



Topik

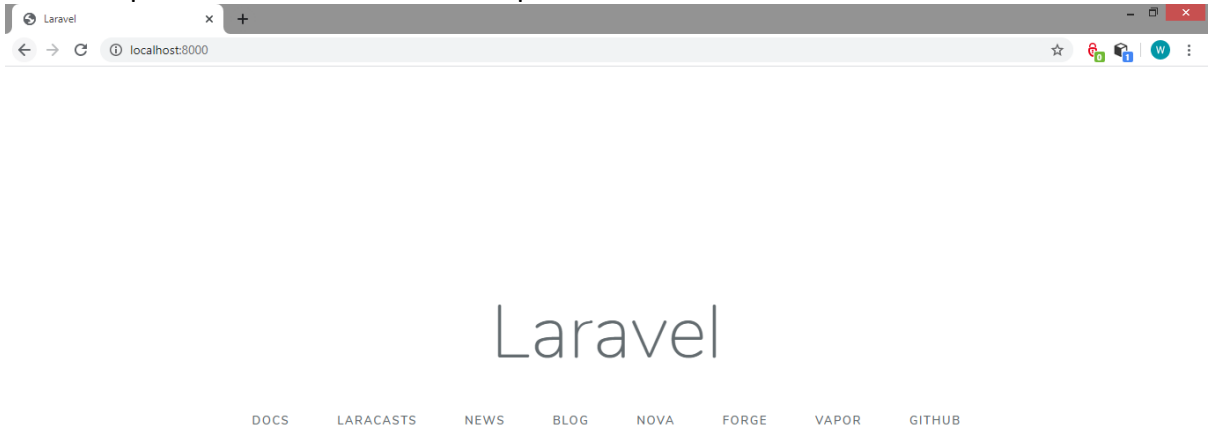
Membuat RESTful API dengan Framework Laravel

Tujuan

Mahasiswa diharapkan dapat:

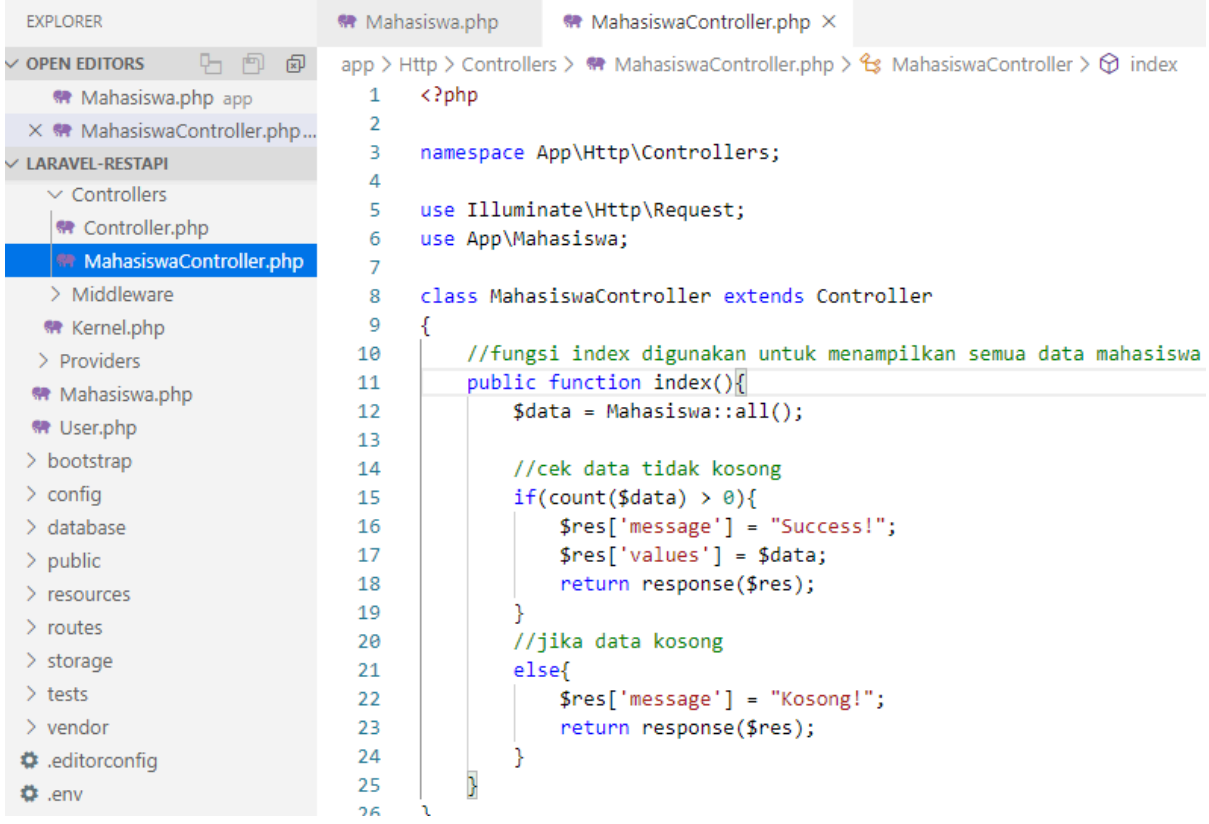
1. Memahami bagaimana cara membuat RESTful API menggunakan Laravel

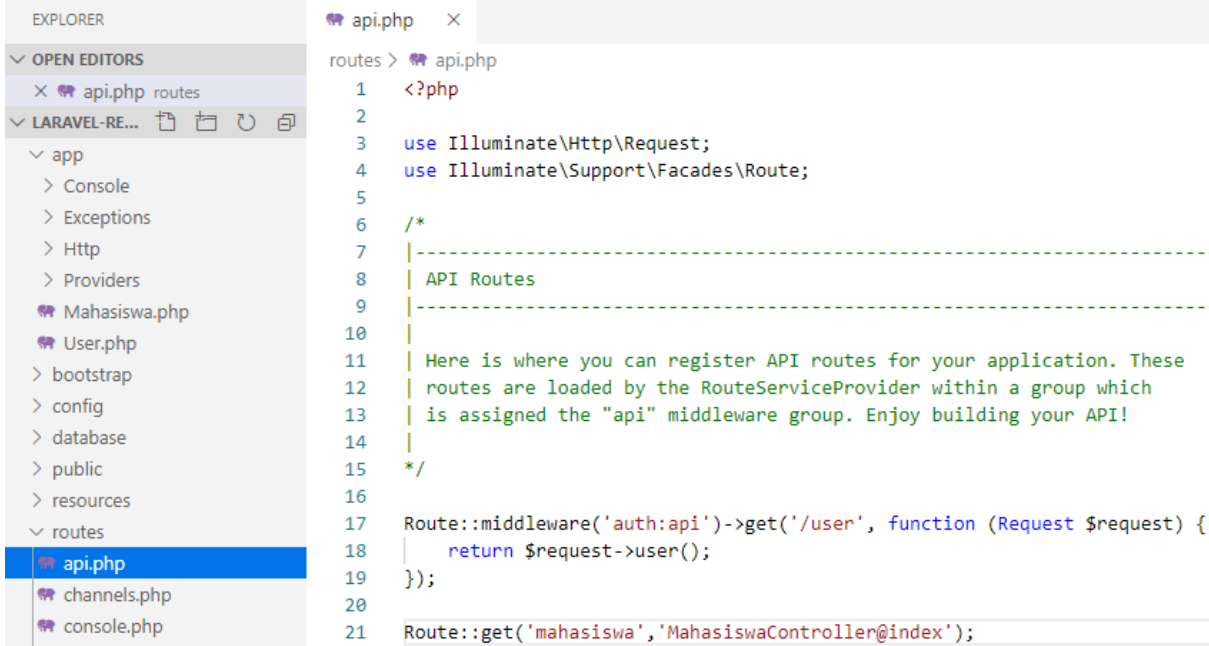
Praktikum: Membuat RESTful API di Laravel

Langkah	Keterangan
1	Buat project baru dengan nama " laravel-restapi ". Buka command prompt, tuliskan perintah berikut. cd C:\xampp\htdocs laravel new laravel-restapi
2	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. 
3	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 "

	 <pre> 1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:1EwWmJTtDMajfbBa6nP4oMTPToV2kfH0zpNXtnA5zi0= 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	<p>Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.</p>

	<div data-bbox="332 121 1534 640">  <pre> 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 </pre> </div> <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 <i>controller</i> ini, kita akan melakukan operasi untuk menampilkan, menambah, mengubah, dan menghapus data.</p> <p>Pertama, kita akan mengubah fungsi index agar saat fungsi index dipanggil, maka aplikasi akan menampilkan seluruh data dari tabel mahasiswa.</p> <div data-bbox="332 1018 1534 1837">  <pre> 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 } </pre> </div>
---	--

	<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');</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>

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

Status: 200 OK Time: 7.64s Size: 1.11 KB

```

{
  "id": 1,
  "nama": "nanda_eka",
  "nim": "1001001001",
  "email": "nandan@gmail.com",
  "jurusan": "teknik informatika",
  "created_at": null,
  "updated_at": null
},
{
  "id": 2,
  "nama": "wahyu",
  "nim": "1001001002",
  "email": "wwahyu@gmail.com",
  "jurusan": "teknik elektro"
}

```

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**.

```

//fungsi untuk menampilkan data dari sebuah ID
public function getId($id)
{
    $data = Mahasiswa::where('id',$id)->get();

    //cek jika data ditemukan
    if(count($data) > 0){
        $res['message'] = "Success!";
        $res['values'] = $data;
        return response($res);
    }

    //jika data tidak ditemukan
    else{
        $res['message'] = "Gagal!";
        return response($res);
    }
}

```

Keterangan:

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

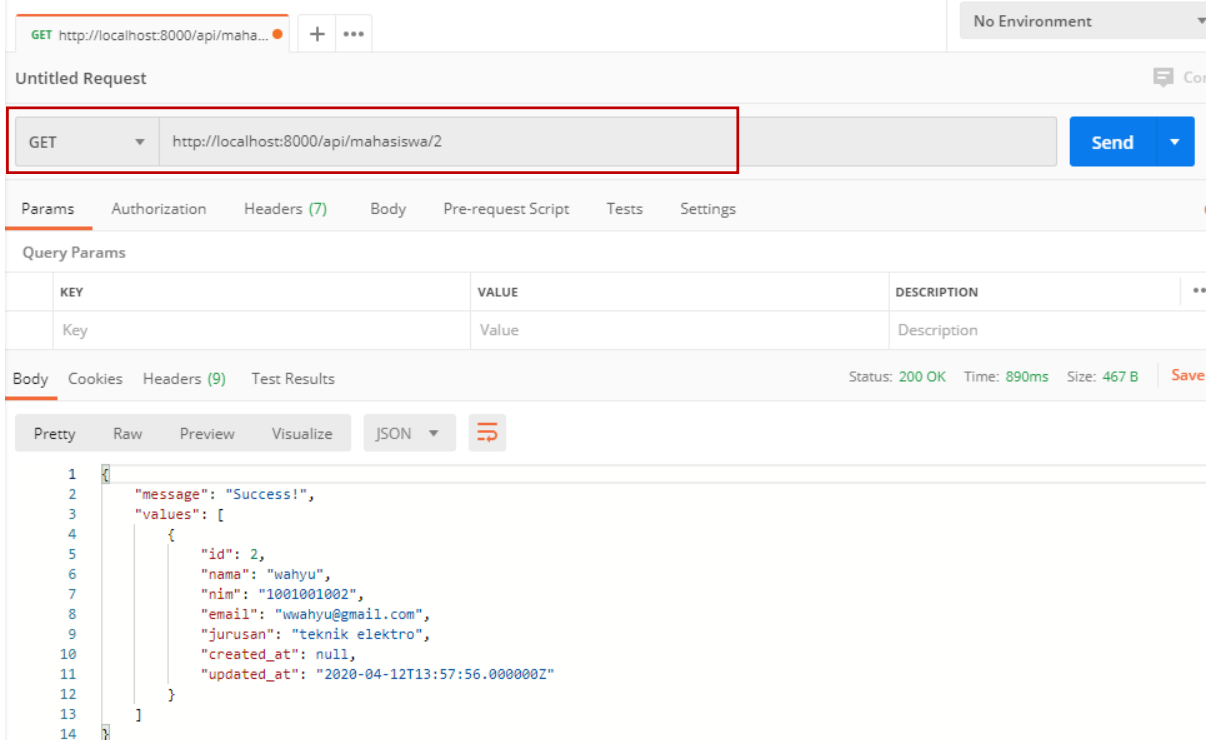
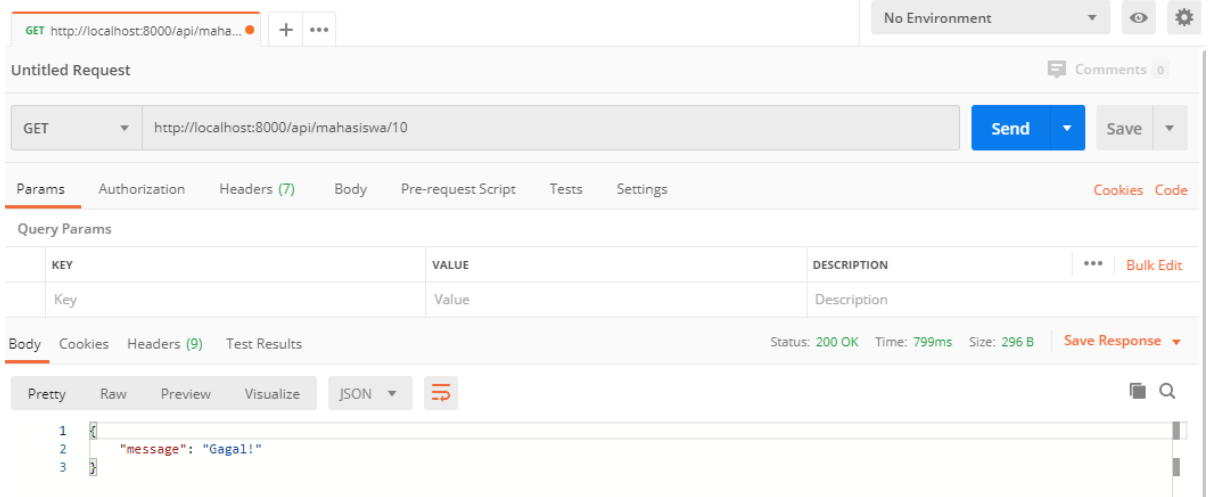
10

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

```

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

```

	Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'
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>  <p>Ketika mencoba menampilkan ID=10 akan muncul pesan "Gagal", karena tidak ada data mahasiswa dengan ID tersebut.</p> 
12	Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php .

EXPLORER

api.php

MahasiswaController.php ×

OPEN EDITORS

api.php routes

MahasiswaController.php...

LARAVEL-RESTAPI

app

Console

Exceptions

Http

Controllers

Controller.php

MahasiswaController.php

Middleware

Kernel.php

Providers

Mahasiswa.php

User.php

app > Http > Controllers > MahasiswaController.php > ...

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

59

60

```
//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 tersimpan
    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**

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

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

14

Kita coba untuk menambahkan data melalui Postman.

The screenshot shows the Postman interface with the following details:

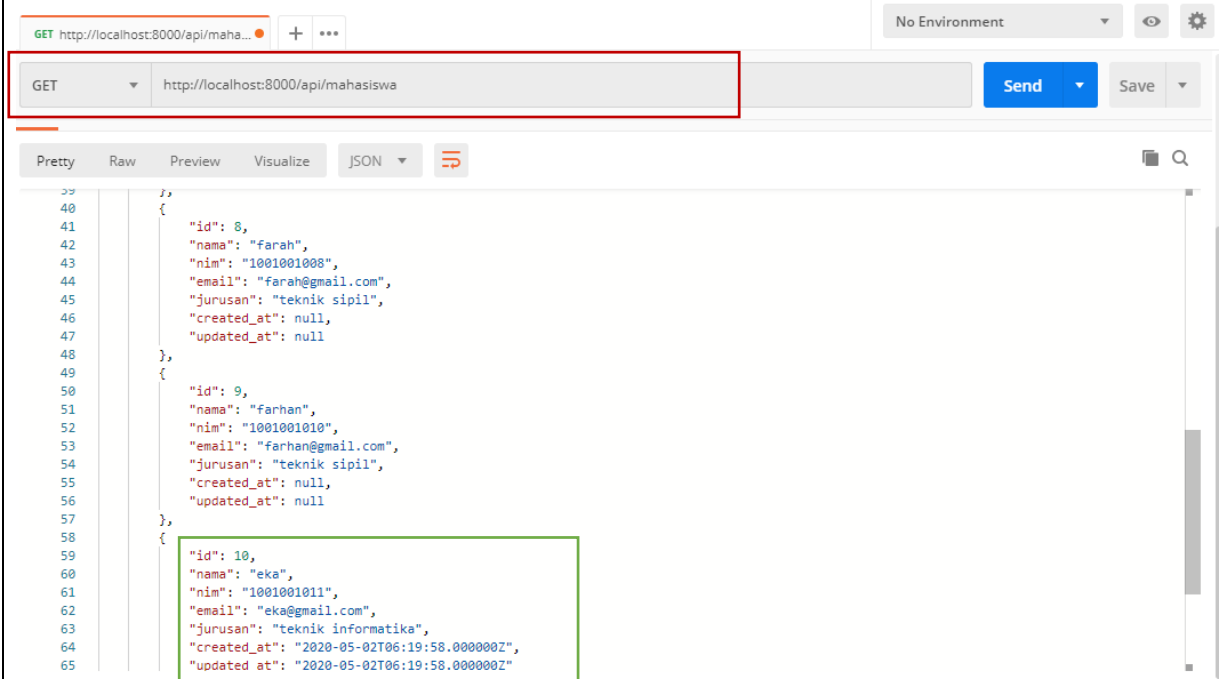
- Request:** POST http://localhost:8000/api/mahasiswa
- Body:** x-www-form-urlencoded
- Parameters:**

KEY	VALUE	DESCRIPTION
nama	eka	
nim	1001001011	
email	eka@gmail.com	
jurusan	teknik informatika	
- Response:** 200 OK, Time: 1169ms, Size: 532 B
- JSON Response:**

```
{
  "message": "Data berhasil ditambah!",
  "value": {
    "nama": "eka",
    "nim": "1001001011",
    "email": "eka@gmail.com",
    "jurusan": "teknik informatika",
    "updated_at": "2020-05-02T06:20:26.000000Z",
    "created_at": "2020-05-02T06:20:26.000000Z",
    "id": 11
  }
}
```

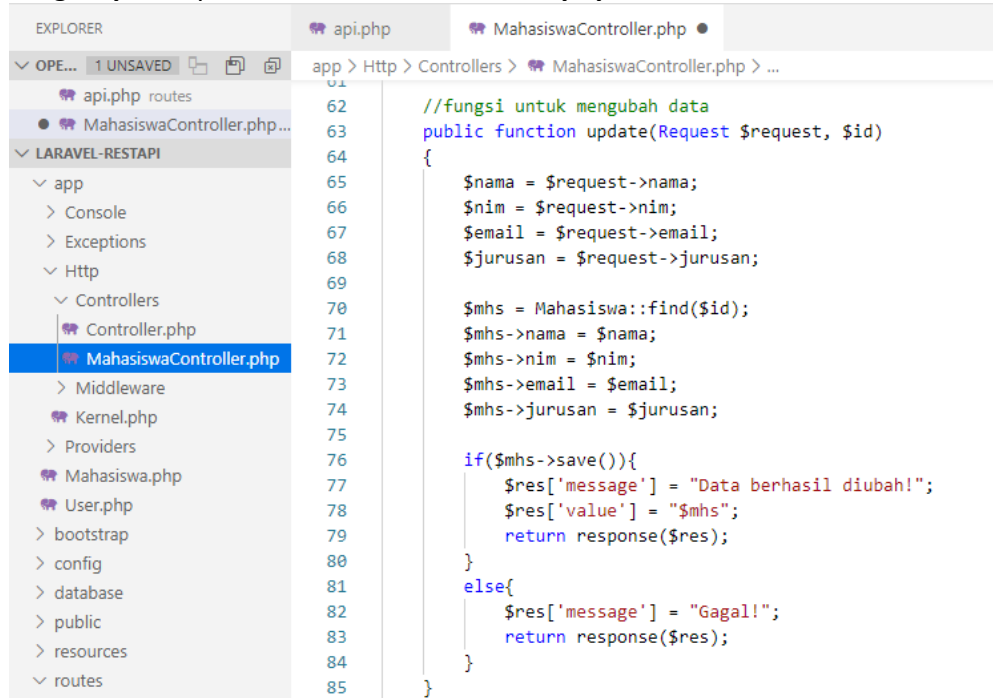
- Isikan url : ***http://localhost:8000/api/mahasiswa***. Karena kita ingin mengirim data ke database, maka perintah yang dipakai adalah '**POST**'.
- Pilih tab **Body** dan pilih radio button **x-www-form-urlencoded**. Isikan nama kolom pada database pada **KEY**, untuk isian datanya tuliskan pada **VALUE**.

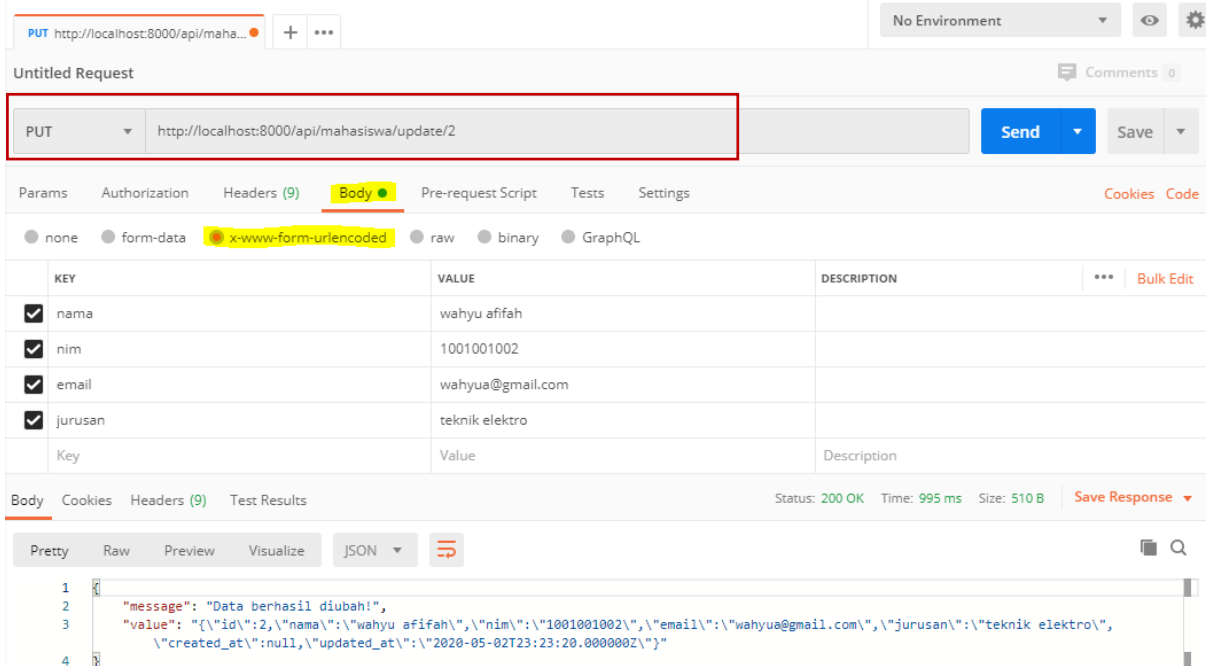
Kemudian coba untuk tampilkan semua data, kita lihat apakah data baru sudah masuk ke database.



15

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



	<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.
16	<p>Tambahkan <i>route</i> untuk memanggil fungsi update pada routes/api.php</p> <pre>26 27 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');</pre> <p>Karena kita ingin memasukkan perubahan data, maka perintah yang dipakai adalah 'put'.</p>
17	<p>Sekarang kita coba untuk mengubah data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah PUT untuk mengubah data.</p> <p>Berikut adalah contoh untuk mengubah data dengan ID=2, maka url diisi : <i>http://localhost:8000/api/mahasiswa/update/2</i>. 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.</p>  <p>Akan muncul pesan berhasil serta perubahan data dari ID=2.</p> <p>Kemudian coba untuk menampilkan data dengan ID=2 untuk melihat apakah data sudah <i>ter-update</i>.</p>

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

Status: 200 OK Time: 804 ms Size: 474 B

```

{
  "message": "Success!",
  "values": [
    {
      "id": 2,
      "nama": "wahyu afifah",
      "nim": "1001001002",
      "email": "wahyua@gmail.com",
      "jurusan": "teknik elektro",
      "created_at": null,
      "updated_at": "2020-05-02T23:23:20.000000Z"
    }
  ]
}

```

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

```

//fungsi untuk menghapus data
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**

```

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

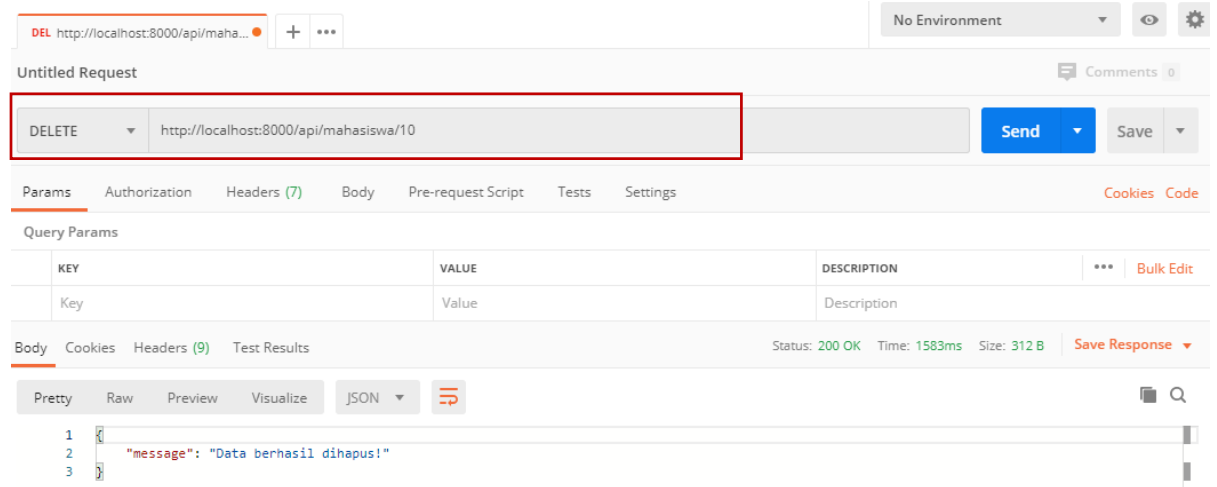
```

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=10, maka url diisi :
http://localhost:8000/api/mahasiswa /10



Muncul pesan berhasil ketika data terhapus dari database.

-- Selamat Mengerjakan --