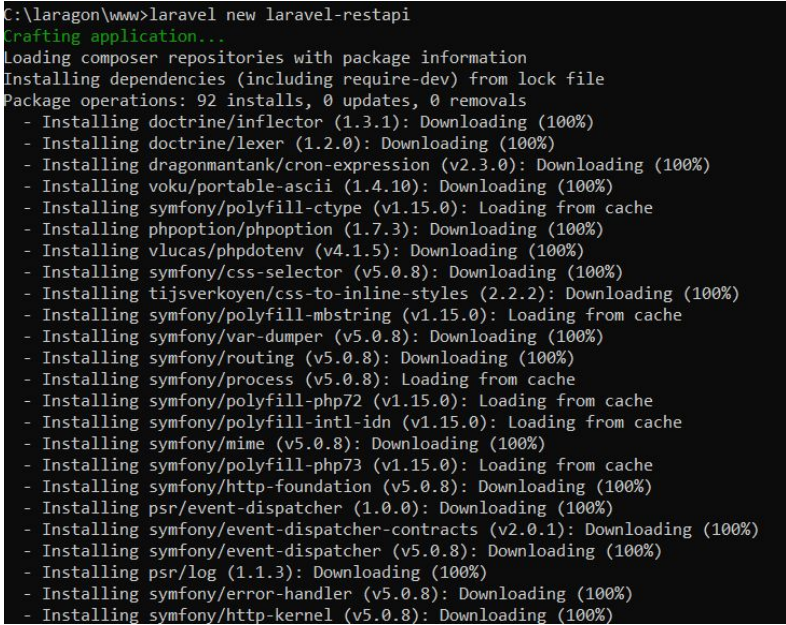
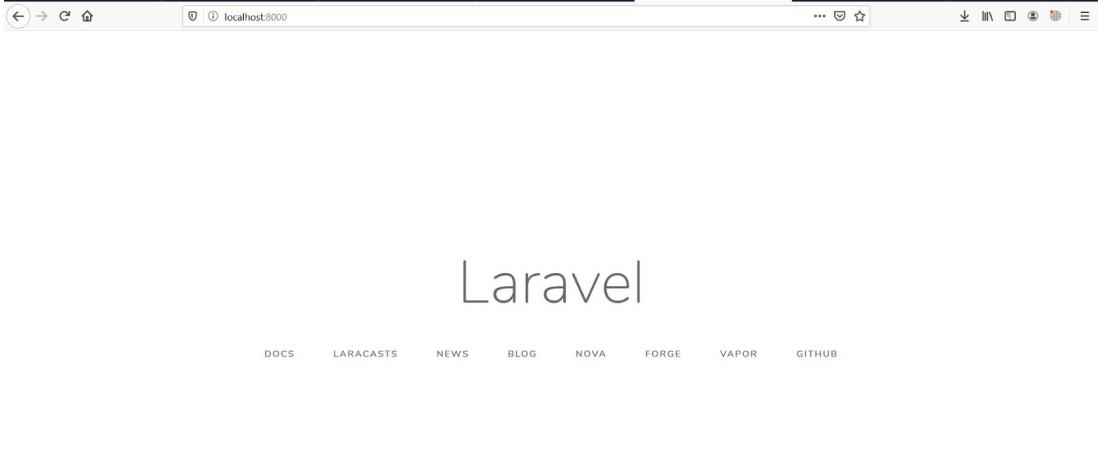


PEMROGRAMAN WEB LANJUT
JOBSHEET 13



Oleh:
Facko Ellanda
1841720154

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2020

No	Keterangan
1	<p>Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut.</p> <pre>cd C:\laragon\www 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> 

- 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
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:+huRZ05X1S1/vrbEevqL5PENIdMKpPnX/Xp0UoYjYfS0=
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=
```

- 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 merupakan perintah untuk menyertakan pembuatan controller

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

C:\laragon\www\laravel-restapi>_
```

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

```
▼ Controllers
  Controller.php
  MahasiswaController.php
  > Middleware
  Kernel.php
  > Providers
  Mahasiswa.php
  User.php
```

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

```
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
```

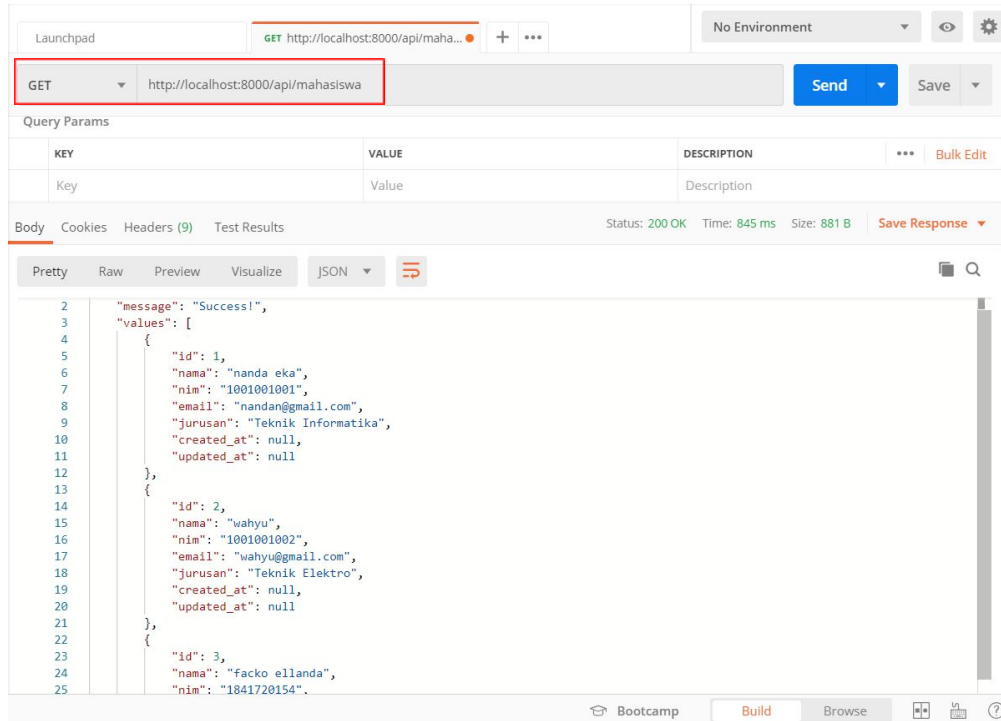
- 6 Modifikasi isi dari MahasiswaController.php untuk dapat mengolah data pada tabel 'mahasiswa'. Pada controller ini, kita akan melakukan operasi untuk menampilkan, menambah, mengubah, dan menghapus data. Pertama, kita akan mengubah fungsi index agar saat fungsi index dipanggil, maka aplikasi akan menampilkan seluruh data dari tabel mahasiswa.

```
app > Http > Controllers > MahasiswaController.php
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         $data = Mahasiswa::all();
12
13         if(count($data) > 0){
14             $res['message'] = "Success!";
15             $res['values'] = $data;
16             return response($res);
17         }
18         else{
19             $res['message'] = "Kosong!";
20             return response($res);
21         }
22     }
23 }
24
```

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

```
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
```

- 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.

```
app > Http > Controllers > MahasiswaController.php

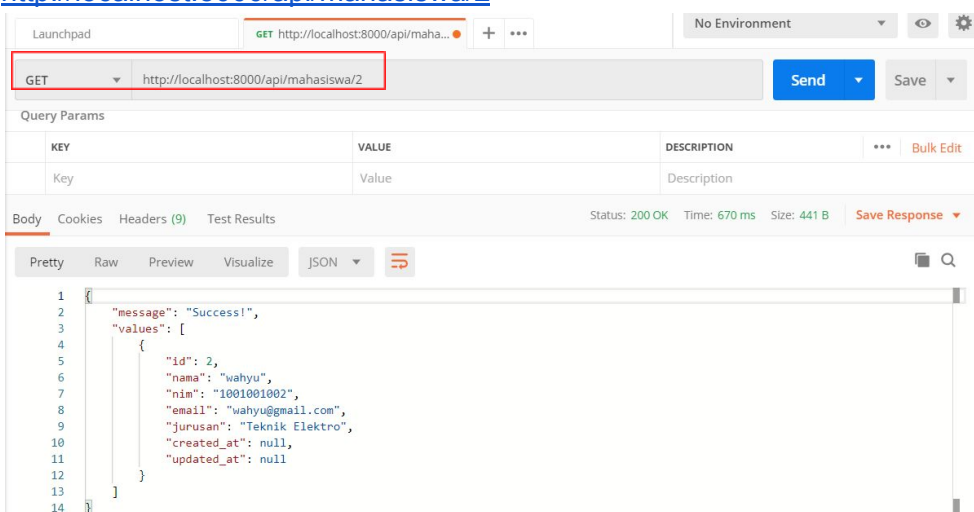
23 public function getId ($id){
24     $data = Mahasiswa::where('id', $id)->get();
25
26     if(count($data) > 0){
27         $res['message'] = "Success!";
28         $res['values'] = $data;
29         return response($res);
30     }
31     else{
32         $res['message'] = "Gagal!";
33         return response($res);
34     }
35 }
36
37
```

- 10 Tambahkan route untuk memanggil fungsi getId pada routes/api.php

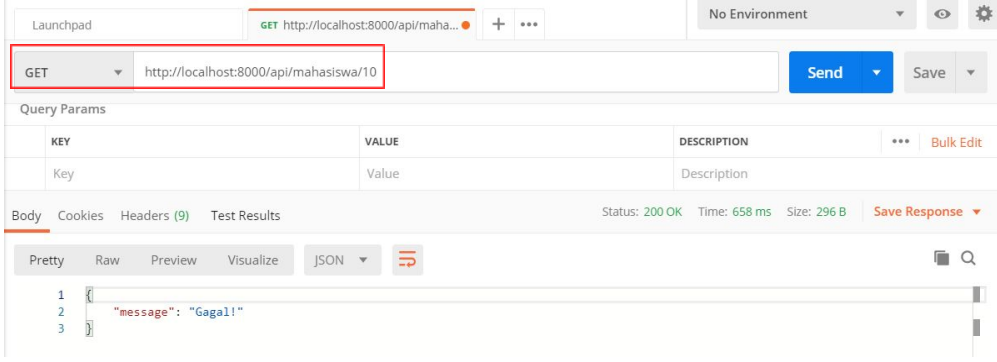
```
23 get('/mahasiswa/{id}', 'MahasiswaController@getId');
```

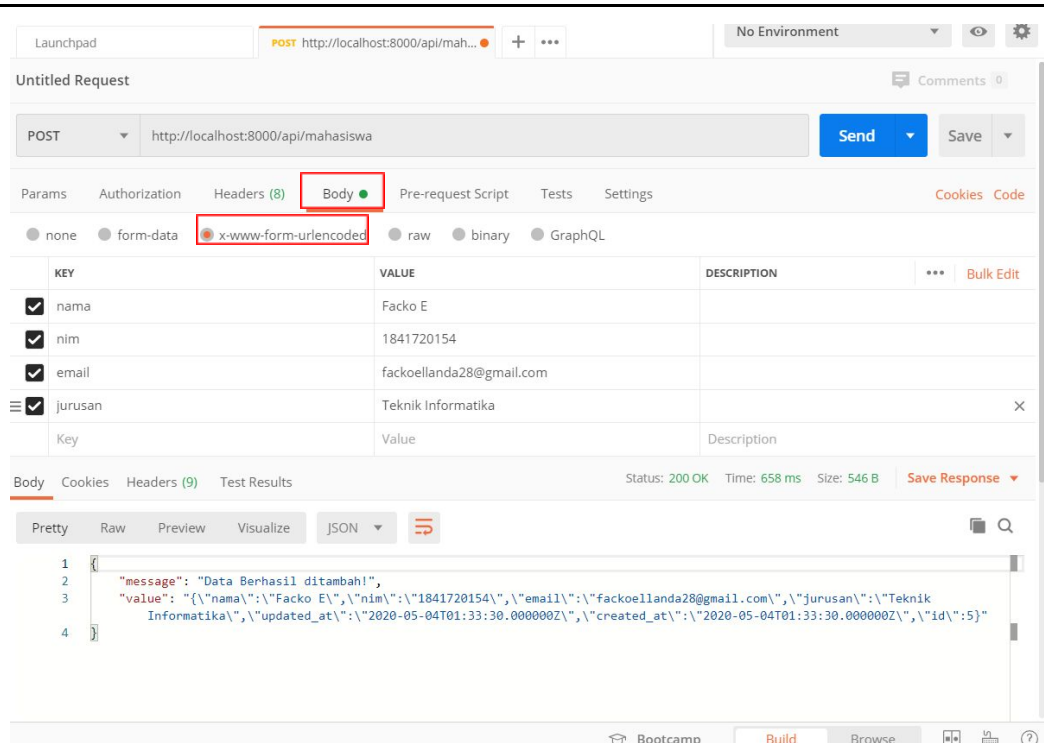
Karena kita akan mengambil data maka menggunakan perintah get

- 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>

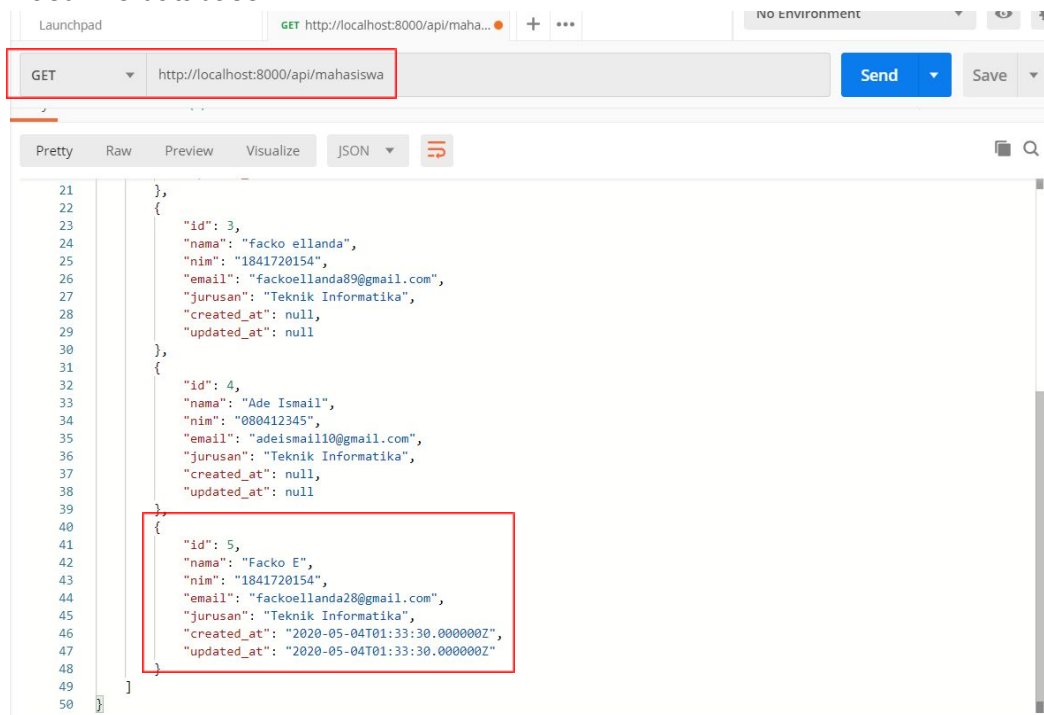


Ketika mencoba menampilkan ID=10 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut.

	
12	<p>Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php.</p> <pre> app > Http > Controllers > 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['value'] = "\$mhs"; 46 return response(\$res); 47 } 48 } 49 } 50 </pre>
13	<p>Tambahkan route untuk memanggil fungsi create pada routes/api.php.</p> <pre> 25 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.

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

- 16 Tambahkan route untuk memanggil fungsi update pada 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.

The screenshot shows the Postman interface for a PUT request. The URL is `http://localhost:8000/api/mahasiswa/update/1`. The request body is set to `x-www-form-urlencoded` and contains the following data:

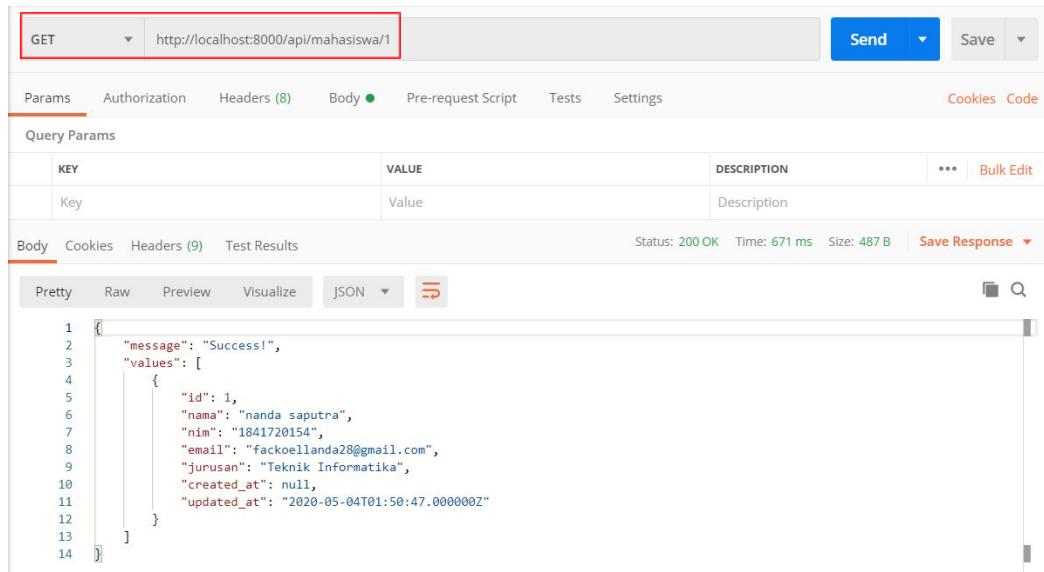
KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> nama	nanda saputra	
<input checked="" type="checkbox"/> nim	1841720154	
<input checked="" type="checkbox"/> email	fackoellanda28@gmail.com	
<input checked="" type="checkbox"/> jurusan	Teknik Informatika	
Key	Value	Description

The response status is 200 OK. The response body is a JSON object:

```
{
  "message": "Data Berhasil diubah!",
  "value": "{\n  \"id\":1,\n  \"nama\": \"nanda saputra\",\n  \"nim\": \"1841720154\",\n  \"email\": \"fackoellanda28@gmail.com\",\n  \"jurusan\": \"Teknik Informatika\",\n  \"created_at\": null,\n  \"updated_at\": \"2020-05-04T01:50:47.000000Z\"}"
}
```

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.

```

app > Http > Controllers > MahasiswaController.php

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

```

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

```

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

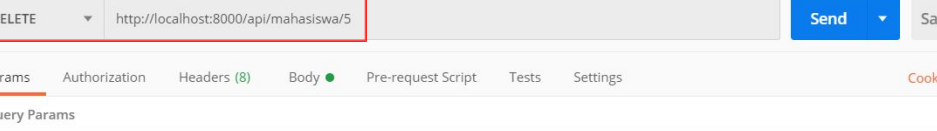
```

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.

Contoh kita akan menghapus data dengan id = 5, sesuai dengan yang ada dalam database, id 5 disini merupakan data yang ditambahkan dalam tahap sebelumnya
http://localhost:8000/api/mahasiswa /5

Muncul



The screenshot displays the REST Client interface. At the top, a red box highlights the DELETE method and the URL `http://localhost:8000/api/mahasiswa/5`. Below this, there are tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings. The Body tab is selected, showing the response in JSON format. The response is `{\"message\": \"Data Berhasil dihapus!\"}`. The interface also includes a Send button, a Save button, and a Cookies/Code section.