

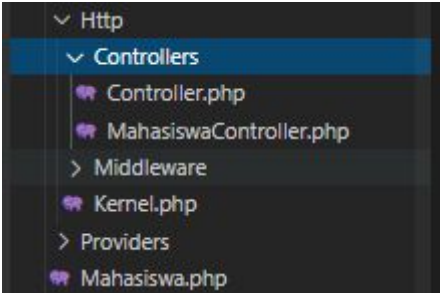
**LAPORAN JOBSHEET 13 PEMROGRAMAN WEB LANJUT**  
**Restful API Laravel**



Dibuat Oleh :

Nama : Gusti Ananda  
Nim : 1841720131  
Kelas : TI-2B

**JURUSAN TEKNOLOGI INFORMASI,  
PRODI D-IV TEKNIK INFORMATIKA  
POLINEMA 2020/2021**

No	Laporan
1	<p>konfigurasi database pada file .env. Kita akan menggunakan database latihan_laravel</p> <pre data-bbox="296 443 831 707"> DB_CONNECTION=mysql DB_HOST=localhost DB_PORT=3306 DB_DATABASE=latihan_laravel DB_USERNAME=root DB_PASSWORD= </pre>
2	<p>Buat model dengan nama Mahasiswa beserta controllernya dengan command <b>php artisan make:model Mahasiswa -c</b></p> <pre data-bbox="296 871 1386 920"> C:\xampp\htdocs\Web_Lanjut\Jobsheet13&gt;php artisan make:model Mahasiswa -c Model created successfully. </pre> 
3	<p>Ubah isi model Mahasiswa.php untuk mengelola tabel “mahasiswa” yang terdapat pada database latihan_laravel</p> <pre data-bbox="296 1377 946 1776"> 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>

4

Pada MahasiswaController.php, ubah fungsi index agar saat fungsi index dipanggil, maka aplikasi akan menampilkan seluruh data dari tabel mahasiswa.

```
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     public function index()
11     {
12         $data = Mahasiswa::all();
13
14         if (count($data) > 0) {
15             $res['message'] = "Success!";
16             $res['values'] = $data;
17             return response($res);
18         } else {
19             $res['message'] = "Kosong!";
20             return response($res);
21         }
22     }
23 }
```

5

Tambahkan Route pada api.php

```
routes > api.php
1  <?php
2
3  use Illuminate\Http\Request;
4  use Illuminate\Support\Facades\Route;
5
6  Route::middleware('auth:api')->get('/user', function (Request $request) {
7      return $request->user();
8  });
9
10 Route::get('mahasiswa', 'MahasiswaController@index');
```

6

Tambahkan fungsi baru untuk mendapatkan data di tabel mahasiswa yaitu **getId** pada **MahasiswaController.php**

```
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 }
```

Tambahkan Route pada api.php

```
23 Route::get('mahasiswa/{id}', 'MahasiswaController@getId');|
24
```

7

Tambahkan fungsi baru untuk membuat data baru di tabel mahasiswa yaitu **create** pada **MahasiswaController.php**.

```
36 public function create(Request $request){
37     $mhs = new Mahasiswa();
38     $mhs->nama = $request->nama;
39     $mhs->nim = $request->nim;
40     $mhs->email = $request->email;
41     $mhs->jurusan = $request->jurusan;
42
43     if ($mhs->save()) {
44         $res['message'] = "Data Berhasil ditambah";
45         $res['values'] = "$mhs";
46         return response($res);
47     }
48 }
```

Tambahkan Route pada api.php

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

8

Tambahkan fungsi baru untuk mengupdate data di tabel mahasiswa yaitu **update** pada **MahasiswaController.php**

```
50     public function update(Request $request, $id){
51         $nama = $request->nama;
52         $nim = $request->nim;
53         $email = $request->email;
54         $jurusan = $request->jurusan;
55
56         $mhs = Mahasiswa::find($id);
57         $mhs->nama = $nama;
58         $mhs->nim = $nim;
59         $mhs->email = $email;
60         $mhs->jurusan = $jurusan;
61
62         if ($mhs->save()) {
63             $res['message'] = "Data Berhasil diubah";
64             $res['values'] = "$mhs";
65             return response($res);
66         } else {
67             $res['message'] = "Gagal!";
68             return response($res);
69         }
70     }
```

Tambahkan Route pada api.php

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

9

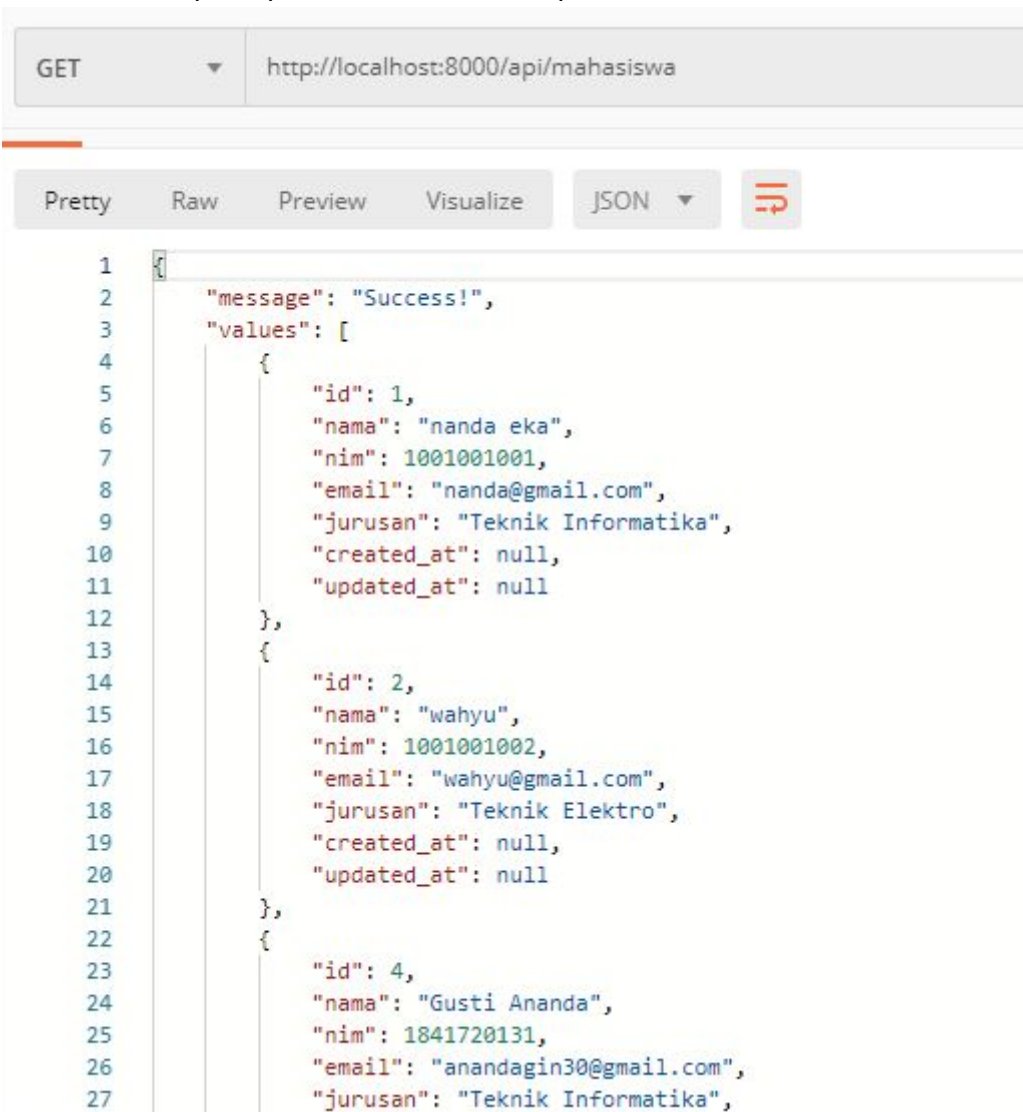
Tambahkan fungsi untuk menghapus data di tabel mahasiswa. yaitu **delete** pada **MahasiswaController.php**.

```
72     public function delete($id){
73
74         $mhs = Mahasiswa::where('id', $id);
75
76         if ($mhs->delete()) {
77             $res['message'] = "Data Berhasil dihapus";
78             return response($res);
79         } else {
80             $res['message'] = "Gagal!";
81             return response($res);
82         }
83
84     }
```

Tambahkan Route pada api.php

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

10

Gunakan **GET** pada postman untuk mendapatkan data

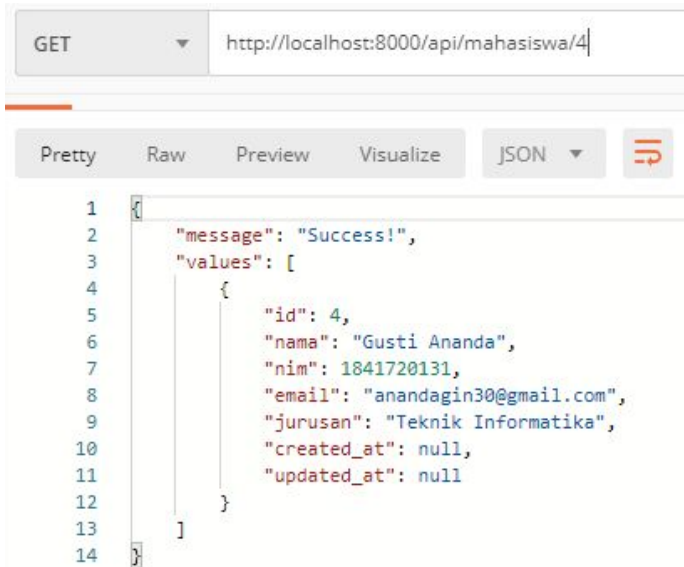
The screenshot shows the Postman interface. At the top, the method is set to 'GET' and the URL is 'http://localhost:8000/api/mahasiswa'. Below the URL bar, there are tabs for 'Pretty', 'Raw', 'Preview', and 'Visualize', with 'Pretty' selected. To the right of these tabs is a dropdown menu set to 'JSON' and a red icon. The response body is displayed in a code editor with line numbers 1 through 27. The JSON data is as follows:

```
1 {
2   "message": "Success!",
3   "values": [
4     {
5       "id": 1,
6       "nama": "nanda eka",
7       "nim": 1001001001,
8       "email": "nanda@gmail.com",
9       "jurusan": "Teknik Informatika",
10      "created_at": null,
11      "updated_at": null
12    },
13    {
14      "id": 2,
15      "nama": "wahyu",
16      "nim": 1001001002,
17      "email": "wahyu@gmail.com",
18      "jurusan": "Teknik Elektro",
19      "created_at": null,
20      "updated_at": null
21    },
22    {
23      "id": 4,
24      "nama": "Gusti Ananda",
25      "nim": 1841720131,
26      "email": "anandagin30@gmail.com",
27      "jurusan": "Teknik Informatika",
```



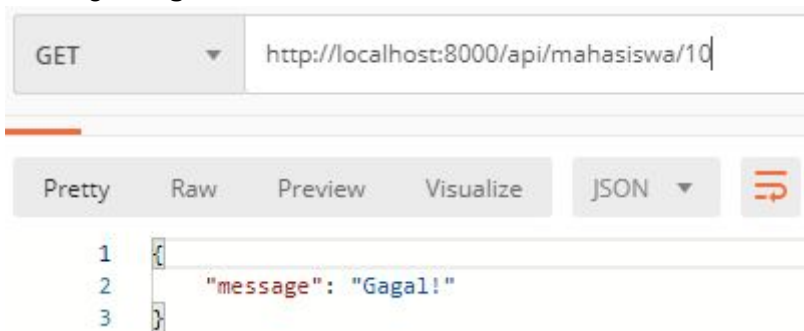
11

Gunakan **GET** lalu tambahkan id mahasiswa pada postman untuk mendapatkan data sesuai id yang dimasukkan



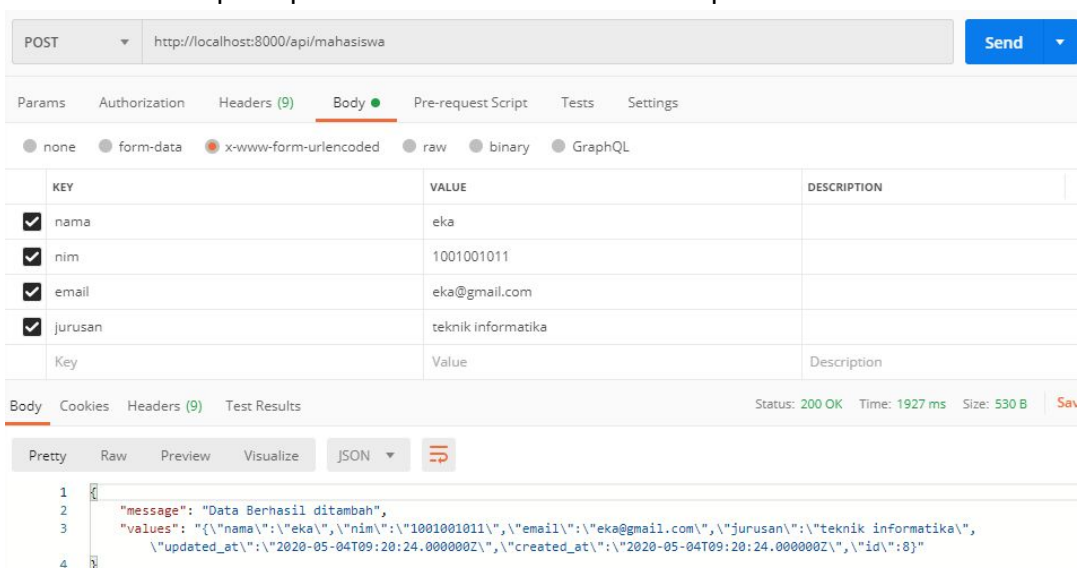
12

Apabila kita mencoba memasukkan id mahasiswa yang salah maka akan muncul message **Gagal!**



13

Gunakan **POST** pada postman untuk menambah data pada tabel mahasiswa



14

Saat kita cek menggunakan **GET** maka data yang baru saja dibuat akan muncul

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

Pretty Raw Preview Visualize JSON

[
  {
    "nama": "wahyu",
    "nim": 1001001002,
    "email": "wahyu@gmail.com",
    "jurusan": "Teknik Elektro",
    "created_at": null,
    "updated_at": null
  },
  {
    "id": 4,
    "nama": "Gusti Ananda",
    "nim": 1841720131,
    "email": "anandagin30@gmail.com",
    "jurusan": "Teknik Informatika",
    "created_at": null,
    "updated_at": null
  },
  {
    "id": 8,
    "nama": "eka",
    "nim": 1001001011,
    "email": "eka@gmail.com",
    "jurusan": "teknik informatika",
    "created_at": "2020-05-04T09:20:24.000000Z",
    "updated_at": "2020-05-04T09:20:24.000000Z"
  }
]
```



15

Gunakan **PUT** lalu tambahkan id mahasiswa pada postman untuk mengubah isi data sesuai id yang dimasukkan

The screenshot shows a Postman PUT request to the URL `http://localhost:8000/api/mahasiswa/update/2`. The request body is set to `x-www-form-urlencoded`. The body contains the following data:

KEY	VALUE	DESCRIPTION
<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

The response status is 200 OK, with a time of 1428 ms and a size of 508 B. The response body is shown in JSON format:

```

1 {
2   "message": "Data Berhasil diubah",
3   "values": "{\"id\":\"2\",\"nama\":\"wahyu afifah\",\"nim\":\"1001001002\",\"email\":\"wahyu@gmail.com\",\"jurusan\":\"teknik elektro\",
4     \"created_at\":null,\"updated_at\":\"2020-05-04T09:23:29.000000Z\"}"

```

16

Gunakan **DELETE** lalu tambahkan id mahasiswa pada postman untuk menghapus data sesuai id yang dimasukkan

The screenshot shows a Postman DELETE request to the URL `http://localhost:8000/api/mahasiswa/8`. The request body is set to `x-www-form-urlencoded`. The body contains the following data:

KEY	VALUE
<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

The response status is 200 OK, with a time of 1428 ms and a size of 508 B. The response body is shown in JSON format:

```

1 {
2   "message": "Data Berhasil dihapus"
3 }

```

Saat kita cek menggunakan **GET** maka data yang tadi dibuat akan terhapus

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

Pretty Raw Preview Visualize JSON

1 {
2   "message": "Success!",
3   "values": [
4     {
5       "id": 1,
6       "nama": "nanda eka",
7       "nim": 1001001001,
8       "email": "nanda@gmail.com",
9       "jurusan": "Teknik Informatika",
10      "created_at": null,
11      "updated_at": null
12    },
13    {
14      "id": 2,
15      "nama": "wahyu afifah",
16      "nim": 1001001002,
17      "email": "wahyu@gmail.com",
18      "jurusan": "teknik elektro",
19      "created_at": null,
20      "updated_at": "2020-05-04T09:23:29.000000Z"
21    },
22    {
23      "id": 4,
24      "nama": "Gusti Ananda",
25      "nim": 1841720131,
26      "email": "anandagin30@gmail.com",
27      "jurusan": "Teknik Informatika",
```