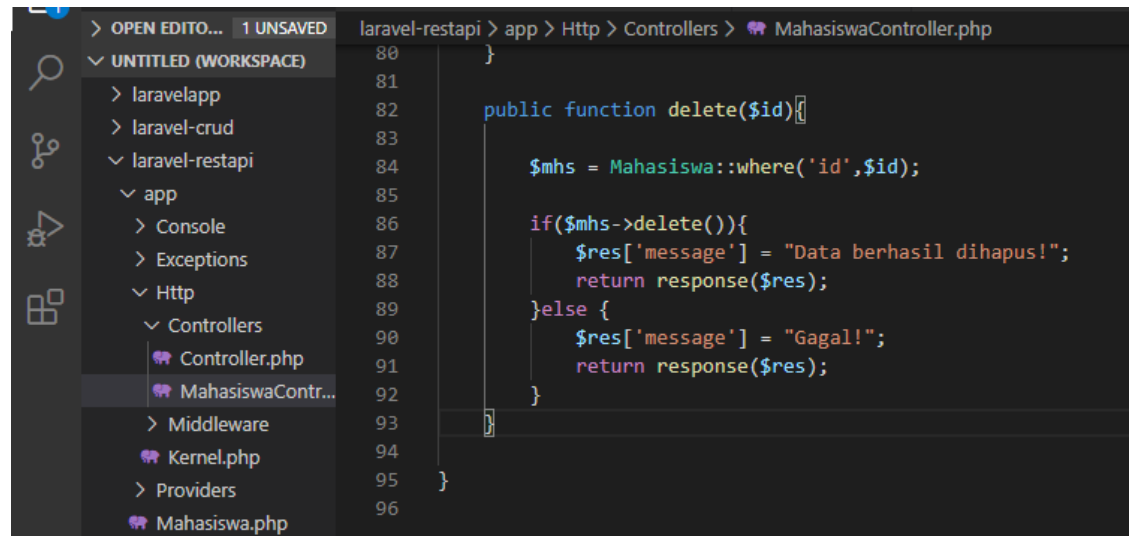
18 Terakhir kita akan membuat fungsi untuk menghapus data dengan nama **delete** di **MahasiswaController.php**.



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.

Laporan:



```
laravel-restapi > app > Http > Controllers > MahasiswaController.php
80 }
81
82 public function delete($id){
83
84     $mhs = Mahasiswa::where('id',$id);
85
86     if($mhs->delete()){
87         $res['message'] = "Data berhasil dihapus!";
88         return response($res);
89     }else {
90         $res['message'] = "Gagal!";
91         return response($res);
92     }
93
94 }
95
96 }
```

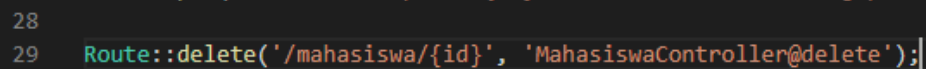
19

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'.

Laporan:



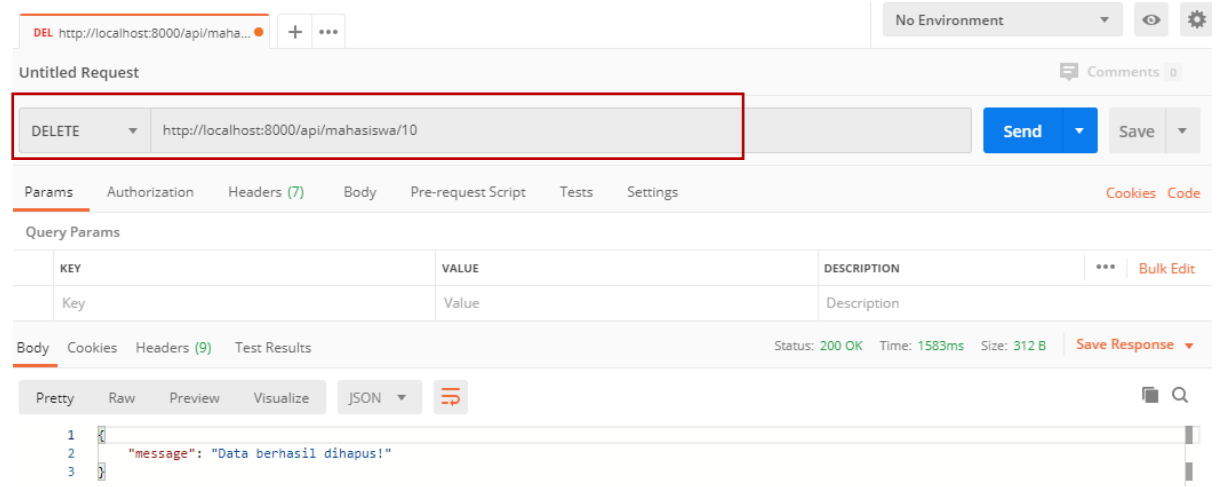
```
28
29 Route::delete('/mahasiswa/{id}', 'MahasiswaController@delete');
```

20

Buka Postman untuk mencoba menghapus data dari database. Gunakan perintah **DELETE** untuk mengubah data.

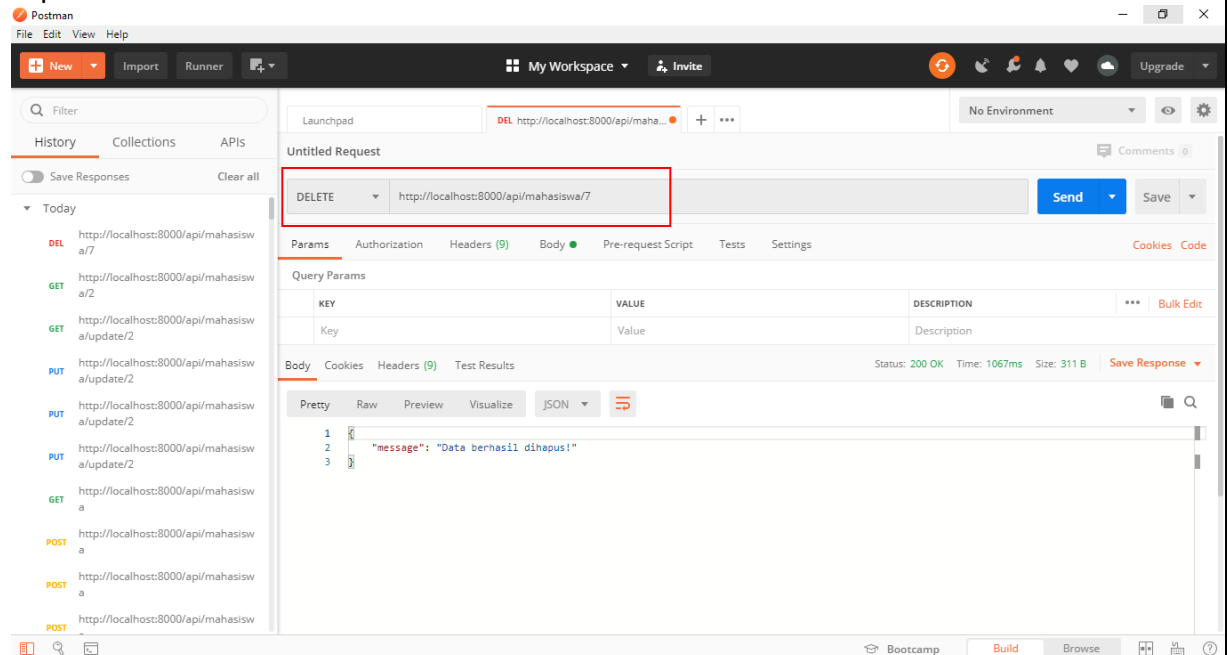
Berikut adalah contoh untuk menghapus data dengan ID=10, maka url diisi :

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



Muncul pesan berhasil ketika data terhapus dari database.

Laporan:



-- Selamat Mengerjakan --