

LAPORAN PRAKTIKUM PEMROGRAMAN WEBSITE LANJUT

Restful API laravel

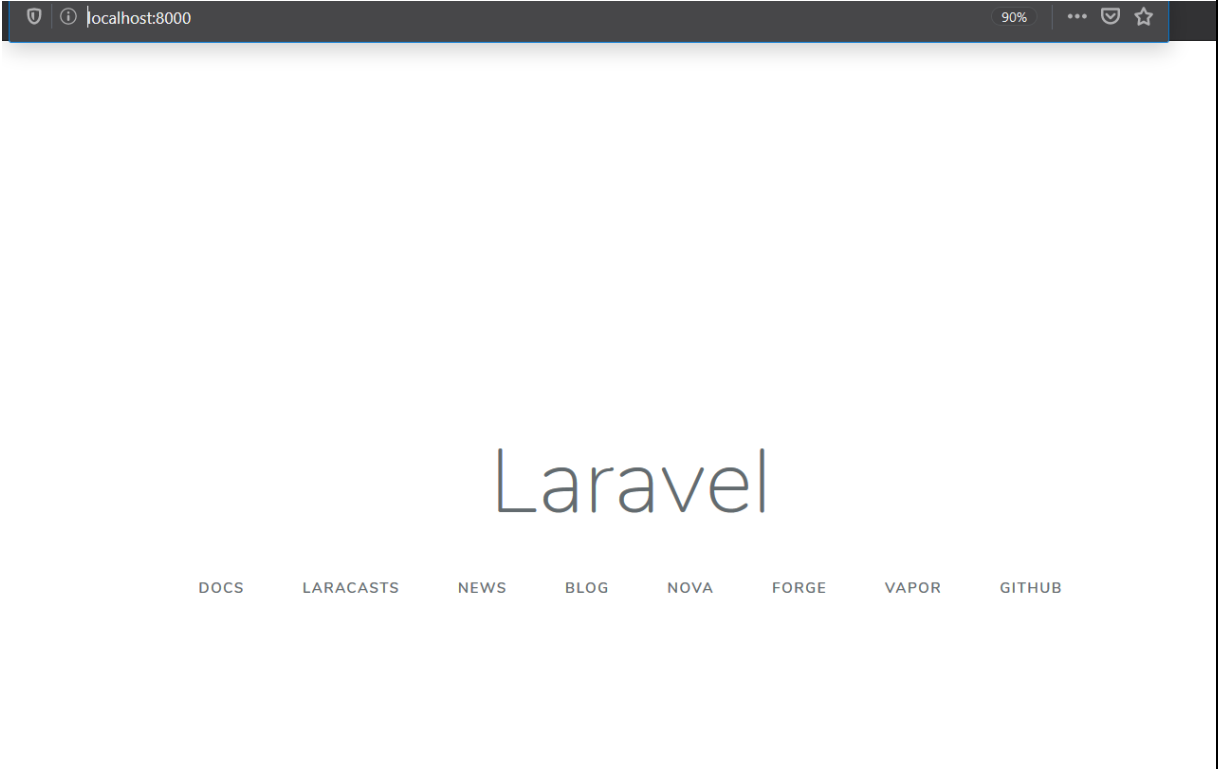


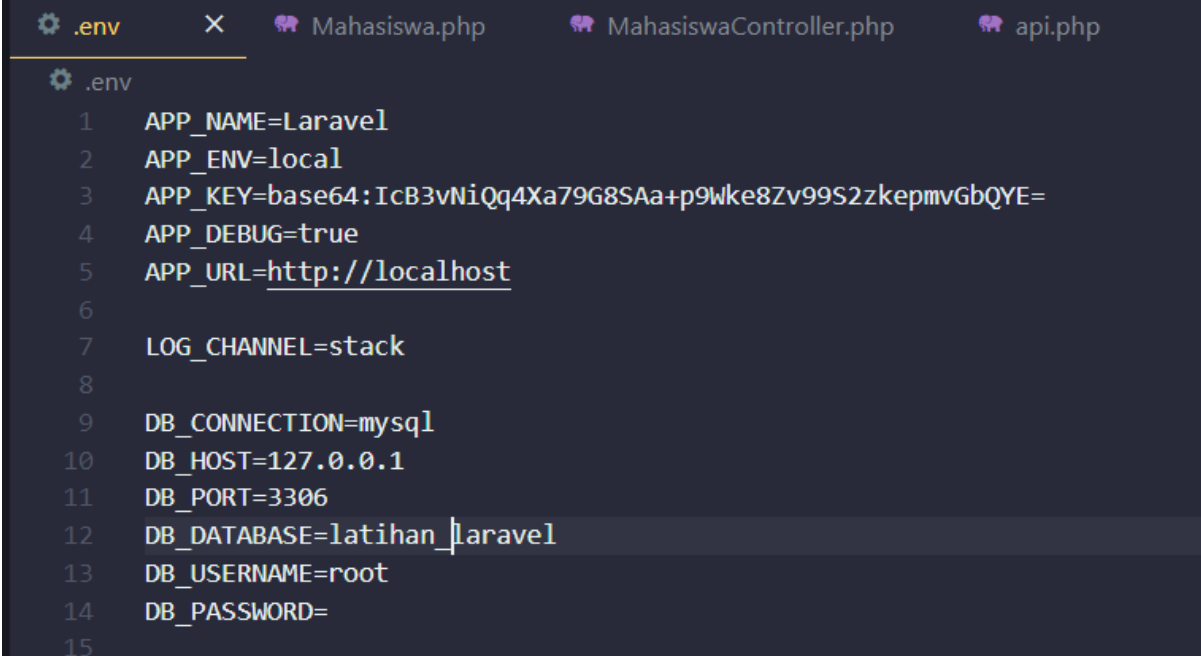
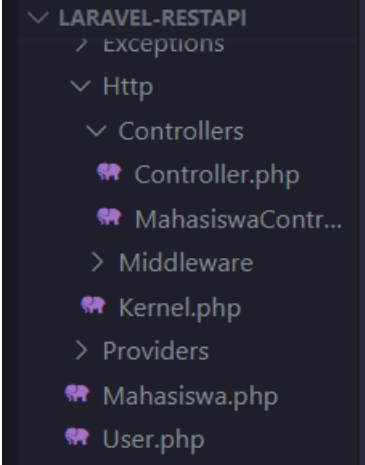
Oleh :

**A.SAFA DHIATA / NIM : 1941723012
KELAS TI 2B**

**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2020**

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

	 <pre> .env 1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:IcB3vNiQq4Xa79G8SAa+p9Wke8Zv99S2zkepmvGbQYE= 4 APP_DEBUG=true 5 APP_URL=http://localhost 6 7 LOG_CHANNEL=stack 8 9 DB_CONNECTION=mysql 10 DB_HOST=127.0.0.1 11 DB_PORT=3306 12 DB_DATABASE=latihan_laravel 13 DB_USERNAME=root 14 DB_PASSWORD= 15 </pre>
4	<p>Buat model dengan nama Mahasiswa, buat juga controllernya. Untuk membuat model dan controllernya sekaligus tuliskan perintah berikut pada <i>command prompt</i> (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 <i>controller</i> <p>Sehingga pada project laravel-restapi akan bertambah dua file yaitu model Mahasiswa.php serta controller MahasiswaController.php.</p> 
5	Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.

```

.env      Mahasiswa.php X      MahasiswaController.php
app > Mahasiswa.php > PHP Intelephense > Mahasiswa
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

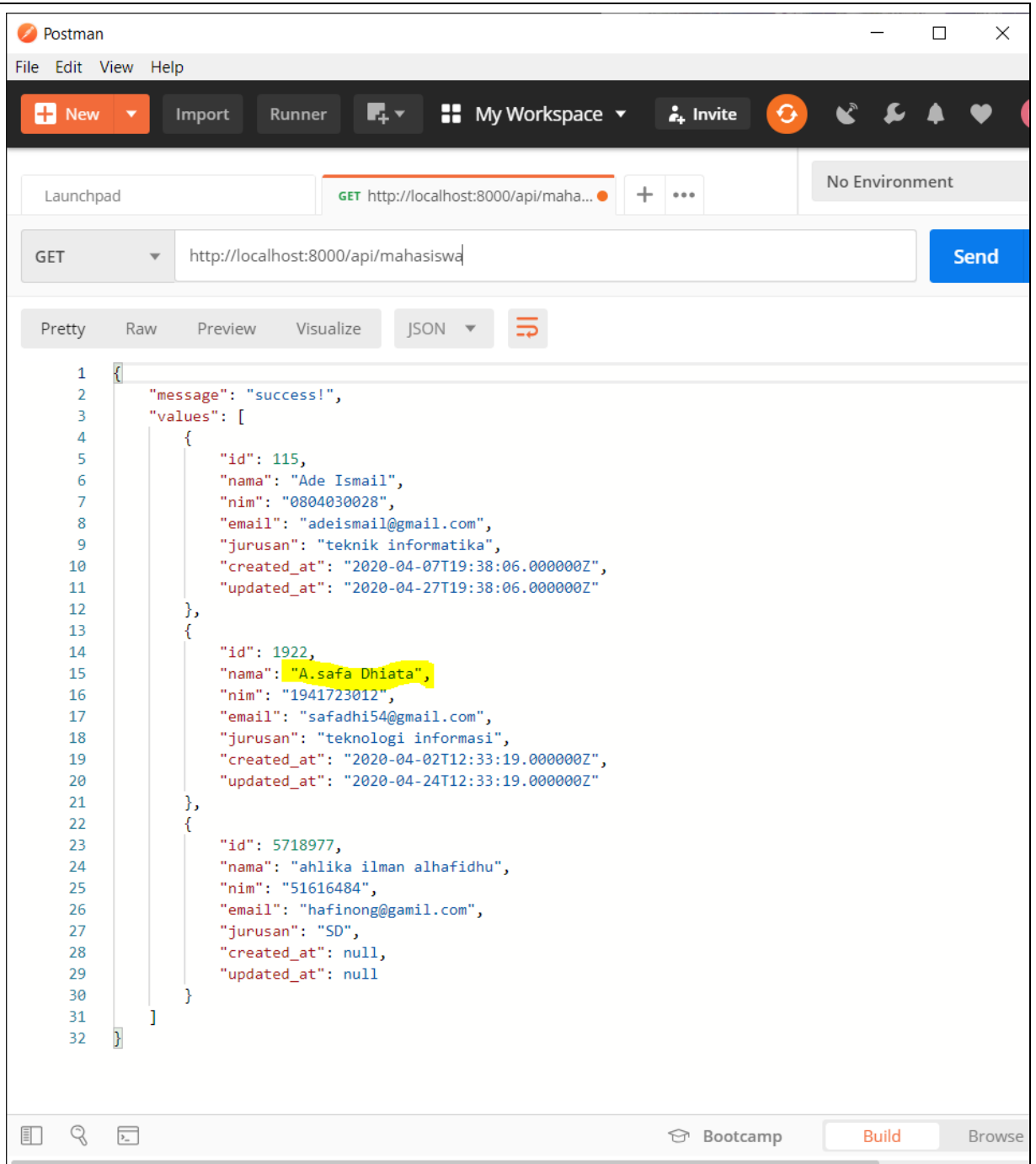
- 6 Kemudian kita akan memodifikasi isi dari **MahasiswaController.php** untuk dapat mengolah data pada tabel ‘mahasiswa’. Pada *controller* ini, kita akan melakukan operasi untuk menampilkan, menambah, mengubah, dan menghapus data. Pertama, kita akan mengubah fungsi index agar saat fungsi index dipanggil, maka aplikasi akan menampilkan seluruh data dari tabel mahasiswa.

```

.env      Mahasiswa.php      MahasiswaController.php X      api.php
app > Http > Controllers > MahasiswaController.php > PHP Intelephense > MahasiswaController
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     public function index(){
11         $data = Mahasiswa::all();
12
13         if (count($data) > 0) {
14             $res['message'] = "Success!";
15             $res['values'] = $data;
16             return response($res);
17         } else {
18             $res['message'] = "Kosong!";
19             return response($res);
20         }
21     }
22 }

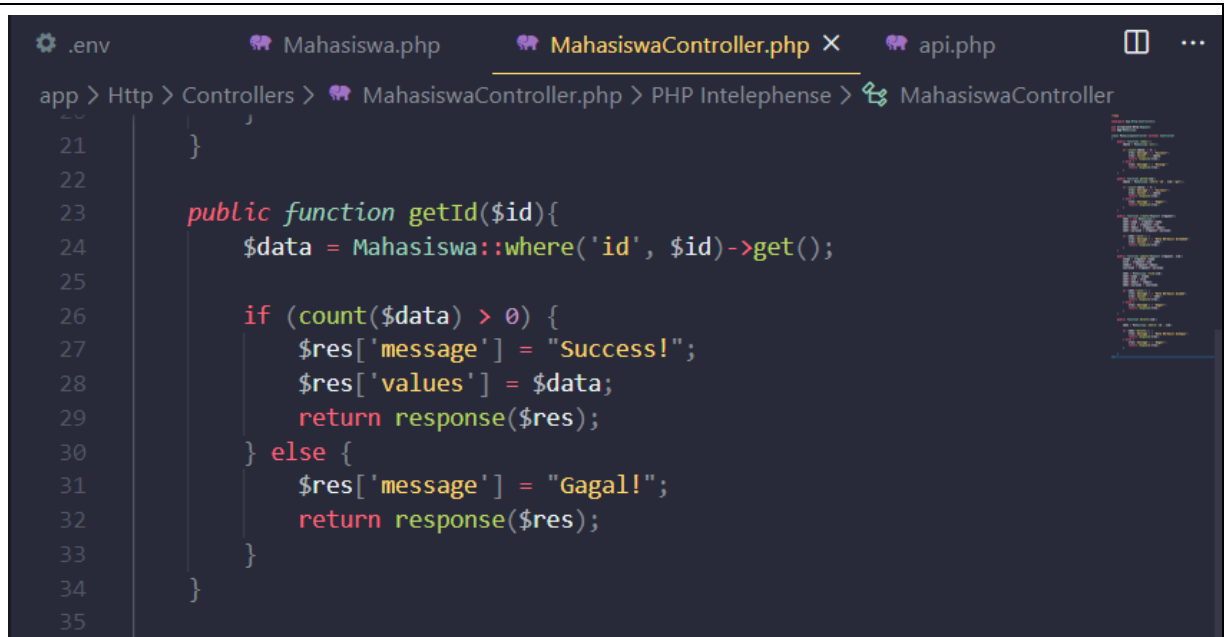
```

	<p>Keterangan:</p> <ul style="list-style-type: none"> • 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	<p>Tambahkan route untuk memanggil fungsi index pada file routes/api.php (Line 21).</p>  <pre> 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 </pre> <p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'.</p>
8	<p>Ketikkan perintah php artisan serve pada <i>command prompt</i>. 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.</p>



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

9



```
.env  Mahasiswa.php  MahasiswaController.php X  api.php
app > Http > Controllers > MahasiswaController.php > PHP Intelephense > MahasiswaController

21     }
22
23     public function getId($id){
24         $data = Mahasiswa::where('id', $id)->get();
25
26         if (count($data) > 0) {
27             $res['message'] = "Success!";
28             $res['values'] = $data;
29             return response($res);
30         } else {
31             $res['message'] = "Gagal!";
32             return response($res);
33         }
34     }
35 }
```

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

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**

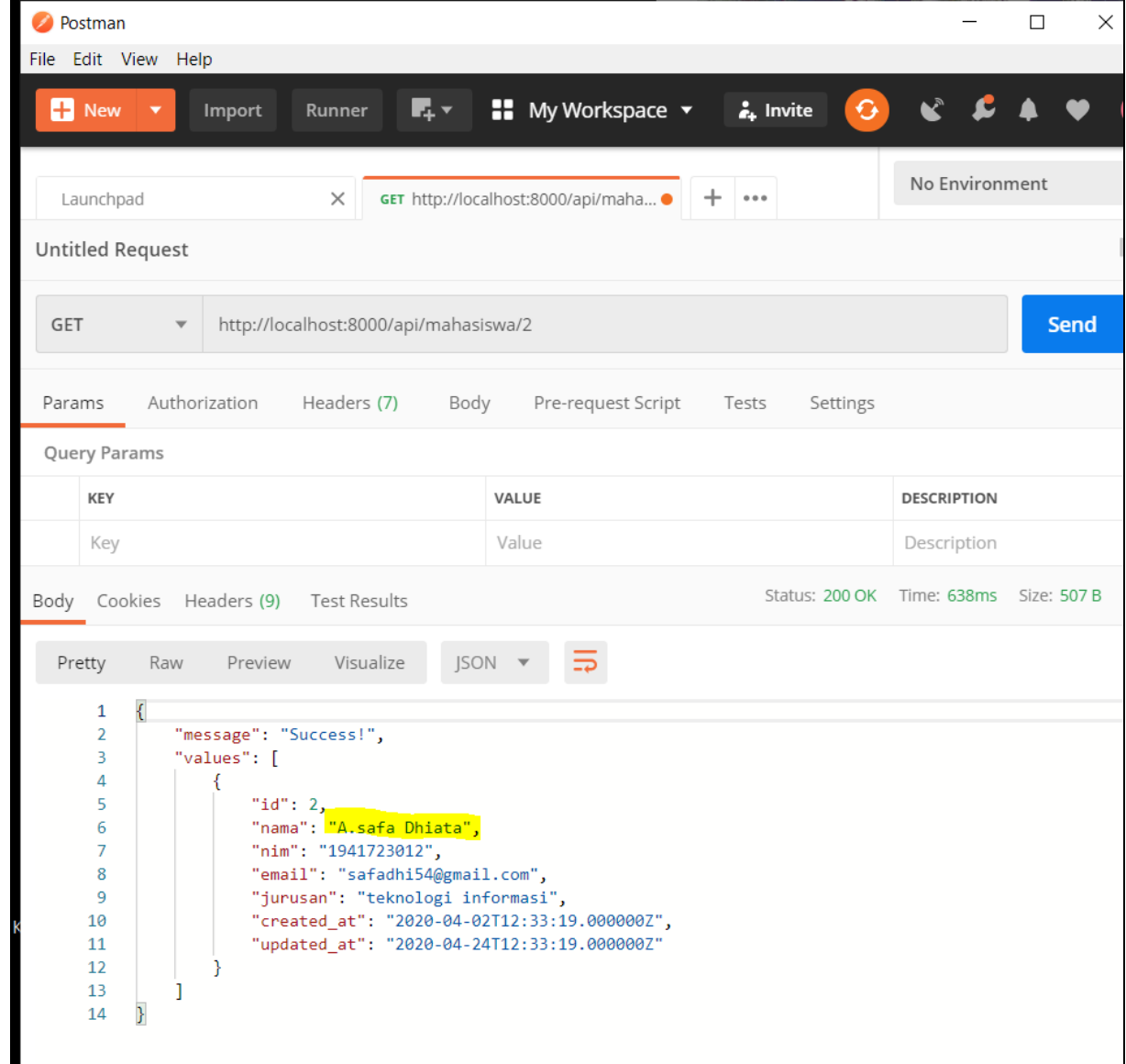
```
Route::get('mahasiswa/{id}', 'MahasiswaController@getId');
```

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.

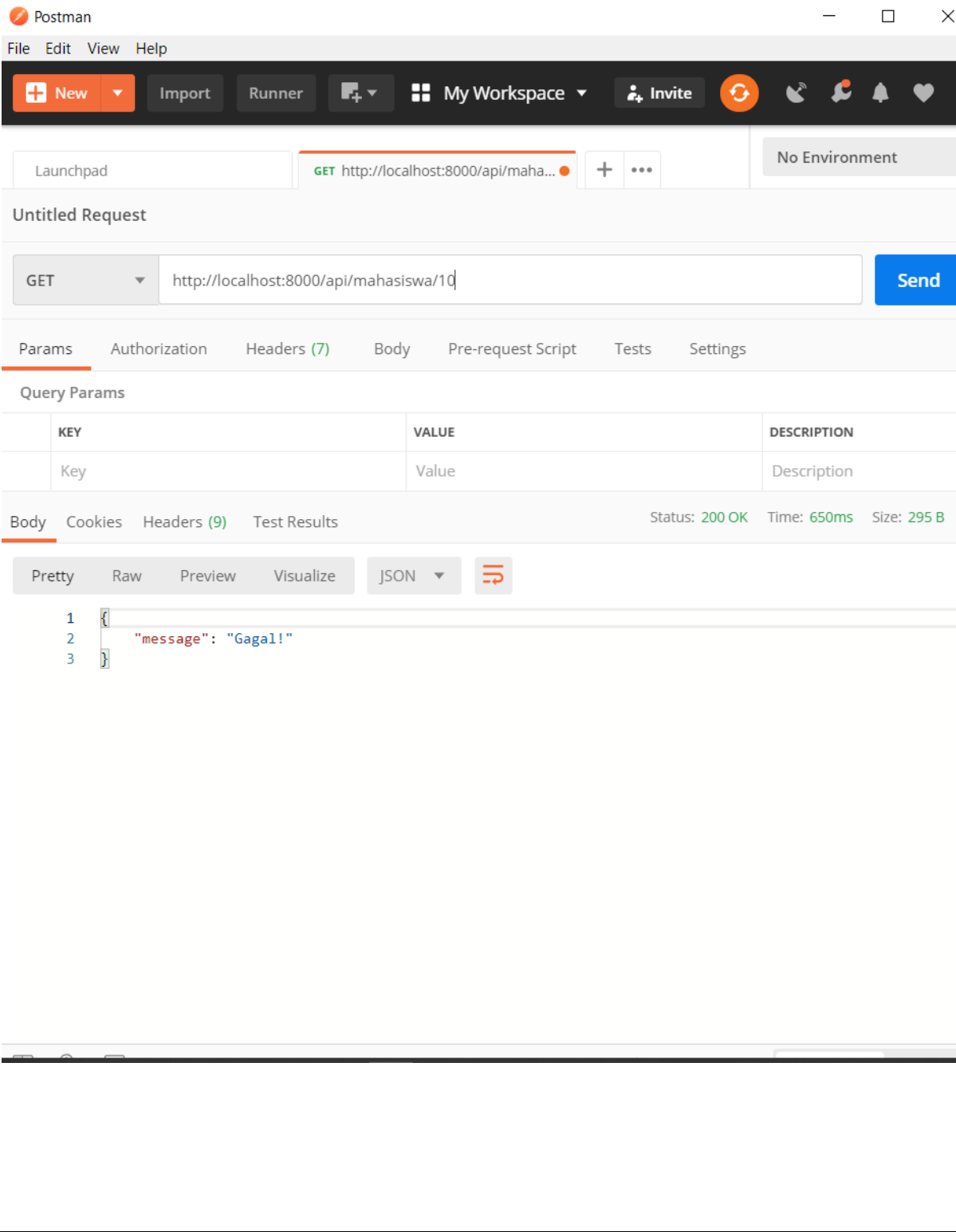
Di bawah ini adalah contoh untuk menampilkan data dengan ID=2, maka url diisi :
<http://localhost:8000/api/mahasiswa/2>



The screenshot shows the Postman interface with a GET request to `http://localhost:8000/api/mahasiswa/2` sent. The response is a JSON object with a success message and student details for ID 2. The response is displayed in the 'Body' tab, formatted as JSON.

```
1 {
2   "message": "Success!",
3   "values": [
4     {
5       "id": 2,
6       "nama": "A.safa Dhiata",
7       "nim": "1941723012",
8       "email": "safadhi54@gmail.com",
9       "jurusan": "teknologi informasi",
10      "created_at": "2020-04-02T12:33:19.000000Z",
11      "updated_at": "2020-04-24T12:33:19.000000Z"
12    }
13  ]
14 }
```

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

	 <p>The screenshot shows the Postman interface. At the top, there's a menu bar with File, Edit, View, and Help. Below it is a toolbar with buttons for New, Import, Runner, My Workspace, and Invite. The main area shows an 'Untitled Request' with a GET method and the URL 'http://localhost:8000/api/mahasiswa/10'. The 'Send' button is visible. Below the request, there are tabs for Params, Authorization, Headers (7), Body, Pre-request Script, Tests, and Settings. The 'Body' tab is selected, showing a JSON response: { 'message': 'Gagal!' }. The status bar at the bottom indicates 'Status: 200 OK', 'Time: 650ms', and 'Size: 295 B'.</p>
12	<p>Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php.</p>

```

public function create(Request $request){
    $mhs = new Mahasiswa();
    $mhs->nama = $request->nama;
    $mhs->nim = $request->nim;
    $mhs->email = $request->email;
    $mhs->jurusan = $request->jurusan;

    if ($mhs->save()) {
        $res['message'] = "Data Berhasil ditambah";
        $res['values'] = "$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 ditambahkan.

13

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

```
Route::post('mahasiswa', 'MahasiswaController@create');
```

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

14

Kita coba untuk menambahkan data melalui Postman.

Postman

File Edit View Help

New

Import

Runner

My Workspace

Invite

Upgrade

Launchpad

POST http://localhost:8000/api/mah...

+ ...

No Environment

Untitled Request

Comments 0

POST

http://localhost:8000/api/mahasiswa

Send

Save

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

Cod

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<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		

Body

Cookies

Headers (9)

Test Results

Status: 200 OK Time: 672ms Size: 536 B Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

{

"message": "Data Berhasil ditambah",

"values": "{\\"nama\\":\\"eka\\",\\"nim\\":\\"1001001011\\",\\"email\\":\\"eka@gmail.com\\",\\"jurusan\\":\\"teknik informatika\\",\\"updated_at\\":\\"2020-05-04T11:16:03.000000Z\\",\\"created_at\\":\\"2020-05-04T11:16:03.000000Z\\",\\"id\\":5718980}"

}

- 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.

```

7      "nim": "0804030020",
8      "email": "adeismail@gmail.com",
9      "jurusan": "teknik informatika",
10     "created_at": "2020-04-07T19:38:06.000000Z",
11     "updated_at": "2020-04-27T19:38:06.000000Z"
12   },
13   {
14     "id": 2,
15     "nama": "A.safa Dhiata",
16     "nim": "1941723012",
17     "email": "safadhi54@gmail.com",
18     "jurusan": "teknologi informasi",
19     "created_at": "2020-04-02T12:33:19.000000Z",
20     "updated_at": "2020-04-24T12:33:19.000000Z"
21   },
22   {
23     "id": 3,
24     "nama": "ahlika ilman alhafidhu",
25     "nim": "51616484",
26     "email": "hafinong@gmail.com",
27     "jurusan": "SD",
28     "created_at": null,
29     "updated_at": null
30   },
31   {
32     "id": 5718980,
33     "nama": "eka",
34     "nim": "1001001011",
35     "email": "eka@gmail.com",
36     "jurusan": "teknik informatika",
37     "created_at": "2020-05-04T11:16:03.000000Z",
38     "updated_at": "2020-05-04T11:16:03.000000Z"
39   }
40 ]
41 ]
  
```

15

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

```
public function update(Request $request, $id){
    $nama = $request->nama;
    $nim = $request->nim;
    $email = $request->email;
    $jurusan = $request->jurusan;

    $mhs = Mahasiswa::find($id);
    $mhs->nama = $nama;
    $mhs->nim = $nim;
    $mhs->email = $email;
    $mhs->jurusan = $jurusan;

    if ($mhs->save()) {
        $res['message'] = "Data Berhasil diubah";
        $res['values'] = "$mhs";
        return response($res);
    } else {
        $res['message'] = "Gagal!";
        return response($res);
    }
}
```

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**

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

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 :

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 application interface. At the top, there's a 'Launchpad' section with a 'PUT' request to 'http://localhost:8000/api/mahasiswa/update/2'. Below this, the 'Body' tab is selected, showing a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The table contains the following data:

KEY	VALUE	DESCRIPTION
nama	A.SAFA DHIATA	
nim	1922	
email	SAFADHIIIIATAAAAAAAAAA@gmail.com	
jurusan	teknik informatika TI 2 B	
Key	Value	Description

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

```
{
  "message": "Data Berhasil diubah",
  "values": "{\n  \"id\":2,\n  \"nama\": \"A.SAFA DHIATA\",\n  \"nim\": \"1922\",\n  \"email\": \"SAFADHIIIIATAAAAAAAAAA@gmail.com\",\n  \"jurusan\": \"teknik informatika TI 2 B\",\n  \"created_at\": \"2020-04-02T12:33:19.000000Z\",\n  \"updated_at\": \"2020-05-04T11:20:38.000000Z\"\n}"
}
```

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*.

The screenshot shows the Postman application interface. At the top, there's a menu bar with 'File', 'Edit', 'View', and 'Help'. Below it is a toolbar with buttons for 'New', 'Import', 'Runner', 'My Workspace', 'Invite', and a refresh button. The main area is divided into sections. The first section is 'Launchpad' with a dropdown menu showing 'GET http://localhost:8000/api/maha...'. The second section is 'Untitled Request' with a dropdown menu set to 'GET' and a text input field containing 'http://localhost:8000/api/mahasiswa/2'. A blue 'Send' button is to the right. Below this is a tabbed interface with 'Params', 'Authorization', 'Headers (9)', 'Body', 'Pre-request Script', 'Tests', and 'Settings'. The 'Params' tab is active, showing a table with 'KEY' and 'VALUE' columns. The 'Body' tab is also visible, showing a JSON response. The response is displayed in the 'Body' tab, showing a JSON object with a 'message' and a 'values' array. The 'values' array contains a single object with student data for ID 2. The status bar at the bottom shows 'Status: 200 OK', 'Time: 642ms', and 'Size: 521 B'.

Postman

File Edit View Help

New Import Runner My Workspace Invite

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

Untitled Request

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

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

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 642ms Size: 521 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Success!",
3   "values": [
4     {
5       "id": 2,
6       "nama": "A.SAFA DHIATA",
7       "nim": "1922",
8       "email": "SAFADHIIIIATAAAAA@gmail.com",
9       "jurusan": "teknik informatika TI 2 B",
10      "created_at": "2020-04-02T12:33:19.000000Z",
11      "updated_at": "2020-05-04T11:20:38.000000Z"
12    }
13  ]
14 }
```

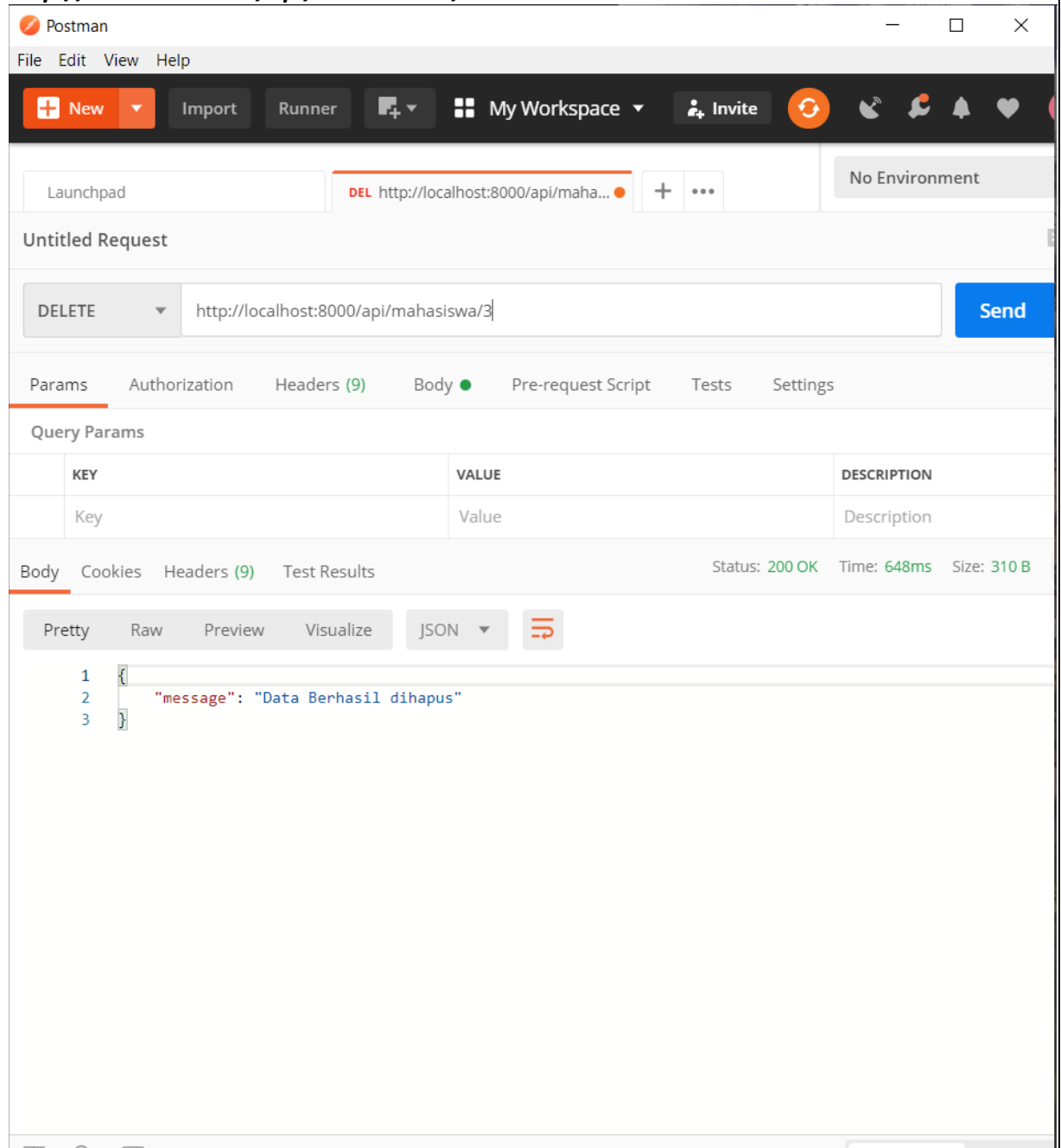
18	<p>Terakhir kita akan membuat fungsi untuk menghapus data dengan nama delete di MahasiswaController.php.</p> <pre> public function delete(\$id){ \$mhs = Mahasiswa::where('id', \$id); if (\$mhs->delete()) { \$res['message'] = "Data Berhasil dihapus"; return response(\$res); } else { \$res['message'] = "Gagal!"; return response(\$res); } } </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> • Fungsi delete menerima parameter id yang menunjukkan ID yang dipilih. • Line 92-99 : <code>\$mhs->delete()</code> digunakan untuk menghapus data dari database, apabila <code>delete()</code> berhasil dijalankan maka akan ditampilkan pesan berhasil.
19	<p>Tambahkan <i>route</i> untuk memanggil fungsi delete pada routes/api.php</p> <pre> Route::delete('mahasiswa/{id}', 'MahasiswaController@delete'); </pre> <p>Karena kita ingin menghapus data, maka perintah yang dipakai adalah 'delete.</p>

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.

-- Selamat Mengerjakan --