

**JOBSHEET 14 : MEMBUAT RESTFUL API
MENGUNAKAN LARAVEL
PEMOGRAMAN WEB LANJUT**

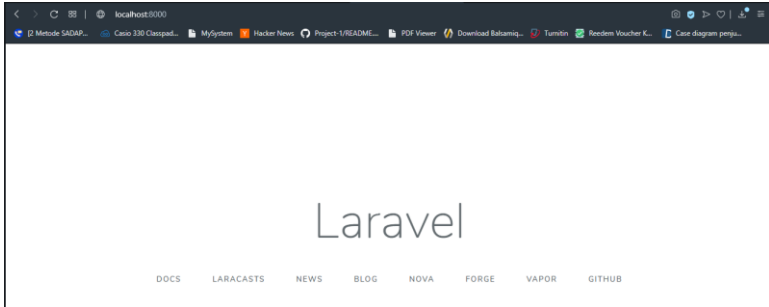
GILANG WAHYU HIDAYAT / 1841720172

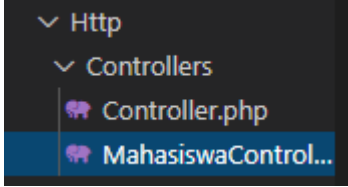
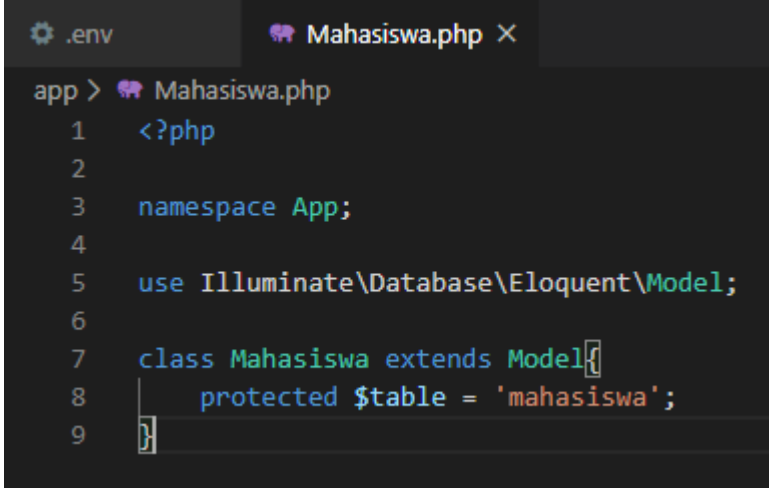


POLITEKNIK NEGERI MALANG

MEI 2020

Praktikum: Membuat RESTful API di Laravel

Langkah	Keterangan
1.	<p>Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut. cd C:\xampp\htdocs laravel new laravel-restapi</p> <pre>PS 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 - Installing voku/portable-ascii (1.4.10): Loading from cache - Installing symfony/polyfill-ctype (v1.15.0): Loading from cache - Installing phpoption/phpoption (1.7.3): Loading from cache - Installing vlucas/phpdotenv (v4.1.5): Downloading (100%) - Installing symfony/css-selector (v5.0.8): Loading from cache</pre>
2.	<p>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.</p> <pre>PS C:\xampp\htdocs\laravel-restapi> php artisan serve Laravel development server started: http://127.0.0.1:8000 0 [Tue May 5 08:10:09 2020] 127.0.0.1:50397 [200]: /favicon.ico</pre> 
3.	<p>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”</p> <pre>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=</pre>

4.	<p>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)</p> <p>php artisan make:model Mahasiswa -c</p> <p>Keterangan : <code>-c</code> merupakan perintah untuk menyertakan pembuatan controller</p> <p>Sehingga pada project laravel-restapi akan bertambah dua file yaitu model Mahasiswa.php serta controller MahasiswaController.php.</p> 
5.	<p>Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.</p>  <pre> app > Mahasiswa.php 1 <?php 2 3 namespace App; 4 5 use Illuminate\Database\Eloquent\Model; 6 7 class Mahasiswa extends Model{ 8 protected \$table = 'mahasiswa'; 9 } </pre>

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.

```

.env  Mahasiswa.php  MahasiswaController.php X
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     //fungsi index digunakan untuk menampilkan semua data mahasiswa
11     public function index()
12     {
13         $data = Mahasiswa::all();
14
15         //cek data tidak kosong
16         if (count($data) > 0) {
17             $res['message'] = "Success!";
18             $res['values'] = $data;
19             return response($res);
20         }
21         //jika datang kosong
22         else {
23             $res['message'] = "kosong!";
24             return response($res);
25         }
26     }

```

7.

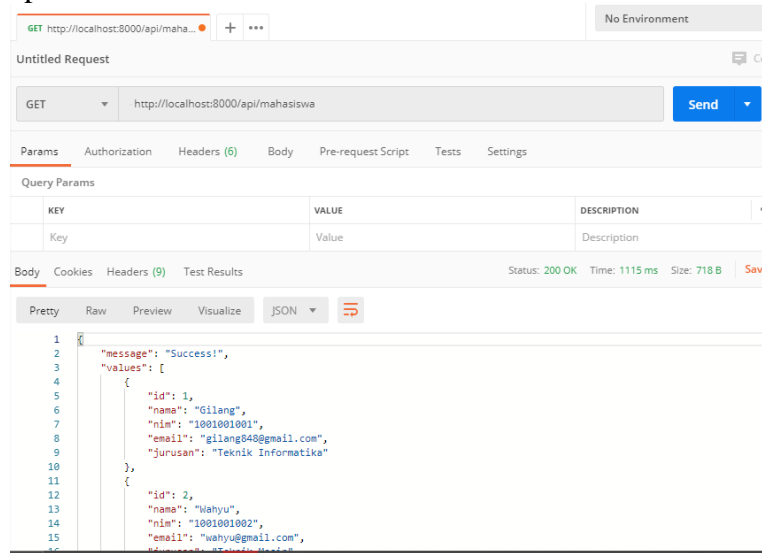
Tambahkan route untuk memanggil fungsi index pada file routes/api.php (Line 21).

```

.env  Mahasiswa.php  MahasiswaController.php  api.php X
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');

```

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.

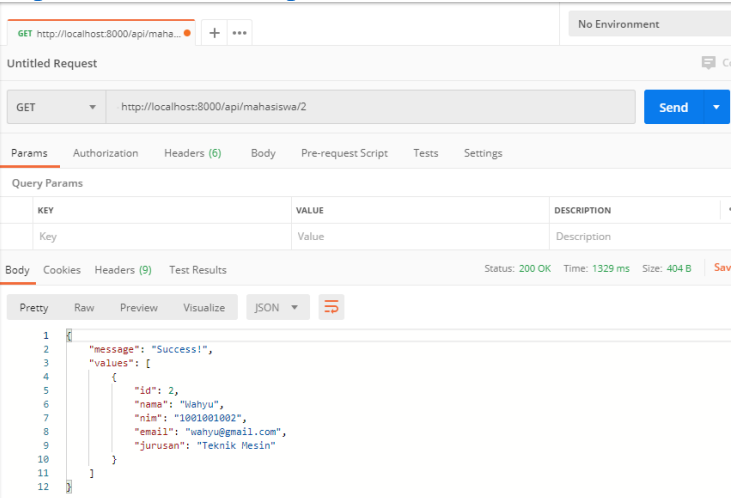
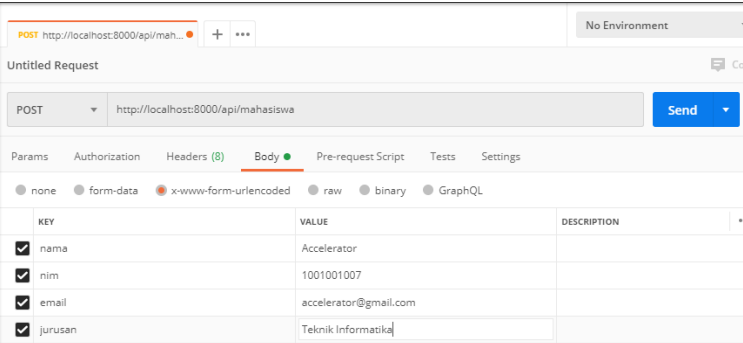


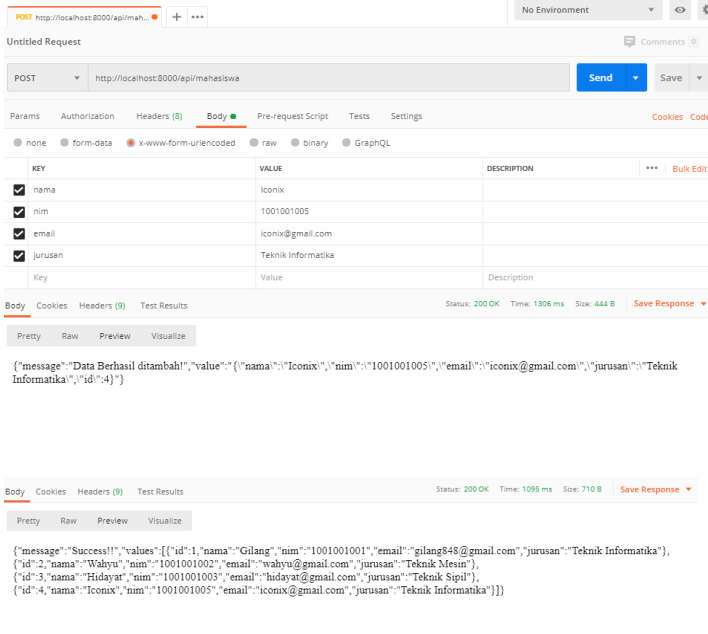
9. Selanjutnya kita akan menambahkan fungsi untuk melihat suatu data ketika dipilih ID tertentu. Buat fungsi baru yaitu `getId` pada `MahasiswaController.php`.

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

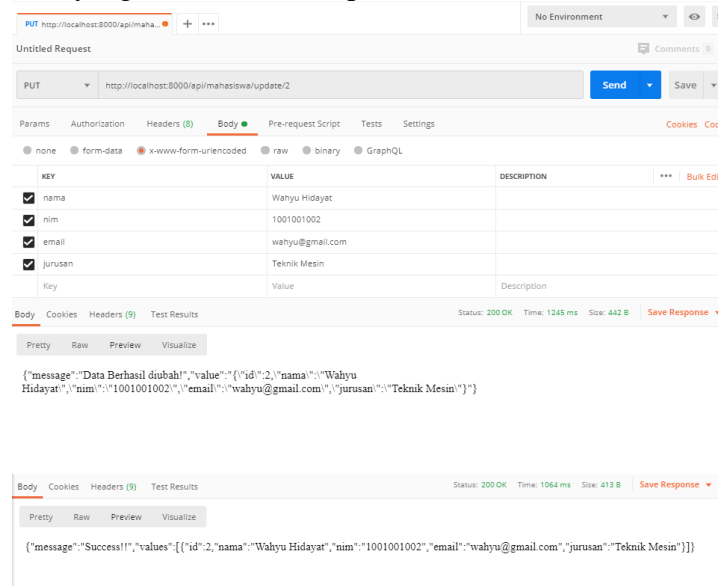
10. Tambahkan route untuk memanggil fungsi `getId` pada `routes/api.php`.

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

11.	<p>Sekarang kita coba untuk menampilkan data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah GET untuk menampilkan data.</p> <p>Di bawah ini adalah contoh untuk menampilkan data dengan ID=2, maka url diisi : http://localhost:8000/api/mahasiswa/2</p> 
12.	<p>Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php.</p> <pre> 46 //fungsi tambah data 47 public function create(Request \$request) 48 { 49 \$mhs = new Mahasiswa(); 50 \$mhs->nama = \$request->nama; 51 \$mhs->nim = \$request->nim; 52 \$mhs->email = \$request->email; 53 \$mhs->jurusan = \$request->jurusan; 54 55 //jika data berhasil tersimpan 56 if (\$mhs->save()) { 57 \$res['message'] = "Data berhasil ditambah!"; 58 \$res['value'] = "\$mhs"; 59 return response(\$res); 60 } 61 } </pre>
13.	<p>Tambahkan route untuk memanggil fungsi create pada routes/api.php.</p> <pre> 25 Route::post('/mahasiswa', 'MahasiswaController@create'); </pre>
14.	<p>Kita coba untuk menambahkan data melalui Postman.</p> 

	
15.	<p>Selanjutnya kita akan menambahkan fungsi untuk mengubah data dari database. Buat fungsi update pada MahasiswaController.php.</p> <pre> 64 public function update(Request \$request, \$id) 65 { 66 \$nama = \$request->nama; 67 \$nim = \$request->nim; 68 \$email = \$request->email; 69 \$jurusan = \$request->jurusan; 70 71 \$mhs = Mahasiswa::find(\$id); 72 \$mhs->nama = \$nama; 73 \$mhs->nim = \$nim; 74 \$mhs->email = \$email; 75 \$mhs->jurusan = \$jurusan; 76 77 if (\$mhs->save()) { 78 \$res['message'] = "Data berhasil diubah!"; 79 \$res['value'] = \$mhs; 80 return response(\$res); 81 } else { 82 \$res['message'] = 'Gagal!'; 83 return response(\$res); 84 } 85 } 86 </pre>
16.	<p>Tambahkan route untuk memanggil fungsi update pada routes/api.php.</p> <pre> 27 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update'); </pre>
17.	<p>Sekarang kita coba untuk mengubah data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah PUT untuk mengubah data.</p> <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</p>

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



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     } else {
96         $res['message'] = "Gagal!";
97         return response($res);
98     }
99 }
100 }
101
```

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

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

20. Buka Postman untuk mencoba menghapus data dari database. Gunakan perintah DELETE untuk mengubah data.

Berikut adalah contoh untuk menghapus data dengan ID=4, maka url diisi : <http://localhost:8000/api/mahasiswa/4>

DEL http://localhost:8000/api/maha...

No Environment

Untitled Request

DELETE

http://localhost:8000/api/mahasiswa/4

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body

Cookies

Headers (9)

Test Results

Status: 200 OK Time: 1213 ms Size: 313 B Save

Pretty

Raw

Preview

Visualize

JSON

1

2

3

"message": "Data berhasil dihapus!"