

LAPORAN PRAKTIKUM

Jobsheet-14

Membuat RESTful API Laravel Mata Kuliah Pemrograman Web Lanjut



Oleh:
MINGGAR PUTRA DHEA R. NIM. 1841720111

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



Topik

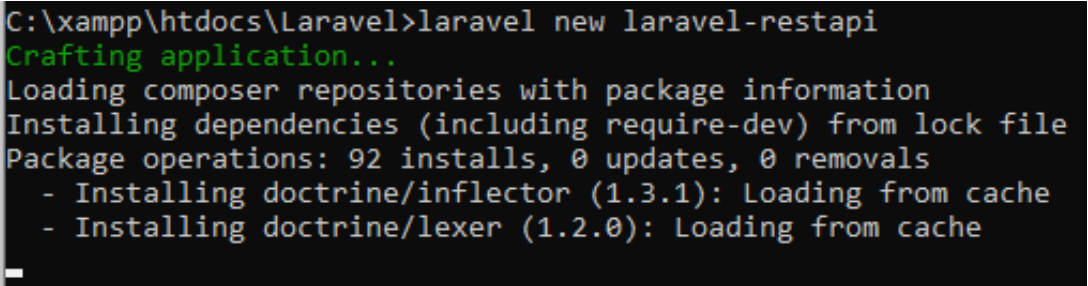
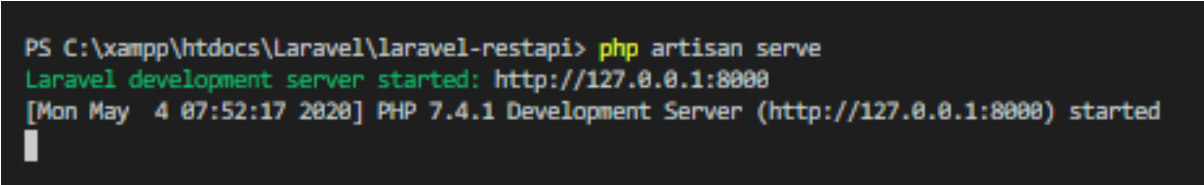
Membuat RESTful API dengan Framework Laravel

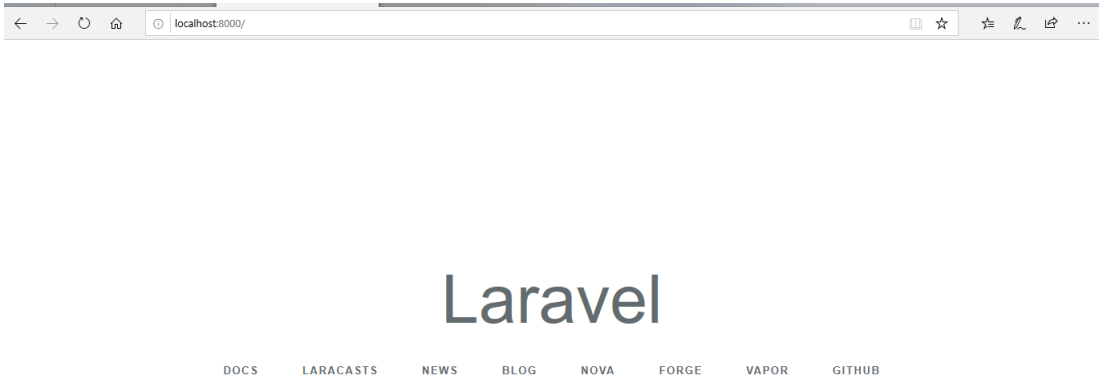
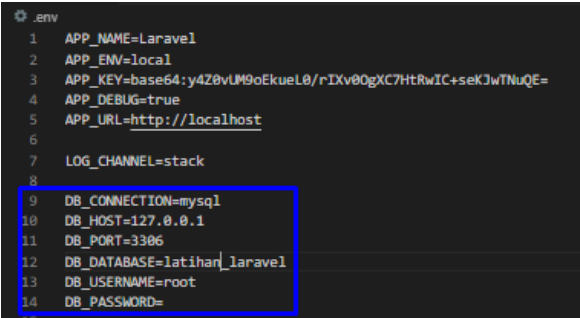
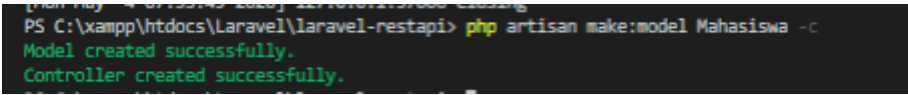
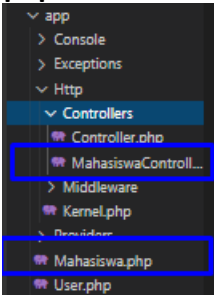
Tujuan

Mahasiswa diharapkan dapat:

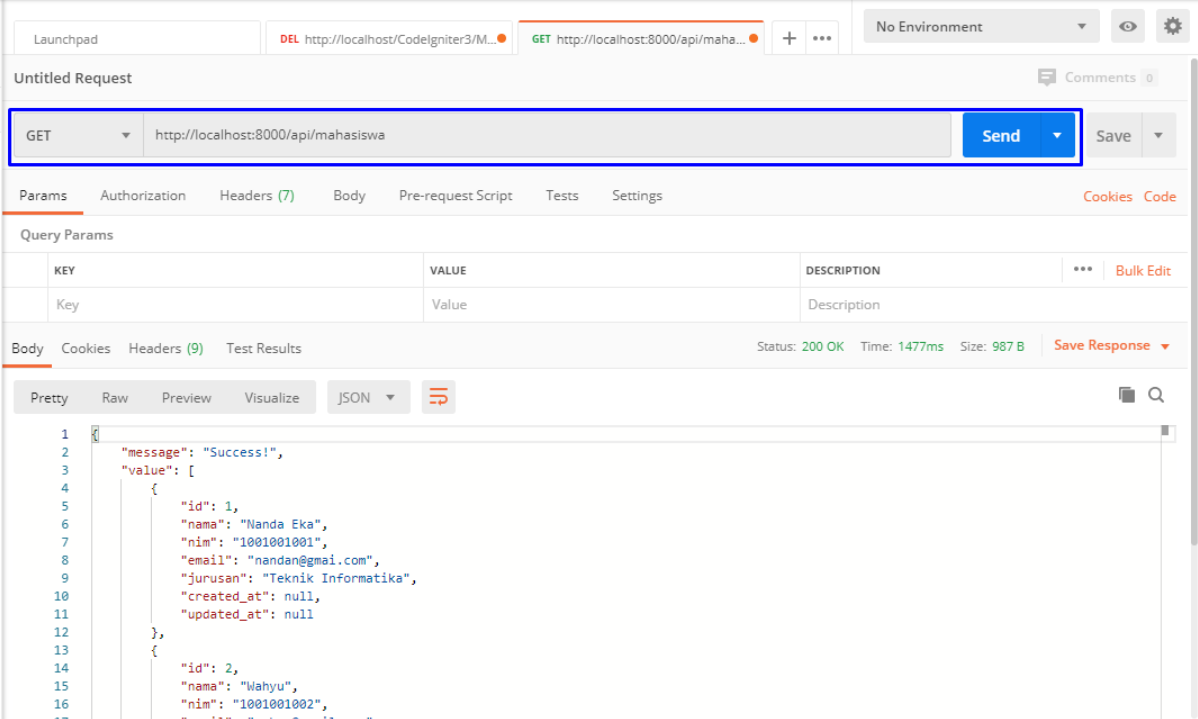
1. Memahami bagaimana cara membuat RESTful API menggunakan Laravel

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> 
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> 
4	<p>Buat model dengan nama Mahasiswa, buat juga controllernya. Untuk membuat model dan <i>controllernya</i> 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> <pre> app > Mahasiswa.php 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> <ul style="list-style-type: none"> Keterangan: 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> <pre> app > Http > Controllers > MahasiswaController.php 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['value'] = \$data; 18 return response(\$res); 19 } 20 //Jika data kosong 21 else{ 22 \$res['message'] = "Kosong!"; 23 return response(\$res); 24 } 25 } 26 } </pre> <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> 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 </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>  <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>

```

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

```

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');

```

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>

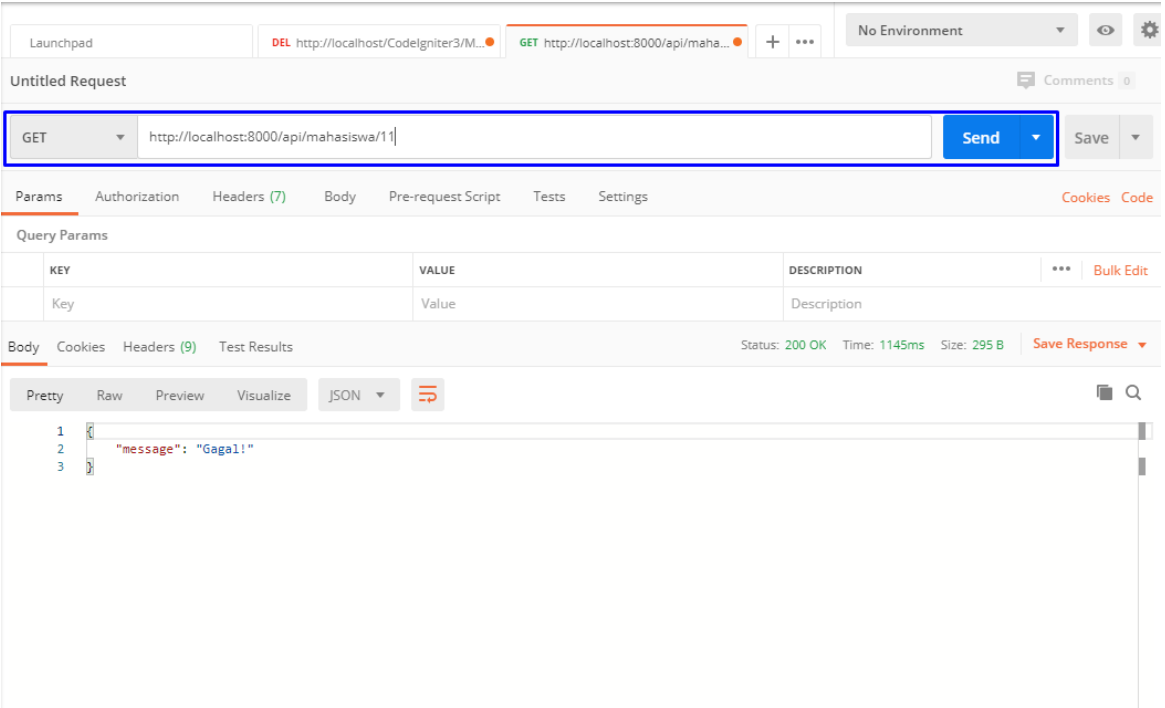
The screenshot shows the Postman interface with a GET request to `http://localhost:8000/api/mahasiswa/2`. The response status is 200 OK. The JSON response is as follows:

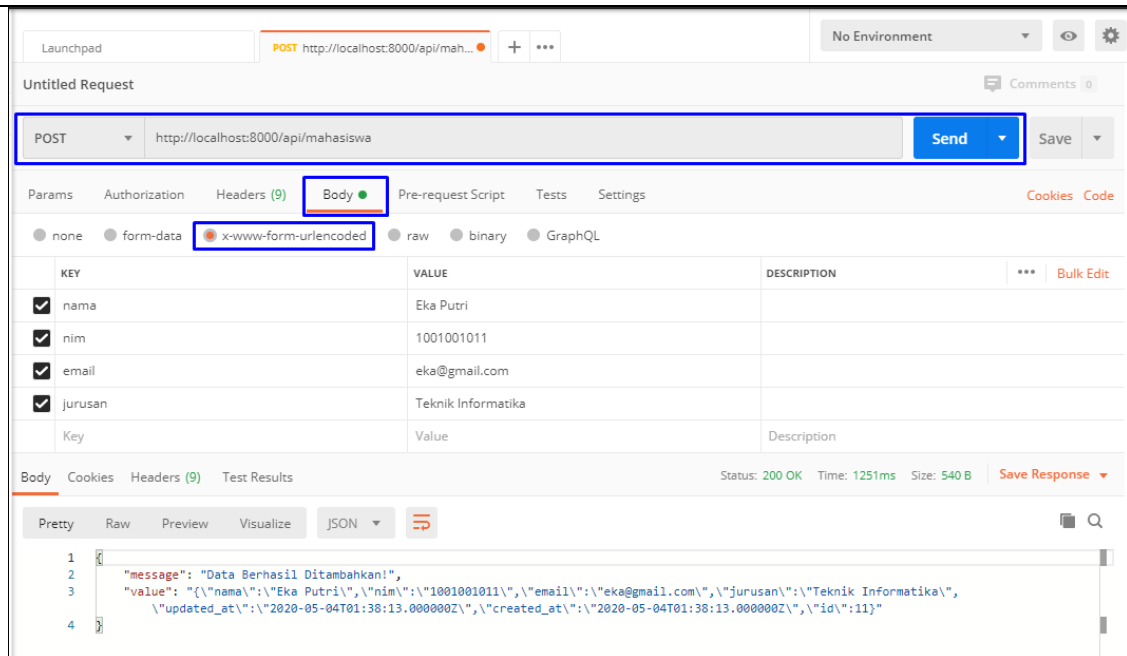
```

{
  "message": "Success!",
  "value": [
    {
      "id": 2,
      "nama": "Wahyu",
      "nim": "1001001002",
      "email": "wahyu@gmail.com",
      "jurusan": "Teknik Elektro",
      "created_at": null,
      "updated_at": null
    }
  ]
}

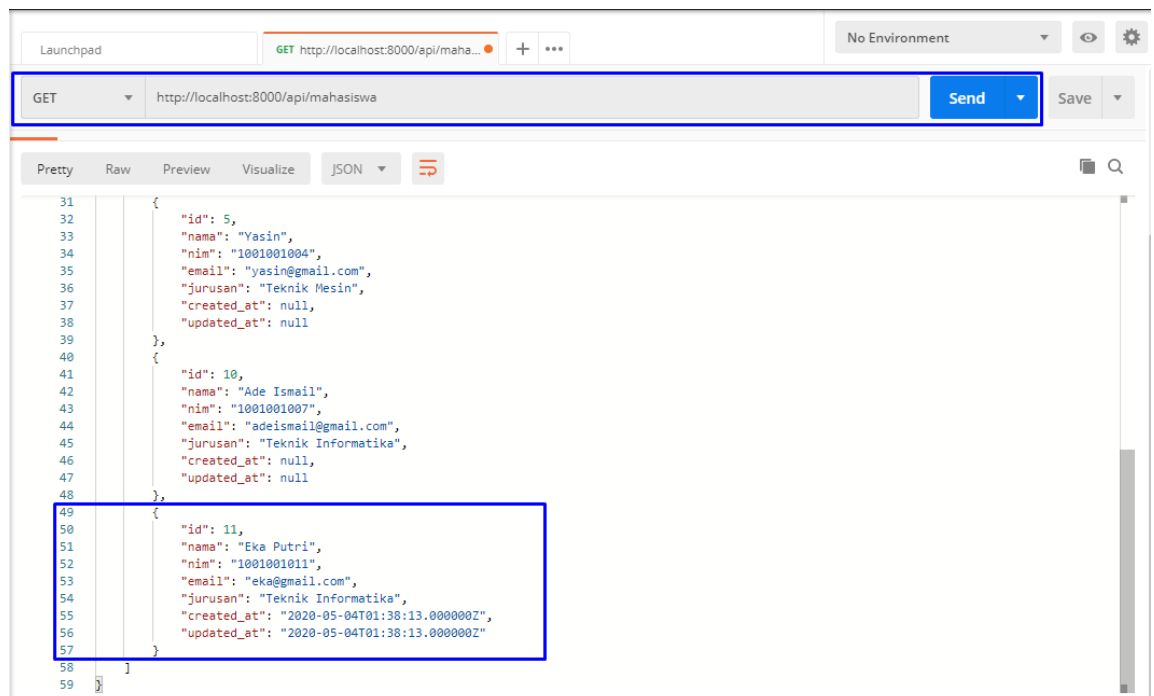
```

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

	<p>mahasiswa dengan ID tersebut.</p> 
12	<p>Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php.</p> <pre> 45 public function create(Request \$request){ 46 \$mhs = new Mahasiswa(); 47 \$mhs->nama = \$request->nama; 48 \$mhs->nim = \$request->nim; 49 \$mhs->email = \$request->email; 50 \$mhs->jurusan = \$request->jurusan; 51 52 //Jika data berhasil tersimpan 53 if(\$mhs->save()){ 54 \$res['message'] = "Data Berhasil Ditambahkan!"; 55 \$res['value'] = \$mhs; 56 return response(\$res); 57 } 58 } </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> Fungsi create menerima parameter Request yang menampung isian data mahasiswa yang akan ditambahkan ke database. <p>Line 55-59 : <code>\$mhs->save()</code> digunakan untuk menyimpan data ke database, apabila <code>save()</code> berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang ditambahkan.</p>
13	<p>Tambahkan <i>route</i> untuk memanggil fungsi <code>create</code> pada routes/api.php</p> <pre> 23 Route::post('/mahasiswa', 'MahasiswaController@create'); </pre> <p>Karena kita ingin menambah data, maka perintah yang dipakai adalah 'post'.</p>
14	<p>Kita coba untuk menambahkan data melalui Postman.</p>



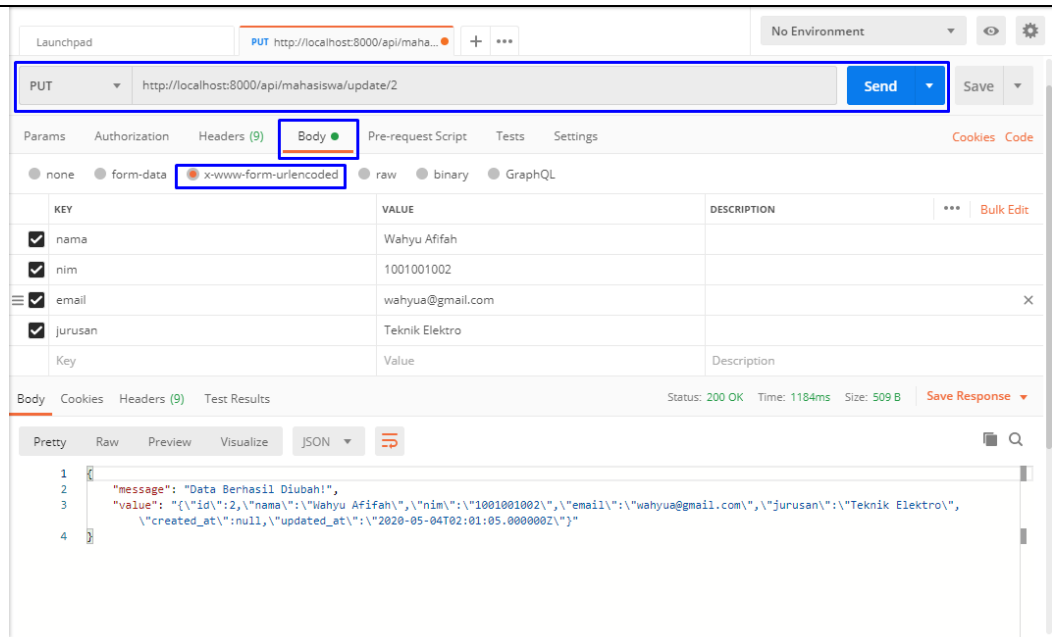
- Isikan url : **http://localhost:8000/api/mahasiswa**. Karena kita ingin mengirim data ke database, maka perintah yang dipakai adalah **'POST'**.
- Pilih tab **Body** dan pilih radio button **x-www-form-urlencoded**. Isikan nama kolom pada database pada **KEY**, untuk isian datanya tuliskan pada **VALUE**.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



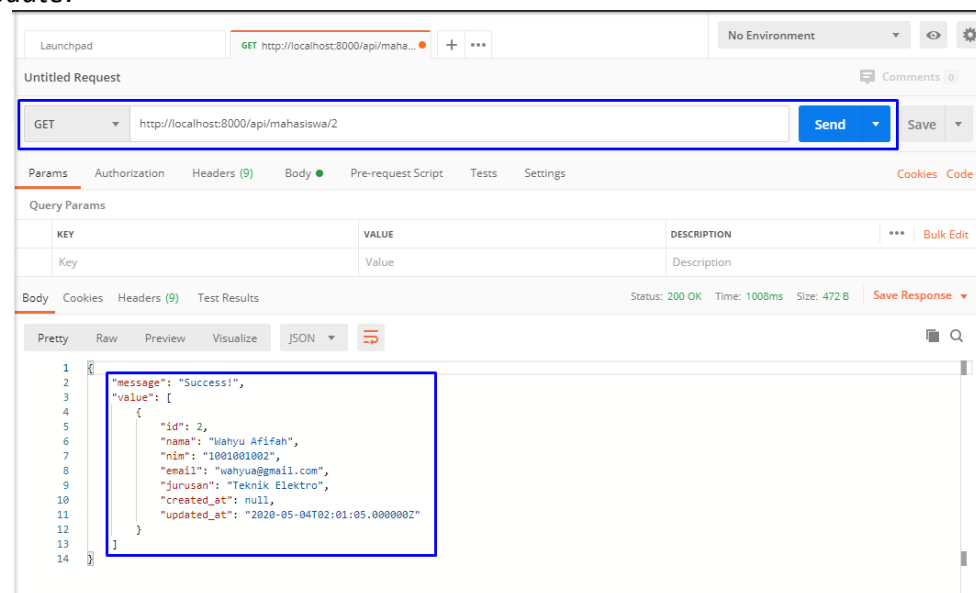
15

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

	<pre> 66 //Fungsi Untuk Mengubah / update Data 67 public function update(Request \$request, \$id){ 68 69 // \$mhs = new Mahasiswa(); 70 \$nama = \$request->nama; 71 \$nim = \$request->nim; 72 \$email = \$request->email; 73 \$jurusan = \$request->jurusan; 74 75 \$mhs = Mahasiswa::find(\$id); 76 \$mhs->nama = \$nama; 77 \$mhs->nim = \$nim; 78 \$mhs->email = \$email; 79 \$mhs->jurusan = \$jurusan; 80 81 //Jika data berhasil tersimpan 82 if(\$mhs->save()){ 83 \$res['message'] = "Data Berhasil Diubah!"; 84 \$res['value'] = \$mhs; 85 return response(\$res); 86 } 87 88 else{ 89 \$res['message'] = "Data Gagal Diubah!"; 90 return response(\$res); 91 } 92 93 </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 : <code>Mahasiswa::find(\$id)</code> digunakan untuk pencarian data pada tabel mahasiswa berdasarkan \$id. <p>Line 76-80 : <code>\$mhs->save()</code> digunakan untuk menyimpan perubahan data ke database, apabila <code>save()</code> berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang diubah.</p>
16	<p>Tambahkan <i>route</i> untuk memanggil fungsi update pada routes/api.php</p> <pre> 24 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 : 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.</p>



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.



18

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

```

94 //fungsi untuk menghapus Data
95 public function delete($id){
96     $mhs = Mahasiswa::where('id', $id);
97
98     if($mhs->delete()){
99         $res['message'] = 'Data Berhasil Dihapus!';
100         return response($res);
101     }
102     else{
103         $res['message'] = "Data Gagal Dihapus!";
104         return response($res);
105     }
106 }

```

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**

```

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

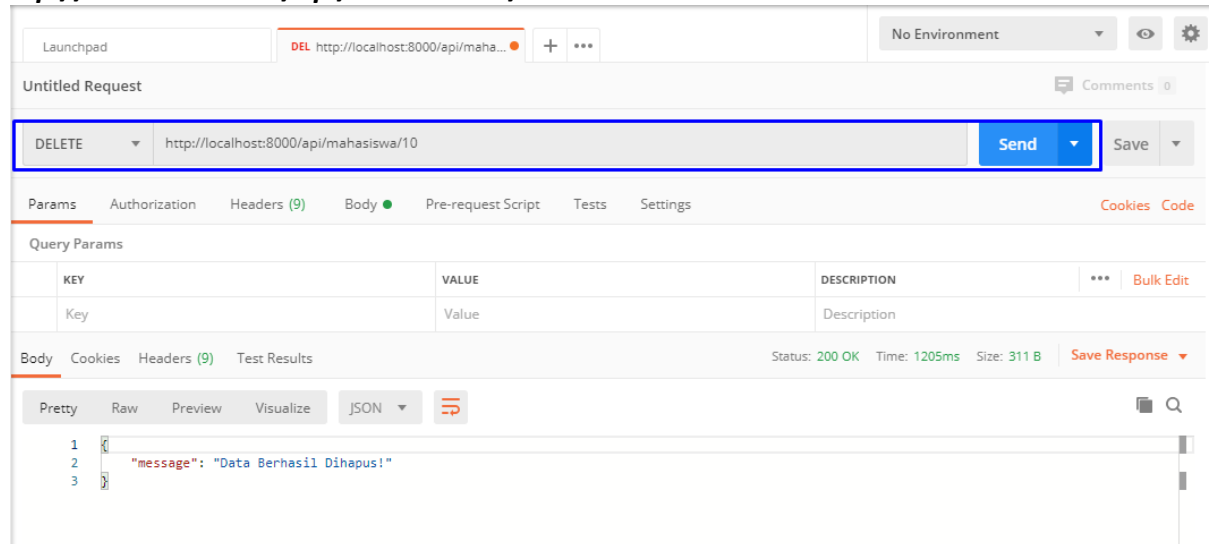
```

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 dan kita coba untuk melihat apakah data mahasiswa dengan ID=10 telah terhapus.

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

Untitled Request Comments 0

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

Params Authorization Headers (9) 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: 1026ms Size: 295 B Save Response

Pretty Raw Preview Visualize JSON

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

-- Selamat Mengerjakan --