

LAPORAN PEMROGRAMAN WEB LANJUT

JOB SHEET 14

MEMBUAT RESTFUL API LARAVEL

Oleh:

KINANTI PERMATA PUTRI

NIM. 1841720022



PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

MEI 2020

A. Praktikum: Membuat RESTful API di Laravel

- 1) Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut.

cd C:\xampp\htdocs>laravel new laravel-restapi

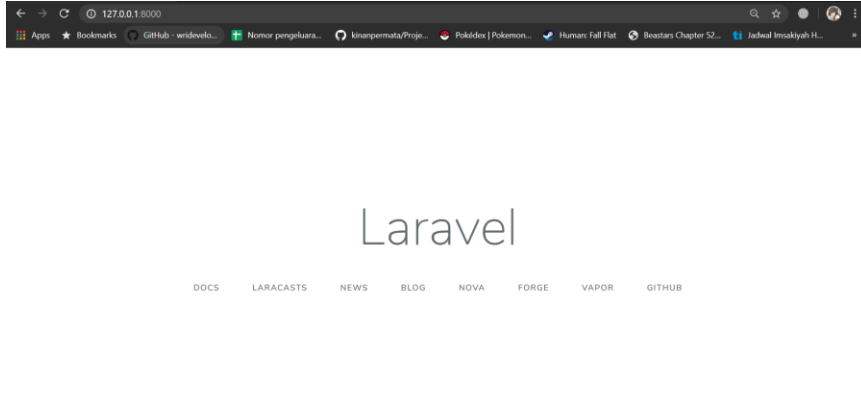
```
C:\xampp\htdocs>laravel new laravel-restapi
Crafting application...
Loading composer repositories with package information
Installing dependencies (including require-dev) from lock file
Package operations: 92 installs, 0 updates, 0 removals
- Installing doctrine/inflector (1.3.1): Loading from cache
- Installing doctrine/lexer (1.2.0): Loading from cache
- Installing dragonmantank/cron-expression (v2.3.0): Loading from cache
```

- 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.

```
C:\xampp\htdocs\laravel-restapi>php artisan serve
Laravel development server started: http://127.0.0.1:8000
```



- 3) Kemudian lakukan konfigurasi database 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”

```
.env
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:q40Uc8ZU71cWMSNJX9DzFNkzVPVUHJoib1QUKkEaMrM=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=latihan_laravel
DB_USERNAME=root
DB_PASSWORD=
```

- 4) Buat model dengan nama Mahasiswa, 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)

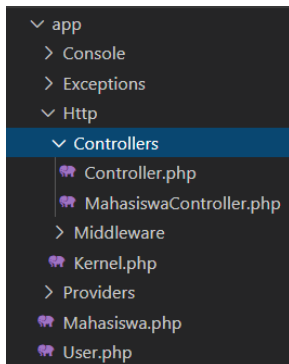
php artisan make:model Mahasiswa -c

```
C:\xampp\htdocs\laravel-restapi>php artisan make:model Mahasiswa -c
Model created successfully.
Controller created successfully.
```

Keterangan:

- -c merupakan perintah untuk menyertakan pembuatan controller

Sehingga pada project laravel-restapi akan bertambah dua file yaitu model Mahasiswa.php serta controller MahasiswaController.php.



- 5) Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.

```
app > Mahasiswa.php > PHP Intelephense > Mahasiswa
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
```

Keterangan:

- Model ini akan mengelola tabel “mahasiswa” yang terdapat pada database latihan_laravel

- 6) Kemudian kita akan memodifikasi isi dari MahasiswaController.php 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.

```
app > Http > Controllers > MahasiswaController.php > PHP Intelephense > MahasiswaControl
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Mahasiswa;
6  use Illuminate\Http\Request;
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 }
27
```

Keterangan:

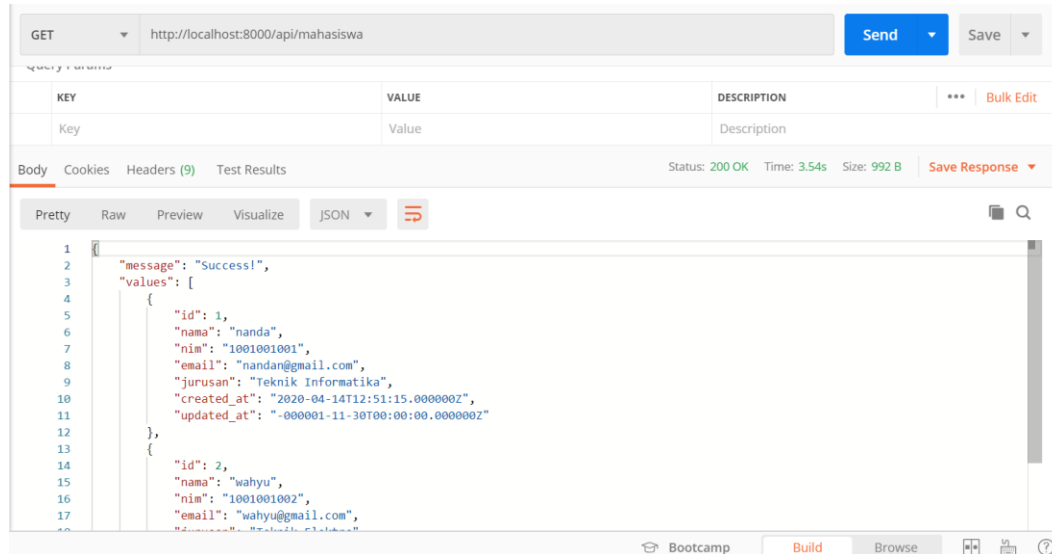
- Tambahkan line 6 agar model Mahasiswa dapat digunakan pada MahasiswaController
- Line 15-19 digunakan untuk memeriksa apakah data > 0 atau data tidak kosong
- Variabel \$res[message] digunakan untuk menampilkan pesan apakah ada data atau tidak ada data di tabel mahasiswa
- Variabel \$array[values] akan menyimpan semua baris data pada tabel mahasiswa

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

```
routes > api.php > ...
1  <?php
2
3  use App\Http\Controllers\MahasiswaController;
4  use Illuminate\Http\Request;
5  use Illuminate\Support\Facades\Route;
6
7  /*
8  |-----
9  | API Routes
10 |-----
11 |
12 | Here is where you can register API routes for your application. These
13 | routes are loaded by the RouteServiceProvider within a group which
14 | is assigned the "api" middleware group. Enjoy building your API!
15 |
16 |*/
17
18 Route::middleware('auth:api')->get('/user', function (Request $request) {
19     return $request->user();
20 });
21
22 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.

```

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

```

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

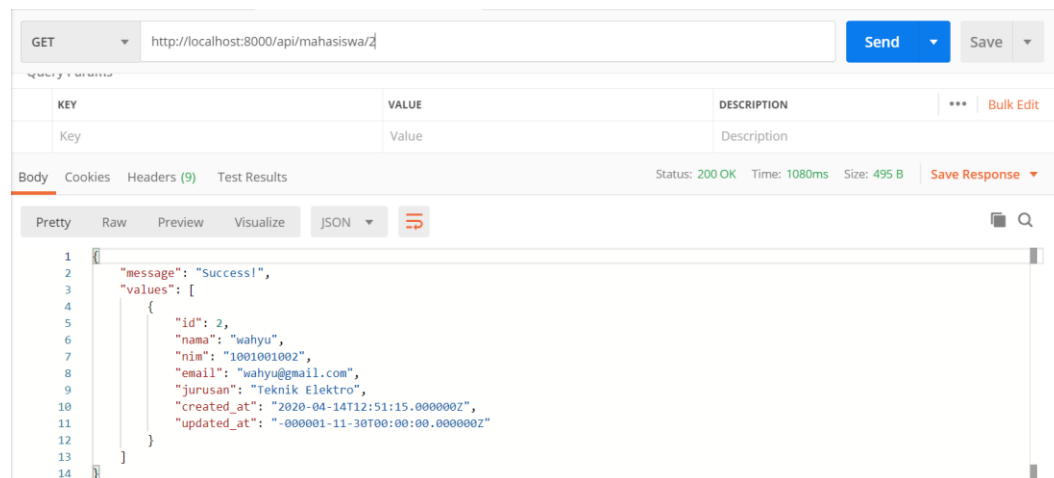
```

Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'

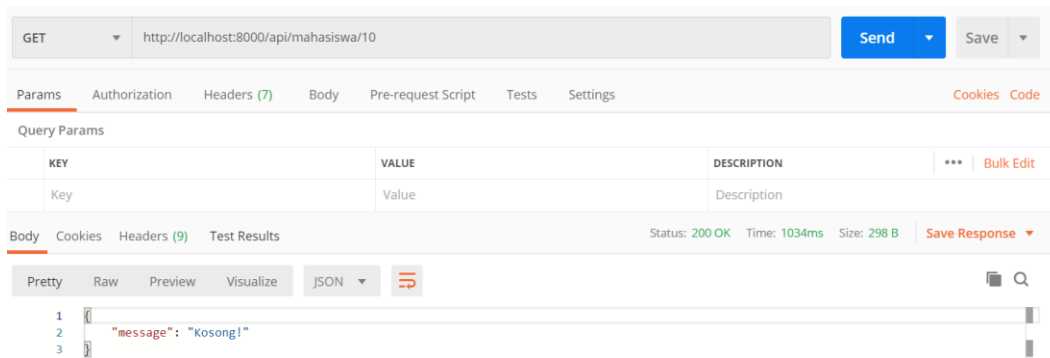
11) Sekarang kita coba untuk menampilkan data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah GET untuk menampilkan data.

Di bawah ini adalah contoh untuk menampilkan data dengan ID=2, maka url diisi :

<http://localhost:8000/api/mahasiswa/2>



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`.

```

44     // Fungsi menambah data
45     public function create(Request $request)
46     {
47         $mhs = new Mahasiswa();
48         $mhs->nama = $request->nama;
49         $mhs->nim = $request->nim;
50         $mhs->email = $request->email;
51         $mhs->jurusan = $request->jurusan;
52
53         // Jika data berhasil tersimpan
54         if($mhs->save()){
55             $res['message'] = "Data berhasil ditambah!";
56             $res['value'] = "$mhs";
57             return response($res);
58         }
59     }

```

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`

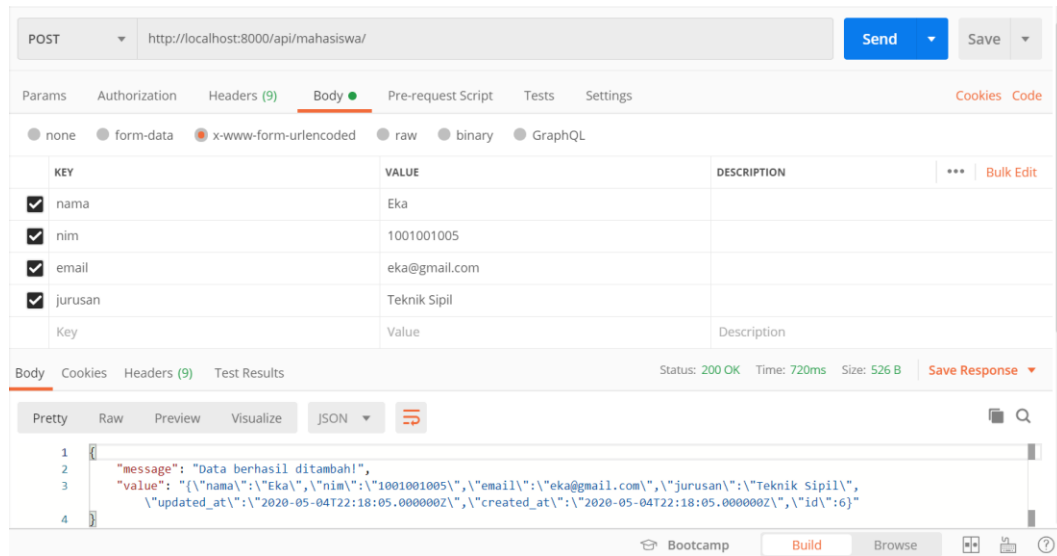
```

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

```

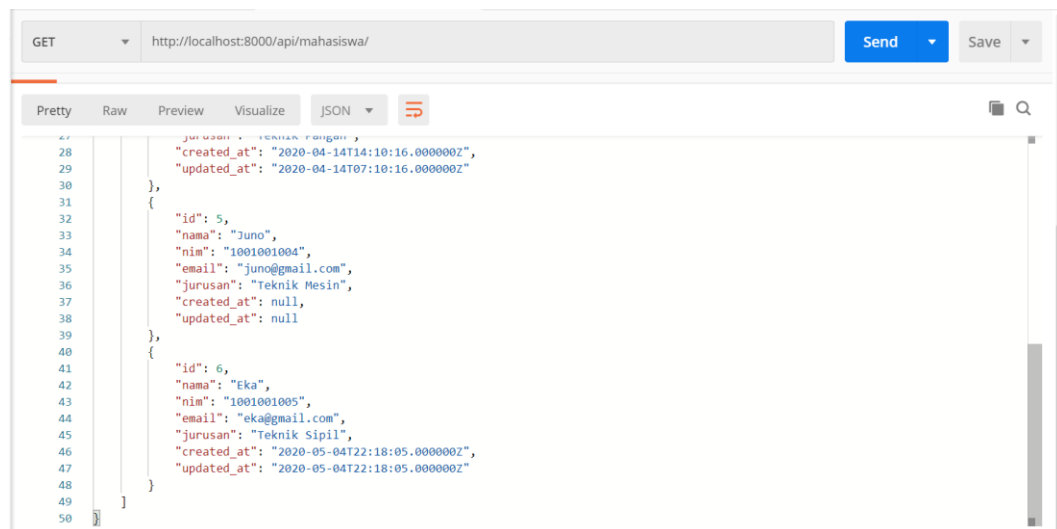
Karena kita ingin menambah data, maka perintah yang dipakai adalah `'post'`.

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.

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

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

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

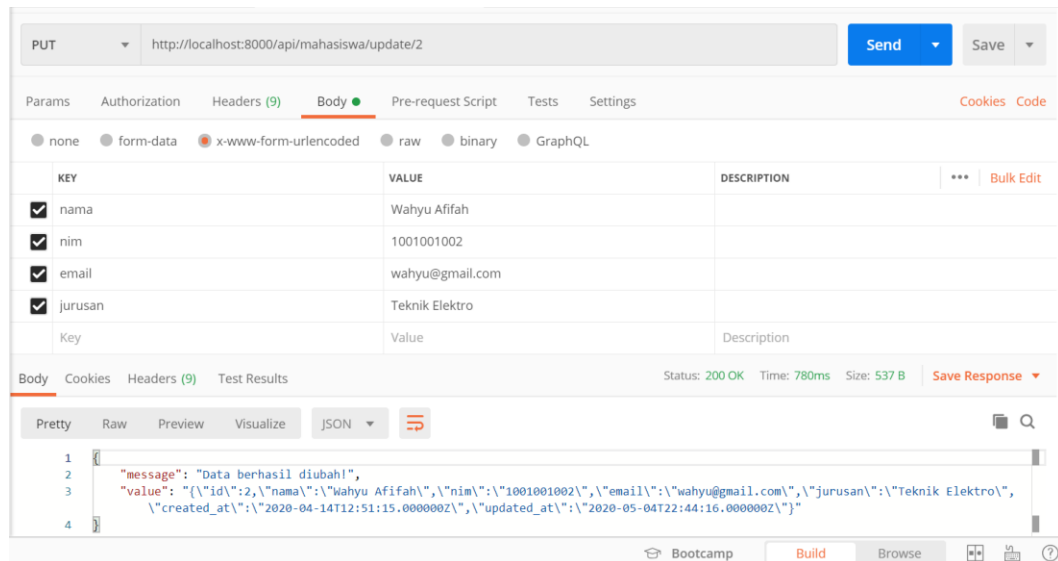
Karena kita ingin memasukkan perubahan data, maka perintah yang dipakai adalah 'put'.

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 :

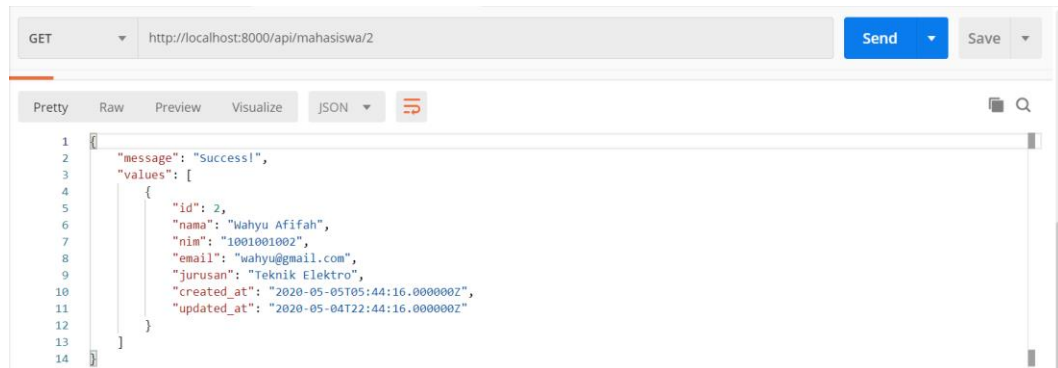
<http://localhost:8000/api/mahasiswa/update/2>. Pilih tab Body dan pilih radio button x-www-

form-urlencoded. Isikan nama kolom pada database pada KEY, untuk isian data yang diubah tuliskan pada VALUE.



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.

```
85 // Fungsi menghapus data
86 public function delete($id)
87 {
88     $mhs = Mahasiswa::where('id', $id);
89
90     if($mhs->delete()){
91         $res['message'] = "Data berhasil dihapus!";
92         return response($res);
93     } else{
94         $res['message'] = "Gagal!";
95         return response($res);
96     }
97 }
```

Keterangan:

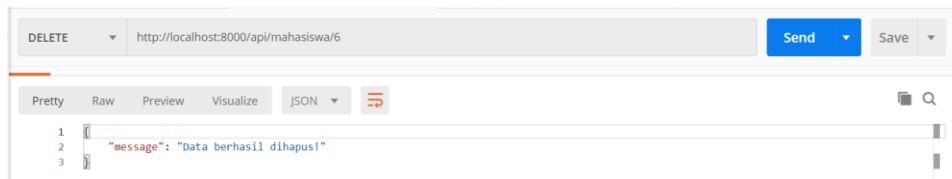
- Fungsi delete menerima parameter id yang menunjukkan ID yang dipilih.
- Line 92-99 : `$mhs->delete()` digunakan untuk menghapus data dari database, apabila `delete()` berhasil dijalankan maka akan ditampilkan pesan berhasil.

19) Tambahkan route untuk memanggil fungsi delete pada routes/api.php

```
26 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.