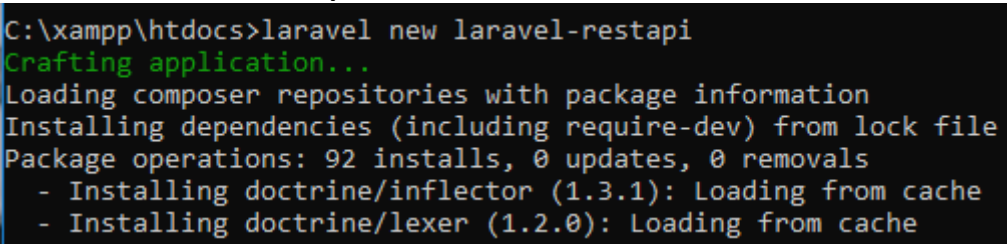
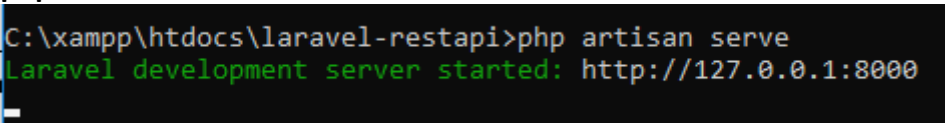
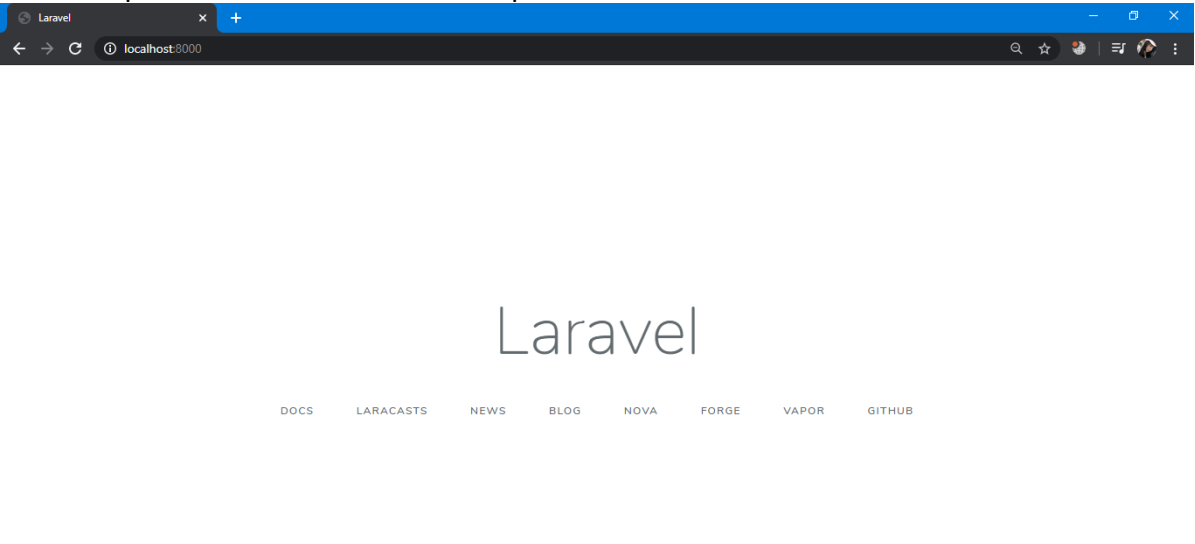


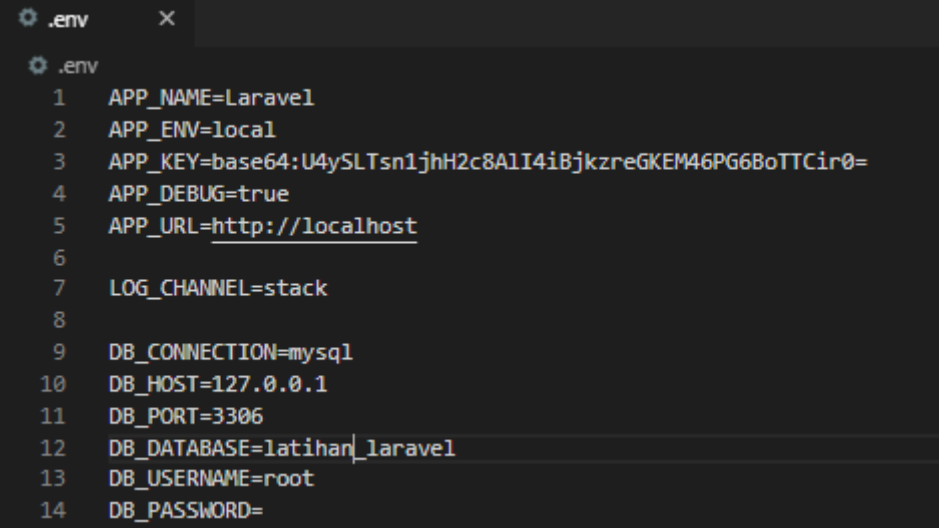
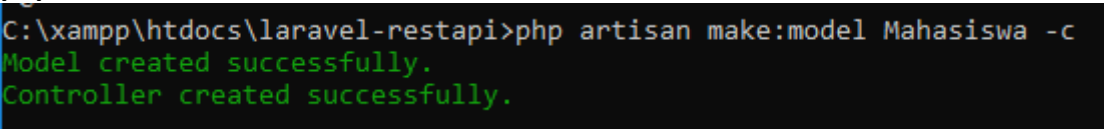
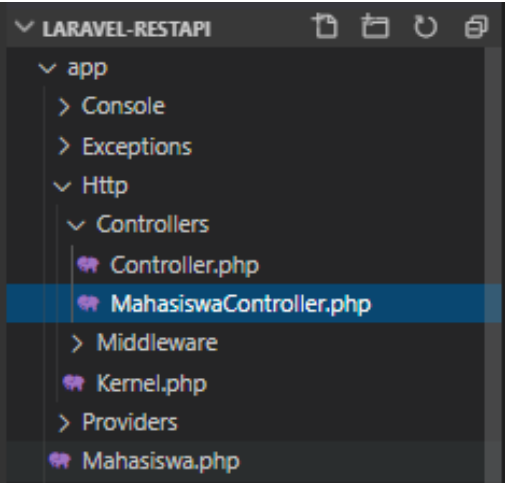
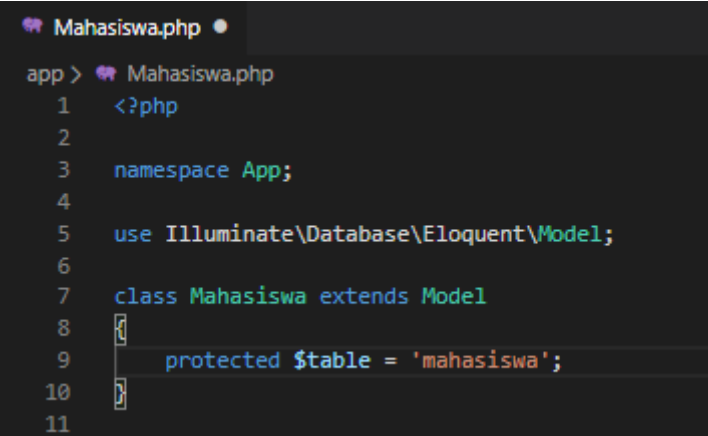


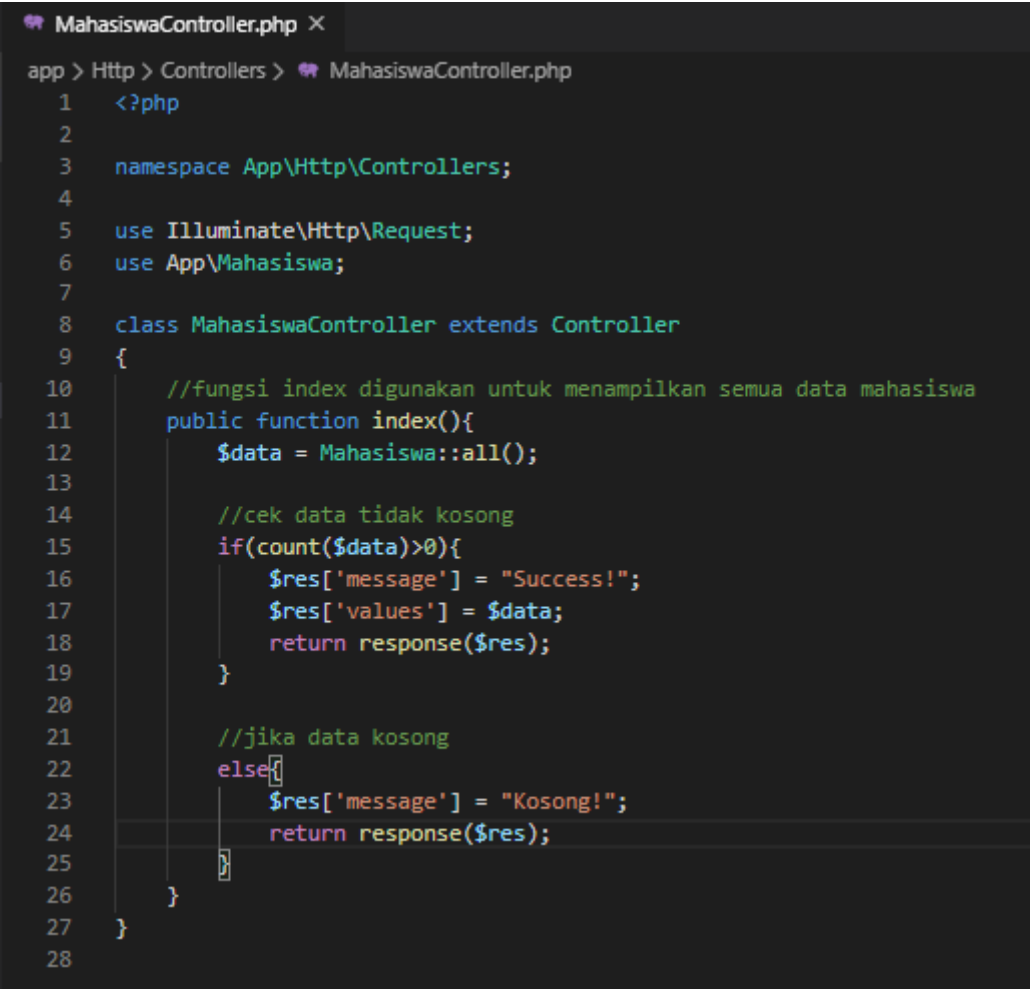
NAMA : QORINDA YULVARISMA  
KELAS : TI-2A  
NIM : 1841720084

## LAPORAN PRAKTIKUM PEMROGRAMAN WEB LANJUT

### 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 database pada file <b>.env</b>. Isikan nama database, username, dan password yang akan digunakan. Pada project ini, kita gunakan database dari latihan di minggu-minggu sebelumnya yaitu <b>"latihan_laravel"</b></p>

	 <pre> .env 1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:U4ySLTsn1jhH2c8AlI4iBjkzreGKEM46PG6BoTTCir0= 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= </pre>
4	<p>Buat <b>model</b> dengan nama <b>Mahasiswa</b>, 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><b>php artisan make:model Mahasiswa -c</b></p>  <pre> C:\xampp\htdocs\laravel-restapi&gt;php artisan make:model Mahasiswa -c Model created successfully. Controller created successfully. </pre> <p>Keterangan :</p> <ul style="list-style-type: none"> <li>-c merupakan perintah untuk menyertakan pembuatan controller</li> </ul> <p>Sehingga pada project laravel-restapi akan bertambah dua file yaitu <b>model Mahasiswa.php</b> serta <b>controller MahasiswaController.php</b>.</p>  <pre> LARAVEL-RESTAPI ├── app │   ├── Console │   ├── Exceptions │   ├── Http │   │   ├── Controllers │   │   │   ├── Controller.php │   │   │   └── MahasiswaController.php │   │   ├── Middleware │   │   ├── Kernel.php │   │   └── Providers │   └── Mahasiswa.php </pre>
5	<p>Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.</p>  <pre> Mahasiswa.php app &gt; Mahasiswa.php 1 &lt;?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>

	<p>Keterangan:</p> <ul style="list-style-type: none"> <li>Model ini akan mengelola tabel “mahasiswa” yang terdapat pada database latihan_laravel</li> </ul>
6	<p>Kemudian kita akan memodifikasi isi dari <b>MahasiswaController.php</b> 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.</p>  <pre> 1  &lt;?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)&gt;0){ 16             \$res['message'] = "Success!"; 17             \$res['values'] = \$data; 18             return response(\$res); 19         } 20 21         //jika data kosong 22         else{ 23             \$res['message'] = "Kosong!"; 24             return response(\$res); 25         } 26     } 27 } 28 </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> <li>Tambahkan line 6 agar model Mahasiswa dapat digunakan pada MahasiswaController</li> <li>Line 15-19 digunakan untuk memeriksa apakah data&gt;0 atau data tidak kosong</li> <li>Variabel \$res[message] digunakan untuk menampilkan pesan apakah ada data atau tidak ada data di tabel mahasiswa</li> <li>Variabel \$array[values] akan menyimpan semua baris data pada tabel mahasiswa</li> </ul>
7	<p>Tambahkan route untuk memanggil fungsi index pada file <b>routes/api.php</b> (Line 21).</p>

```

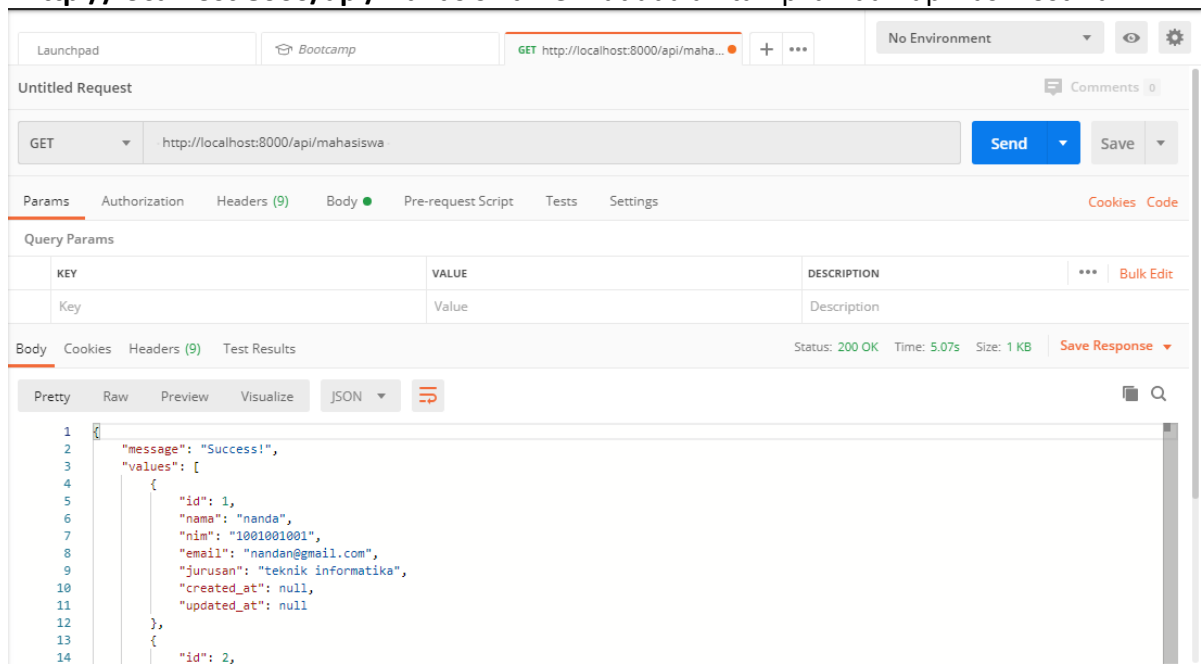
api.php x
routes > api.php
1  <?php
2
3  use Illuminate\Http\Request;
4  use Illuminate\Support\Facades\Route;
5
6  /*
7  |-----
8  |  API Routes
9  |-----
10 |
11 | Here is where you can register API routes for your application. These
12 | routes are loaded by the RouteServiceProvider within a group 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'.

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**.

```

MahasiswaController.php
app > Http > Controllers > MahasiswaController.php
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['values'] = $data;
35         return response($res);
36     }
37     //jika data tidak ditemukan
38     else{
39         $res['message'] = "Gagal!";
40         return response($res);
41     }
42 }
43 }
44

```

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**

```

api.php
routes > api.php
22
23 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>

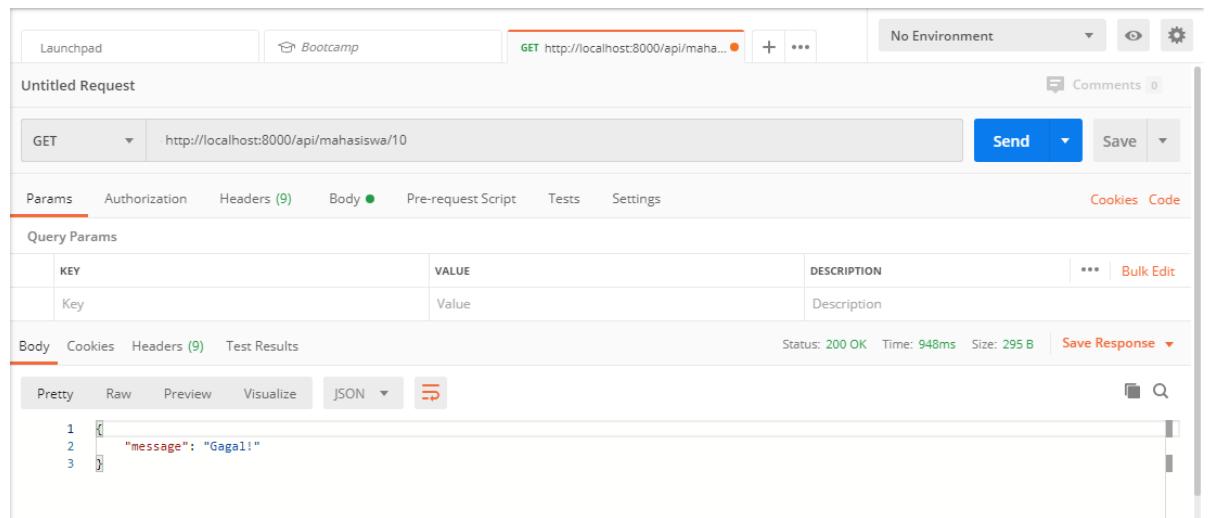
The screenshot shows the Postman interface with a GET request to `http://localhost:8000/api/mahasiswa/2`. The response status is 200 OK. The response body is a JSON object:

```

{
  "message": "Success!",
  "values": [
    {
      "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 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**.

```
MahasiswaController.php X
app > Http > Controllers > MahasiswaController.php
43
44 //fungsi tambah data
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 ditambah!";
55         $res['values'] = $mhs;
56         return response($res);
57     }
58 }
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.

13

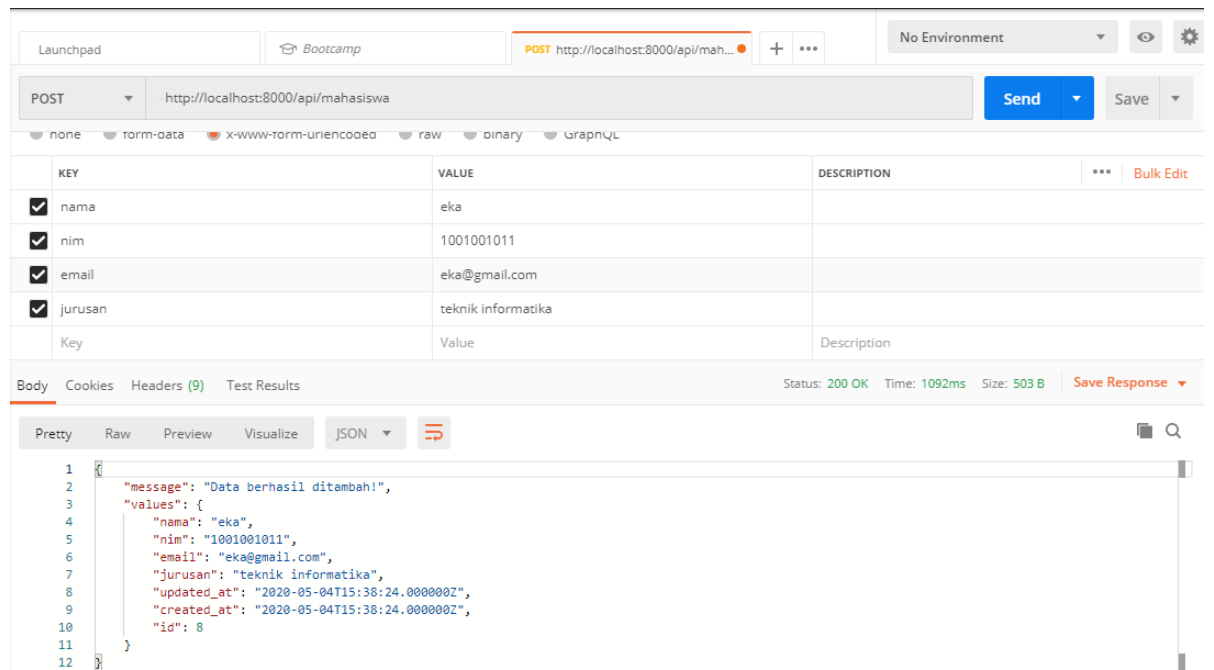
Tambahkan route untuk memanggil fungsi create pada **routes/api.php**

```
api.php X
routes > api.php
25 Route::post('/mahasiswa', 'MahasiswaController@create');
```

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

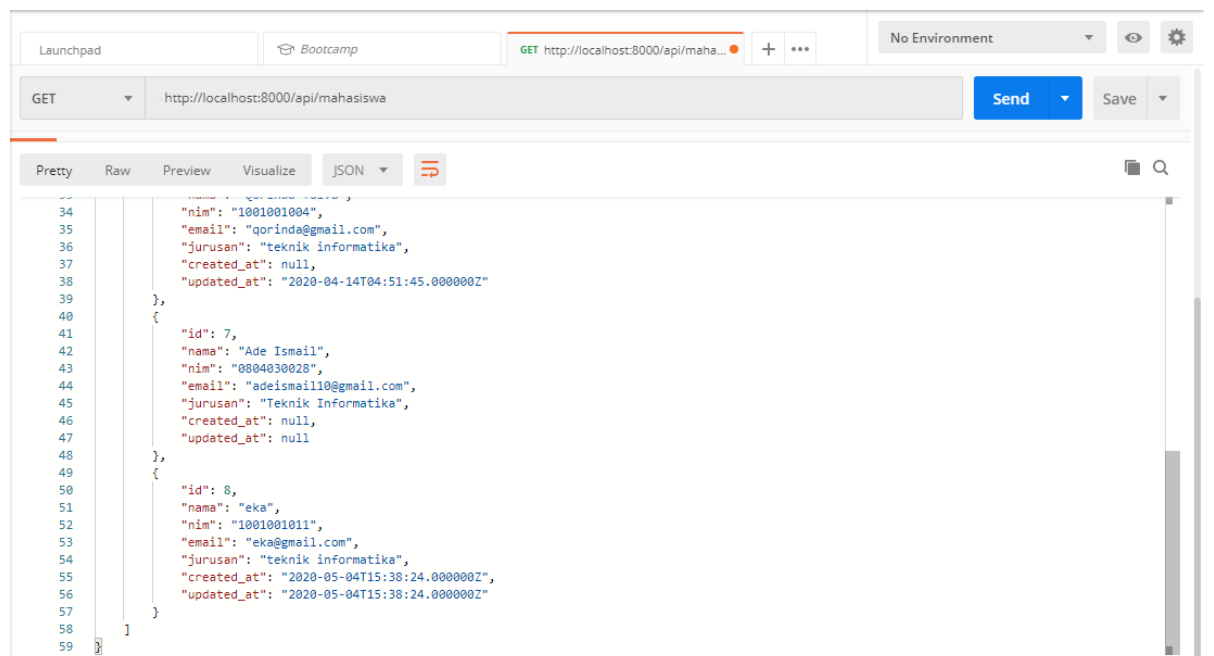
14

Kita coba untuk menambahkan data melalui Postman.



- 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**.

```

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

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

```

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**

```

api.php X
routes > api.php

27 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**.



Launchpad Bootcamp PUT http://localhost:8000/api/maha... No Environment

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

Params Authorization Headers (9) **body** Pre-request Script Tests Settings Lookies Code

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

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

Body Cookies Headers (9) Test Results Status: 200 OK Time: 1095ms Size: 483 B Save Response

Pretty Raw Preview Visualize **JSON** Raw

```

1  {
2    "message": "Data berhasil diubah!",
3    "values": {
4      "id": 2,
5      "nama": "wahyu afifah",
6      "nim": "1001001002",
7      "email": "wahyu@gmail.com",
8      "jurusan": "teknik elektro",
9      "created_at": null,
10     "updated_at": "2020-05-04T15:52:27.000000Z"
11   }

```

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.

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

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

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

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

Body Cookies Headers (9) Test Results Status: 200 OK Time: 929ms Size: 472 B Save Response

Pretty Raw Preview Visualize **JSON** Raw

```

1  {
2    "message": "Success!",
3    "values": [
4      {
5        "id": 2,
6        "nama": "wahyu afifah",
7        "nim": "1001001002",
8        "email": "wahyu@gmail.com",
9        "jurusan": "teknik elektro",
10       "created_at": null,
11       "updated_at": "2020-05-04T15:52:27.000000Z"

```

18

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

```

MahasiswaController.php x
app > Http > Controllers > MahasiswaController.php
83
84 //fungsi untuk menghapus data
85 public function delete($id){
86     $mhs = Mahasiswa::where('id',$id);
87
88     if($mhs->delete()){
89         $res['message'] = "Data berhasil dihapus!";
90         return response($res);
91     }
92     else{
93         $res['message'] = "Gagal!";
94         return response($res);
95     }
96 }
97
98
99
100

```

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**

```

api.php x
routes > api.php
29 Route::delete('/mahasiswa/{id}','MahasiswaController@delete');

```

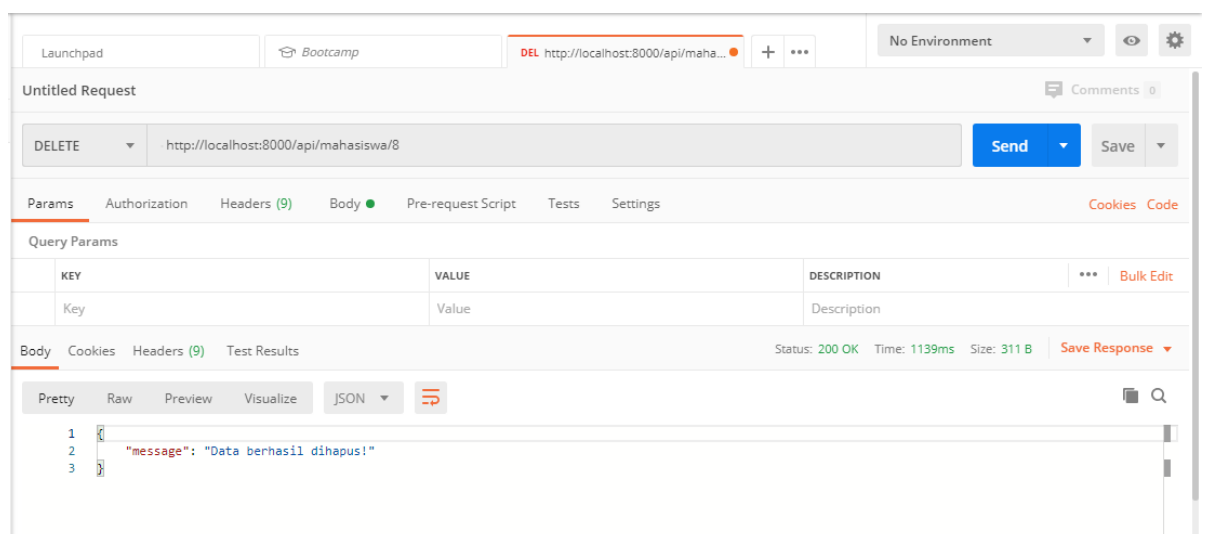
Karena kita ingin menghapus data, maka perintah yang dipakai adalah 'delete.

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.