

**LAPORAN TUGAS PEMROGRAMAN WEB LANJUT**  
(Restful API Laravel )



**Disusun Oleh :**

**Nama: Ahmad Bahrul Ulum**

**Nim: 1841720150**

**PROGRAM STUDI TEKNIK INFORMATIKA  
JURUSAN TEKNOLOGI INFORMASI  
POLITEKNIK NEGERI MALANG  
2020**

No	Praktikum: Membuat RESTful API di Laravel
1	<p>Buat project baru dengan nama “<b>laravel-restapi</b>”. Buka command prompt, tuliskan perintah berikut.</p> <pre><b>cd C:\xampp\htdocs</b> <b>laravel new laravel-restapi</b></pre>
2	<p>Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut.</p> <pre><b>cd C:\laravel-restapi</b> <b>php artisan serve</b></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> <pre>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= 15</pre>
4	<p>Buat <b>model</b> dengan nama <b>Mahasiswa</b>, buat juga <b>controllem</b>nya. Untuk membuat model dan <i>controllem</i>nya sekaligus tuliskan perintah berikut pada <i>command prompt</i> (terlebih dahulu keluar dari php artisan serve dengan mengetik ctrl+C pada keyboard)</p> <pre><b>php artisan make:model Mahasiswa -c</b></pre>

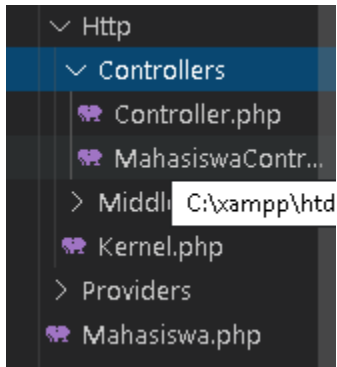
```
C:\xampp\htdocs\laravel-restapi>php artisan make:model Mahasiswa -c
Model created successfully.
Controller created successfully.
```

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
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 }
12
```

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.

```
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  class MahasiswaController extends Controller
8  {
9      //fungsi index digunakan untuk menampilkan semua data mahasiswa
10     public function index(){
11         $data = mahasiswa::all();
12
13         //cek data tidak kosong
14         if(count($data) > 0){
15             $res['message'] = "Success!";
16             $res['values'] = $data;
17             return response($res);
18         }
19         //jika data kosong
20         else{
21             $res['message'] = "Kosong!";
22             return response($res);
23         }
24     }
25 }
26
```

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

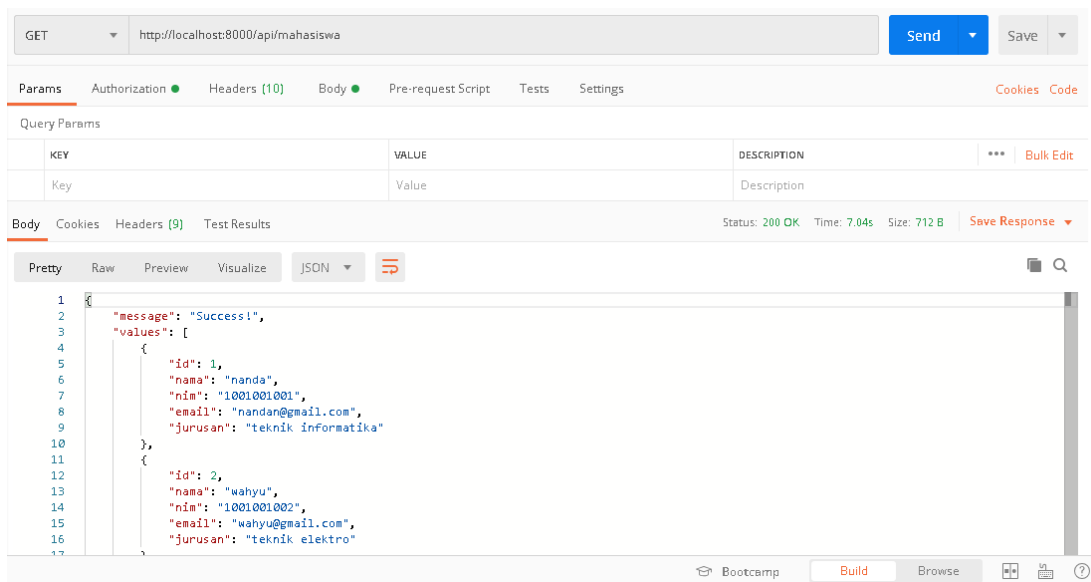
7 Tambahkan route untuk memanggil fungsi index pada file **routes/api.php** (Line 21).

```
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.

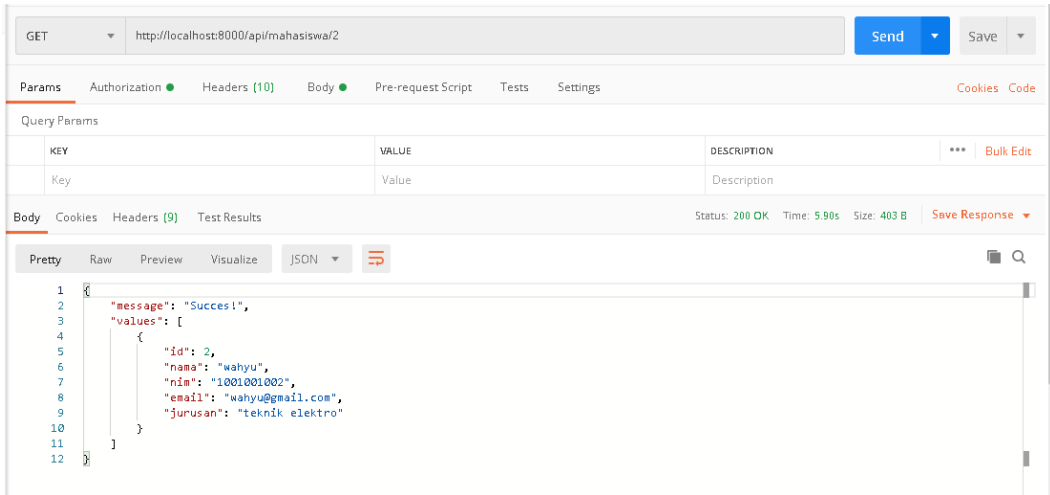
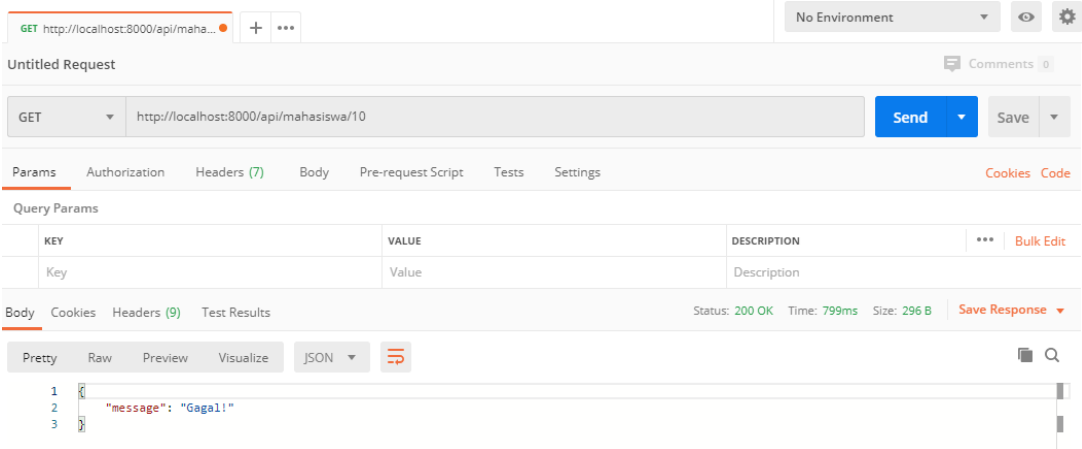
9

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 {
30     $data = Mahasiswa::where('id',$id)->get();
31
32     //cek jika data ditemukan
33     if(count($data) > 0){
34         $res['message'] = "Sukses!";
35         $res['values'] = $data;
36         return response($res);
37     }
38     //jika data tidak ditemukan
39     else{
40         $res['message'] = "Gagal!";
41         return response($res);
42     }
43 }
44
45 }
```

Keterangan:

- Fungsi `getId` menerima parameter `$id` yang menunjukkan ID mahasiswa

	<p>yang dipilih</p> <p>Line 30 merupakan pemanggilan model untuk membaca data berdasarkan ID</p>
10	<p>Tambahkan <i>route</i> untuk memanggil fungsi <code>getId</code> pada <b>routes/api.php</b></p> <pre>Route::get('/mahasiswa/{id}', 'MahasiswaController@getId');</pre> <p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'</p>
11	<p>Sekarang kita coba untuk menampilkan data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah <b>GET</b> untuk menampilkan data.</p> <p>Di bawah ini adalah contoh untuk menampilkan data dengan ID=2, maka url diisi :</p> <p><b><i>http://localhost:8000/api/mahasiswa/2</i></b></p>  <p>Ketika mencoba menampilkan ID=10 akan muncul pesan "Gagal", karena tidak ada data mahasiswa dengan ID tersebut.</p> 

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

```
44
45 //fungsi tambah data
46 public function create(Request $request)
47 {
48     $mhs = new Mahasiswa();
49     $mhs->nama = $request->nama;
50     $mhs->nim = $request->nim;
51     $mhs->email = $request->email;
52     $mhs->jurusan = $request->jurusan;
53
54     //jika data berhasil tersimpan
55     if($mhs->save()){
56         $res['message'] = "Data berhasil ditambah!";
57         $res['value'] = "$mhs";
58         return response($res);
59     }
60 }
61
```

Keterangan:

- Fungsi **create** menerima parameter **Request** yang menampung isian data mahasiswa yang akan ditambahkan ke database.

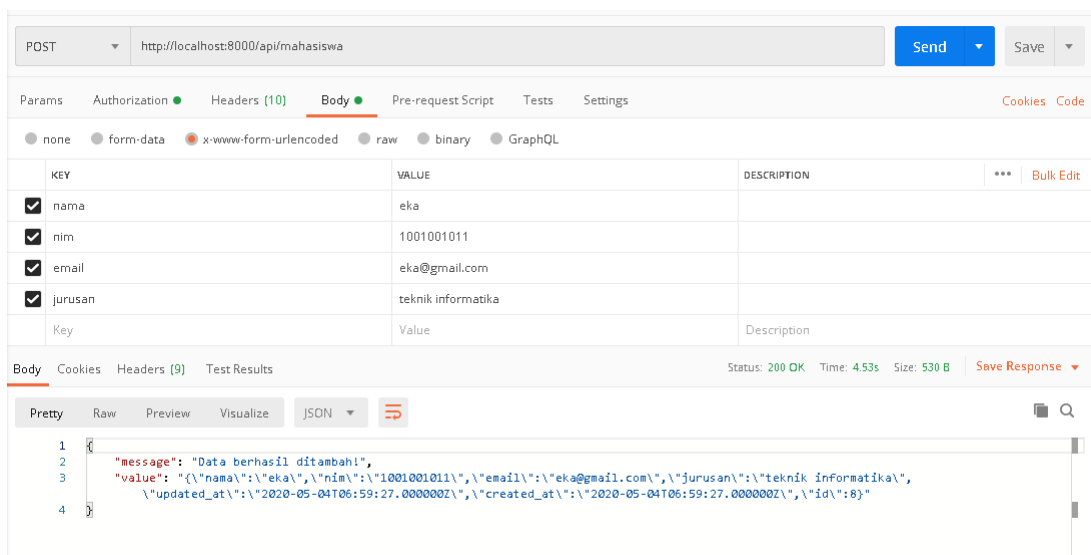
Line 55-59 : `$mhs->save()` digunakan untuk menyimpan data ke database, apabila `save()` berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang ditambah.

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

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

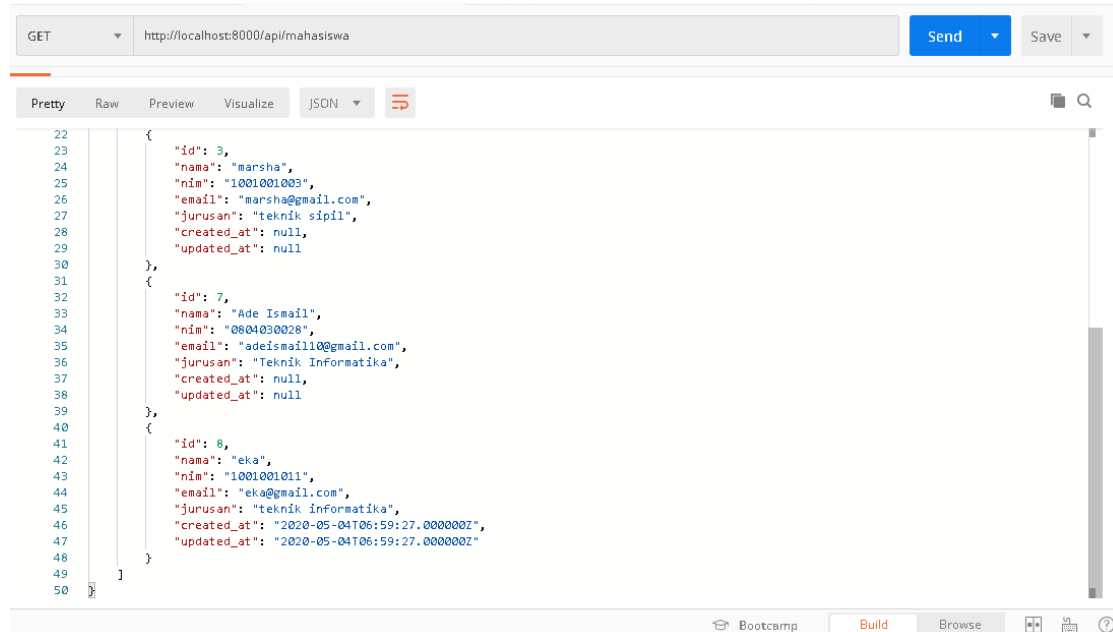
Karena kita ingin menambah data, maka perintah yang dipakai adalah 'post'.

- 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.



```

22  {
23    "id": 3,
24    "nama": "marsha",
25    "nim": "1001001003",
26    "email": "marsha@gmail.com",
27    "jurusan": "teknik sipil",
28    "created_at": null,
29    "updated_at": null
30  },
31  {
32    "id": 7,
33    "nama": "Ade Ismail",
34    "nim": "0804030028",
35    "email": "adeismaill10@gmail.com",
36    "jurusan": "Teknik Informatika",
37    "created_at": null,
38    "updated_at": null
39  },
40  {
41    "id": 8,
42    "nama": "eka",
43    "nim": "1001001011",
44    "email": "eka@gmail.com",
45    "jurusan": "teknik informatika",
46    "created_at": "2020-05-04T06:59:27.000000Z",
47    "updated_at": "2020-05-04T06:59:27.000000Z"
48  }
49  ]
50

```

15

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



	<pre> 62 //fungsi untuk mengubah data 63 public function update(Request \$request, \$id) 64 { 65     \$nama = \$request-&gt;nama; 66     \$nim = \$request-&gt;nim; 67     \$email = \$request-&gt;email; 68     \$jurusan = \$request-&gt;jurusan; 69 70     \$mhs = Mahasiswa::find(\$id); 71     \$mhs-&gt;nama = \$nama; 72     \$mhs-&gt;nim = \$nim; 73     \$mhs-&gt;email = \$email; 74     \$mhs-&gt;jurusan = \$jurusan; 75 76     if(\$mhs-&gt;save()){ 77         \$res['message'] = "Data berhasil diubah!"; 78         \$res['value'] = "\$mhs"; 79         return response(\$res); 80     } 81     else{ 82         \$res['message'] = "Gagal!"; 83         return response(\$res); 84     } 85 } </pre>
	<p>Keterangan:</p> <ul style="list-style-type: none"> <li>• Fungsi <b>update</b> menerima parameter <b>Request</b> yang menampung isian data mahasiswa yang akan diubah dan parameter <b>id</b> yang menunjukkan ID yang dipilih.</li> <li>• Line 70 : Mahasiswa::find(\$id) digunakan untuk pencarian data pada tabel mahasiswa berdasarkan \$id.</li> </ul> <p>Line 76-80 : \$mhs-&gt;save() digunakan untuk menyimpan perubahan data ke database, apabila save() berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang diubah.</p>
16	<p>Tambahkan <i>route</i> untuk memanggil fungsi update pada <b>routes/api.php</b></p> <pre>Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');</pre> <p>Karena kita ingin memasukkan perubahan data, maka perintah yang dipakai adalah 'put'.</p>
17	<p>Sekarang kita coba untuk mengubah data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah <b>PUT</b> untuk mengubah data.</p>

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**.

The screenshot shows a REST client interface with a PUT request to `http://localhost:8000/api/mahasiswa/update/2`. The 'Body' tab is selected, and the 'x-www-form-urlencoded' radio button is chosen. A table lists the data to be updated:

KEY	VALUE	DESCRIPTION
nama	wahyu afifah	
nim	1001001012	
email	wahyua@gmail.com	
jurusan	teknik elektro	

Below the table, the 'Body' tab shows the JSON response:

```
1 {
2   "message": "Data berhasil diubah!",
3   "value": "{\n  \"id\": 2, \"nama\": \"wahyu afifah\", \"nim\": \"1001001012\", \"email\": \"wahyua@gmail.com\", \"jurusan\": \"teknik elektro\",
4     \"created_at\": null, \"updated_at\": \"2020-05-04T07:17:35.000000Z\"}"
```

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*.

The screenshot shows a REST client interface with a GET request to `http://localhost:8000/api/mahasiswa/2`. The 'Body' tab is selected, and the 'JSON' radio button is chosen. The response shows the updated student data:

```
1 {
2   "message": "Sukses!",
3   "values": [
4     {
5       "id": 2,
6       "nama": "wahyu afifah",
7       "nim": "1001001012",
8       "email": "wahyua@gmail.com",
9       "jurusan": "teknik elektro",
10      "created_at": null,
11      "updated_at": "2020-05-04T07:17:35.000000Z"
12    }
13  ]
14 }
```

18

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

```

87      //fungsi untuk menghapus data
88      public function delete($id)
89      {
90          $mhs = Mahasiswa::where('id',$id);
91
92          if($mhs->delete()){
93              $res['message'] = "Data berhasil dihapus!";
94              return response($res);
95          }
96          else{
97              $res['message'] = "Gagal!";
98              return response($res);
99          }
100      }
101

```

- 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**

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

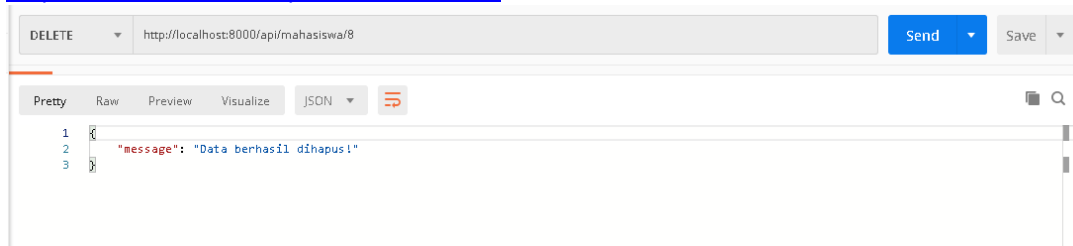
Karena kita ingin menghapus data, maka perintah yang dipakai adalah 'delete'.

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.