

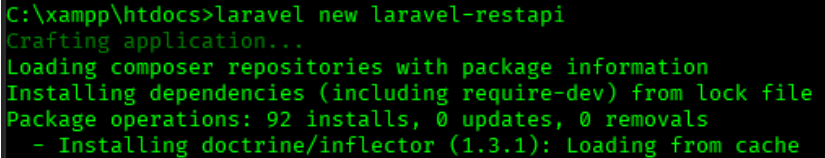
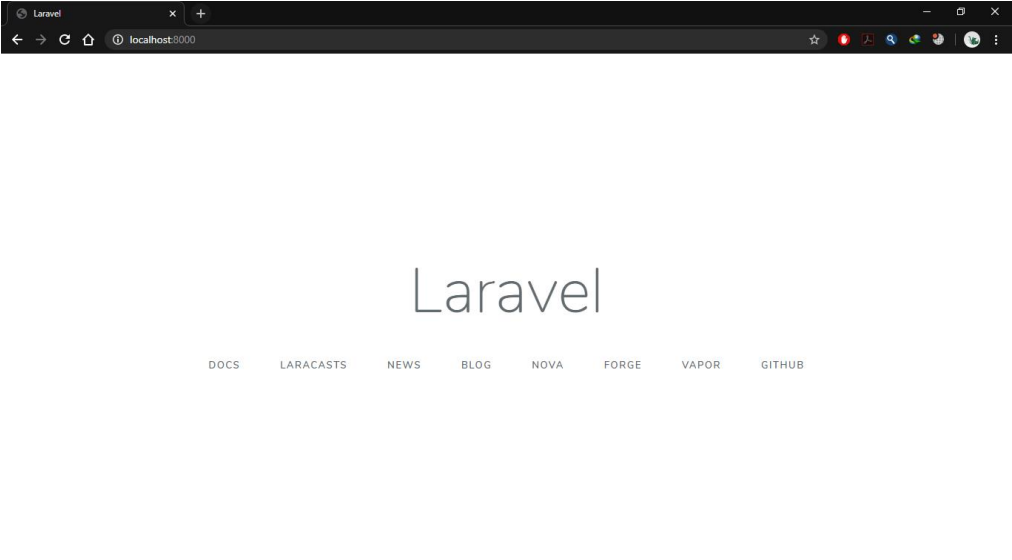
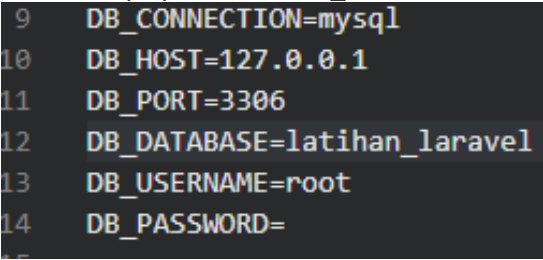
**LAPORAN PRATIKUM  
PEMROGRAMAN WEB LANJUT  
RESTFUL API LARAVEL**



**Untuk mata kuliah :  
Pemrograman Web Lanjut**

**Disusun Oleh:  
Reza Febriansyah (1841720120)**

**JURUSAN TEKNOLOGI INFORMASI  
POLITEKNIK NEGERI MALANG  
2020**

NO	KETERANGAN
1	<p>Buat project baru dengan nama <b>“laravel-restapi”</b>. 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 <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> 

4	<p>Buat <b>model</b> dengan nama <b>Mahasiswa</b>, buat juga <b>controllernya</b>.  <b>php artisan make:model Mahasiswa -c</b>  -c merupakan perintah untuk menyertakan pembuatan <i>controller</i></p> <pre>C:\xampp\htdocs\laravel-restapi&gt;php artisan make:model Mahasiswa -c Model created successfully. Controller created successfully.</pre> 
5	<p>Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.</p> 
6	<p>Kemudian kita akan memodifikasi isi dari <b>MahasiswaController</b></p> 

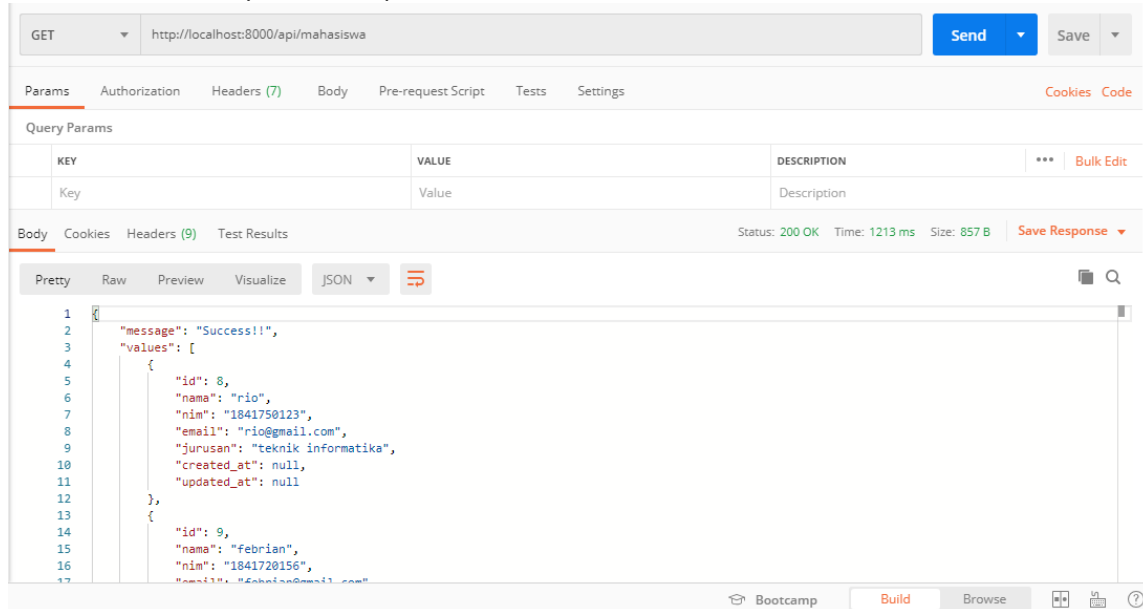
7 Tambahkan route untuk memanggil fungsi index pada file **routes/api.php** (Line 21).

```
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.



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

```
28 public function getId($id){
29
30     $data = Mahasiswa::where('id',$id)->get();
31
32     if(count($data) > 0){
33         $res['message'] = 'Success!!';
34         $res['values'] = $data;
35         return response($res);
36     }
37
38     // jika data kosong
39     else{
40         $res['message'] = "Gagal!!";
41         return response($res);
42     }
43 }
44 }
```

10

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

```
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=9, maka url diisi :

***http://localhost:8000/api/mahasiswa/9***

GET http://localhost:8000/api/mahasiswa/9

Status: 200 OK Time: 1067 ms Size: 444 B

Body

```
1 {
2   "message": "Success!!",
3   "values": [
4     {
5       "id": 9,
6       "nama": "febrian",
7       "nim": "1841720156",
8       "email": "febrian@gmail.com",
9       "jurusan": "teknik mesin",
10      "created_at": null,
11      "updated_at": null
12    }
13  ]
14 }
```

Ketika mencoba menampilkan ID=2 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut

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

Status: 200 OK Time: 5.33 s Size: 298 B

Body

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

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

```
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);  
    }  
}
```

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

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

- 14 Kita coba untuk menambahkan data melalui Postman

POST http://localhost:8000/api/mahasiswa

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
nama	eka	
nim	001004100	
email	eka@gmail.com	
jurusan	teknik informatika	
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 9.79 s Size: 531 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {  
2   "message": "Data Berhasil ditambah!",  
3   "value": "{\n  \"nama\": \"eka\",  
4     \"nim\": \"001004100\",  
     \"email\": \"eka@gmail.com\",  
     \"jurusan\": \"teknik informatika\",  
     \"updated_at\": \"2020-05-04T08:30:25.000000Z\",  
     \"created_at\": \"2020-05-04T08:30:25.000000Z\",  
     \"id\": 12\n}"
```

```
GET http://localhost:8000/api/mahasiswa Send Save
Pretty Raw Preview Visualize JSON
22 {
23   "id": 10,
24   "nama": "adi",
25   "nim": "1841720159",
26   "email": "adi@gmail.com",
27   "jurusan": "teknik informatika",
28   "created_at": null,
29   "updated_at": null
30 },
31 {
32   "id": 11,
33   "nama": "ade ismail",
34   "nim": "1841720157",
35   "email": "adeismail10@gmail.com",
36   "jurusan": "Teknik Informatika",
37   "created_at": null,
38   "updated_at": null
39 },
40 {
41   "id": 12,
42   "nama": "eka",
43   "nim": "001004100",
44   "email": "eka@gmail.com",
45   "jurusan": "teknik informatika",
46   "created_at": "2020-05-04T08:30:25.000000Z",
47   "updated_at": "2020-05-04T08:30:25.000000Z"
48 }
49 ]
50 }
```

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

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

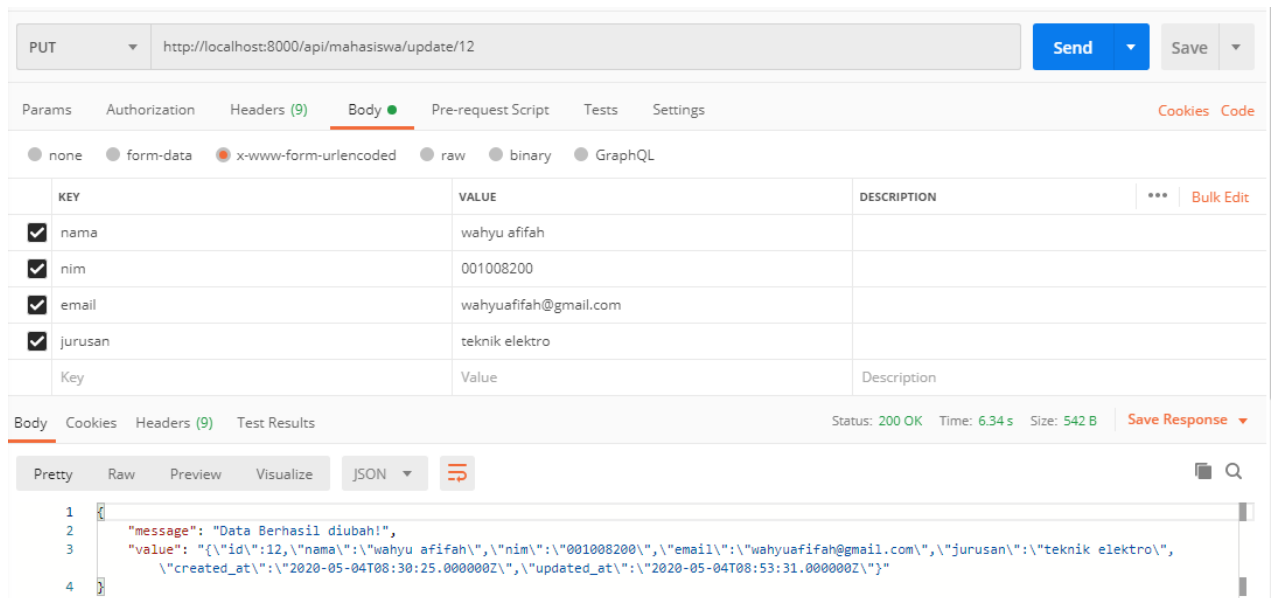
16 Tambahkan *route* untuk memanggil fungsi update pada **routes/api.php**

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

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=12, maka url diisi :

***http://localhost:8000/api/mahasiswa/update/12.***



PUT http://localhost:8000/api/mahasiswa/update/12 Send Save

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

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

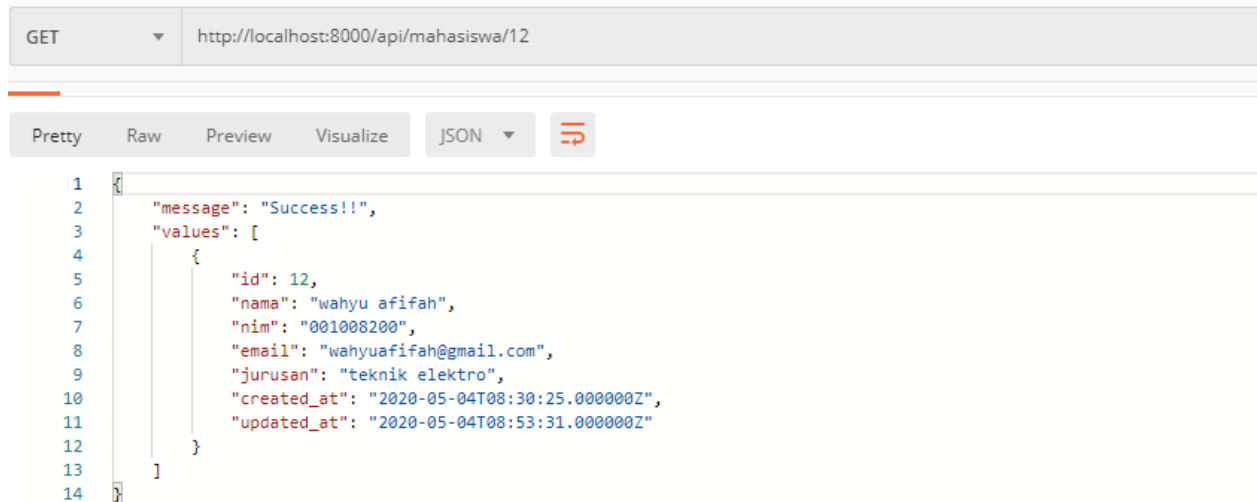
KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	wahyu afifah	
<input checked="" type="checkbox"/> nim	001008200	
<input checked="" type="checkbox"/> email	wahyuafifah@gmail.com	
<input checked="" type="checkbox"/> jurusan	teknik elektro	
Key	Value	Description

Body Cookies Headers (9) Test Results Status: 200 OK Time: 6.34 s Size: 542 B Save Response

Pretty Raw Preview Visualize JSON ⌵ ⌵

```
1 {
2   "message": "Data Berhasil diubah!",
3   "value": {"id":12,"nama":"wahyu afifah","nim":"001008200","email":"wahyuafifah@gmail.com","jurusan":"teknik elektro",
4             "created_at":"2020-05-04T08:30:25.000000Z","updated_at":"2020-05-04T08:53:31.000000Z"}
```

Kemudian coba untuk menampilkan data dengan ID=12 untuk melihat apakah data sudah ter-update.

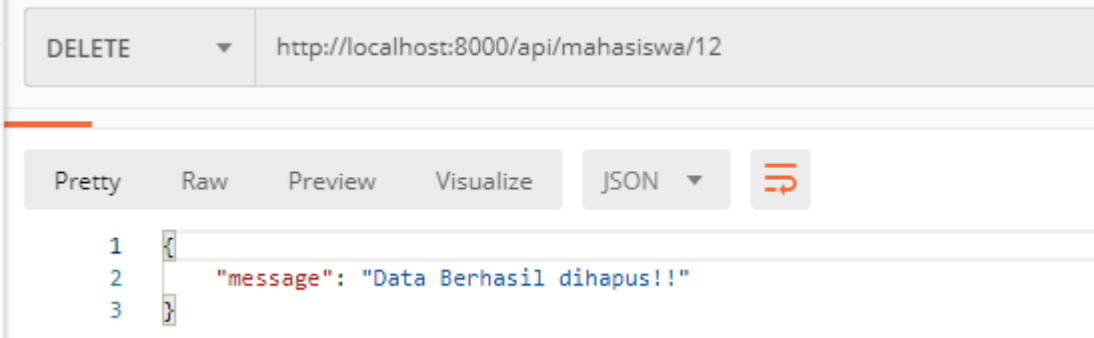


GET http://localhost:8000/api/mahasiswa/12

Pretty Raw Preview Visualize JSON ⌵ ⌵

```
1 {
2   "message": "Success!!",
3   "values": [
4     {
5       "id": 12,
6       "nama": "wahyu afifah",
7       "nim": "001008200",
8       "email": "wahyuafifah@gmail.com",
9       "jurusan": "teknik elektro",
10      "created_at": "2020-05-04T08:30:25.000000Z",
11      "updated_at": "2020-05-04T08:53:31.000000Z"
12    }
13  ]
14 }
```



18	<p>kita akan membuat fungsi untuk menghapus data dengan nama <b>delete</b> di <b>MahasiswaController.php</b>.</p> <pre> public function delete(\$id){      \$mhs = Mahasiswa::where('id',\$id);      if(\$mhs-&gt;delete()){         \$res['message'] = "Data Berhasil dihapus!!";         return response(\$res);     }     else{         \$res['message'] = "Gagal!!";         return response(\$res);     } } </pre>
19	<p>Tambahkan <i>route</i> untuk memanggil fungsi delete pada <b>routes/api.php</b></p> <pre> 29 Route::delete('/mahasiswa/{id}','MahasiswaController@delete'); 30 </pre>
20	<p>Berikut adalah contoh untuk menghapus data dengan ID=12, maka url diisi :  <b><i>http://localhost:8000/api/mahasiswa /12</i></b></p>  <p>The screenshot shows a REST client interface. At the top, the method is set to 'DELETE' and the URL is 'http://localhost:8000/api/mahasiswa/12'. Below the URL bar, there are tabs for 'Pretty', 'Raw', 'Preview', and 'Visualize', with 'Pretty' selected. To the right of these tabs is a dropdown menu set to 'JSON' and a refresh icon. The response area shows a JSON object: { "message": "Data Berhasil dihapus!!" }.</p>