

JOBSHEET 14
PRAKTIKUM WEB LANJUT REST-API



Oleh:
AGSAL FAIRROHMAD A.P/TI-2A
1841720208

Jurusan Teknologi Informasi
Program Studi Teknik Informatika
Politeknik Negeri Malang
2020

Topik

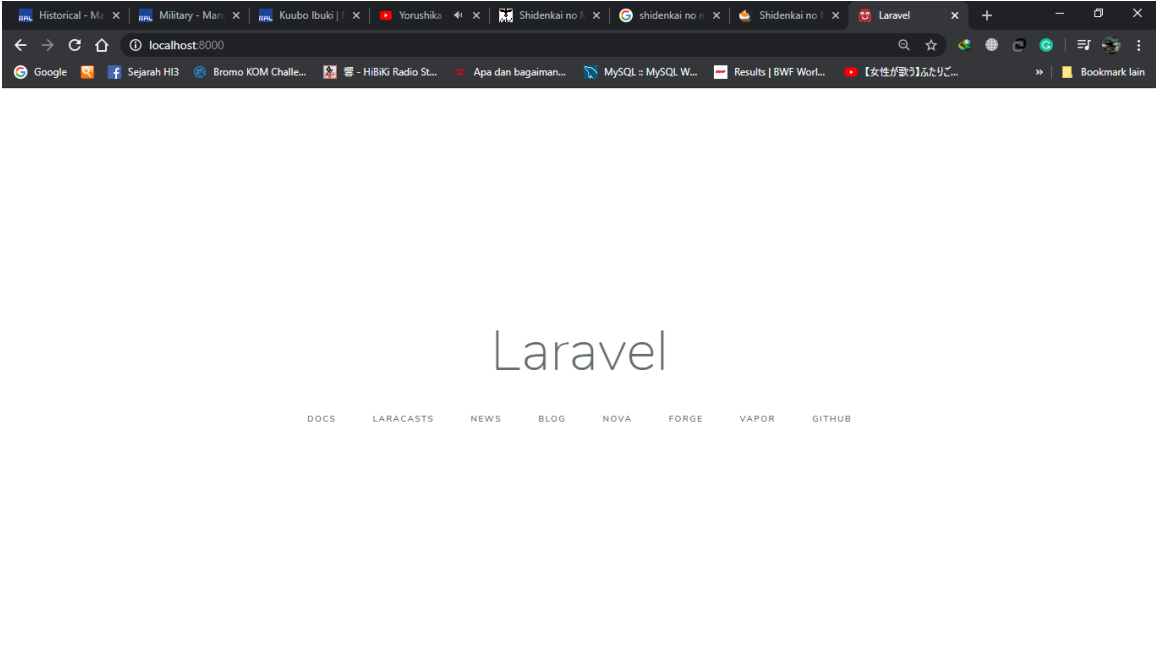
Membuat RESTful API dengan Framework Laravel

Tujuan

Mahasiswa diharapkan dapat:

1. Memahami bagaimana cara membuat RESTful API menggunakan Laravel

Praktikum: Membuat RESTful API di Laravel

Langkah	Keterangan
1	<p>Buat project baru dengan nama “laravel-restapi”. Buka command prompt, tuliskan perintah berikut.</p> <pre>cd C:\xampp\htdocs 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> <p>Akan tampil halaman default Laravel seperti di bawah ini.</p> 
3	<p>Kemudian lakukan konfigurasi <i>database</i> 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>

File Edit Selection View Go Runenv - Untitled (Workspace) - Visua... — □ ×

EXPLORER

OPEN EDITO... 1 UNSAVED

- .env laravel-restapi

UNTITLED (WORKSPACE)

- > siparajack
- > Naive Bayes
- > laravel-crud
- ▼ laravel-restapi
 - > app
 - > bootstrap
 - > config
 - > database
 - > public
 - > resources
 - > routes
 - > storage
 - > tests
 - > vendor
 - .editorconfig
 - .env
 - .env.example
 - .gitattributes
 - .gitignore
 - ! .styleci.yml
 - artisan
 - { } composer.json
 - { } composer.lock
 - { } package-lock.json
 - { } package.json
- > OUTLINE
- > TIMELINE
- > NPM SCRIPTS

.env

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:0YH/gbZZFE+RBZB4wt3o3/u5S1Gv
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=
15
16 BROADCAST_DRIVER=log
17 CACHE_DRIVER=file
18 QUEUE_CONNECTION=sync
19 SESSION_DRIVER=file
20 SESSION_LIFETIME=120
21
22 REDIS_HOST=127.0.0.1
23 REDIS_PASSWORD=null
24 REDIS_PORT=6379
25
```

TERMINAL ... 1: php

```
PS C:\xampp\htdocs> cd laravel-restapi
PS C:\xampp\htdocs\laravel-restapi> php artisan serve
Laravel development server started: http://127.0.0.1:8000
[Tue May 5 19:11:00 2020] 127.0.0.1:49497 [200]: /favicon.ico
[Tue May 5 19:14:28 2020] 127.0.0.1:49611 [200]: /favicon.ico
```

^ [] [] IND 19.14

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

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**.

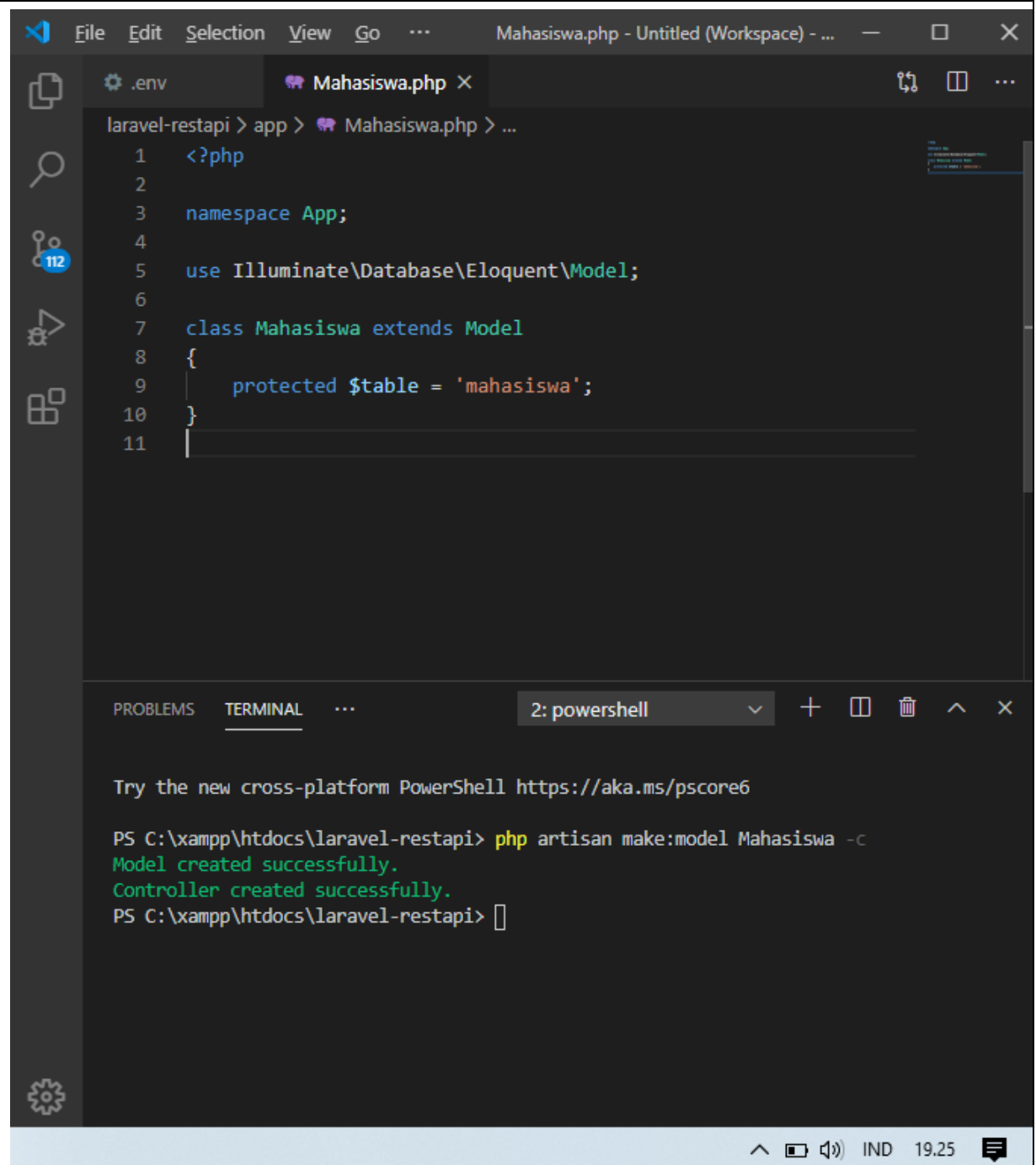
The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER:** The file explorer on the left shows the project structure. The 'Controllers' folder is expanded, showing 'Controller.php' and 'MahasiswaController.php'.
- .env:** The environment file is open in the editor, showing configuration for the application, including database settings.
- TERMINAL:** The terminal at the bottom shows the command `php artisan make:model Mahasiswa -c` being executed, with the output:


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

5

Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.



The screenshot shows a Visual Studio Code editor window with the file 'Mahasiswa.php' open. The code in the editor is as follows:

```
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
```

Below the editor, a terminal window is open with the title '2: powershell'. It displays the following text:

```
Try the new cross-platform PowerShell https://aka.ms/pscore6

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

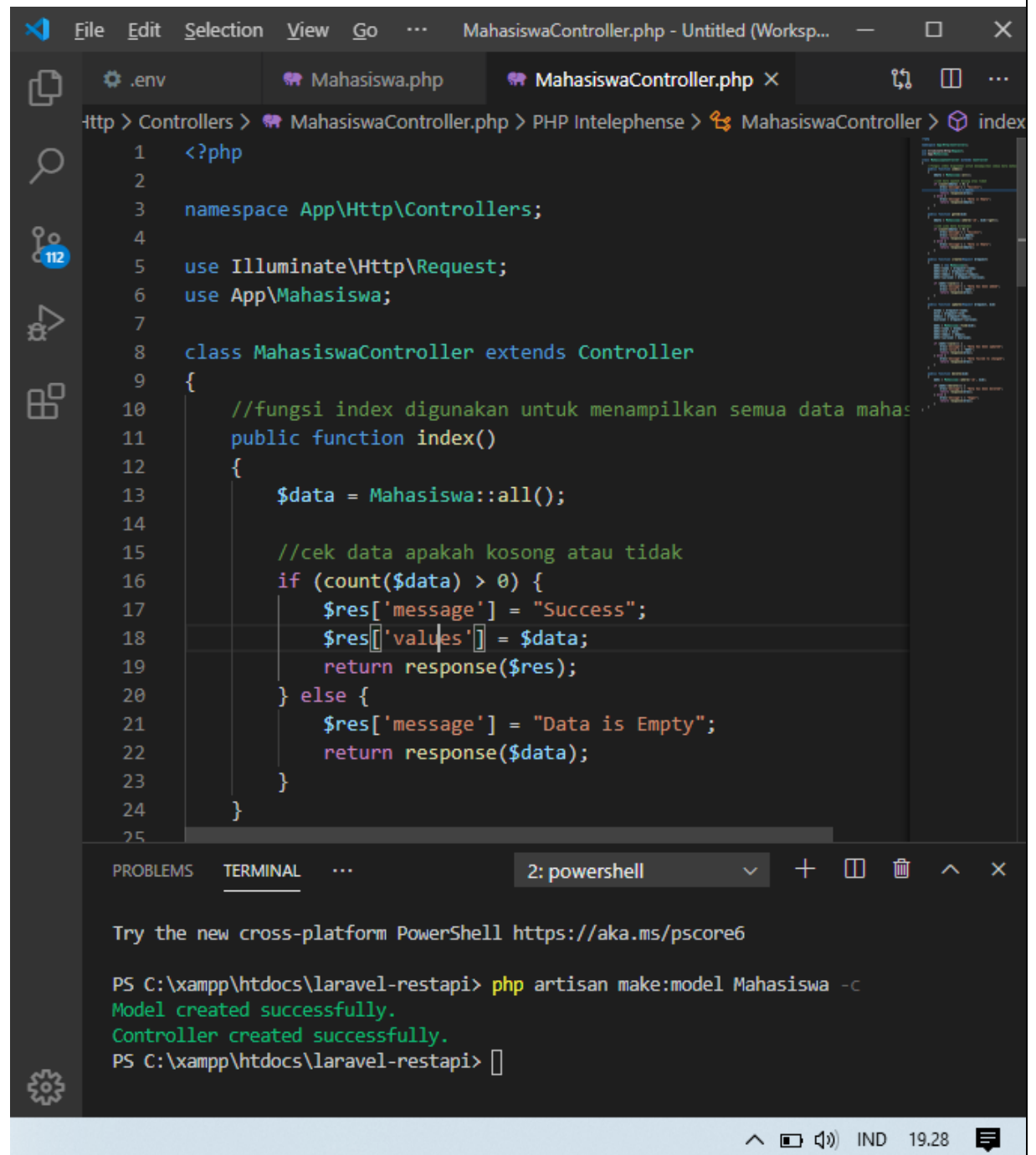
Keterangan:

- 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.



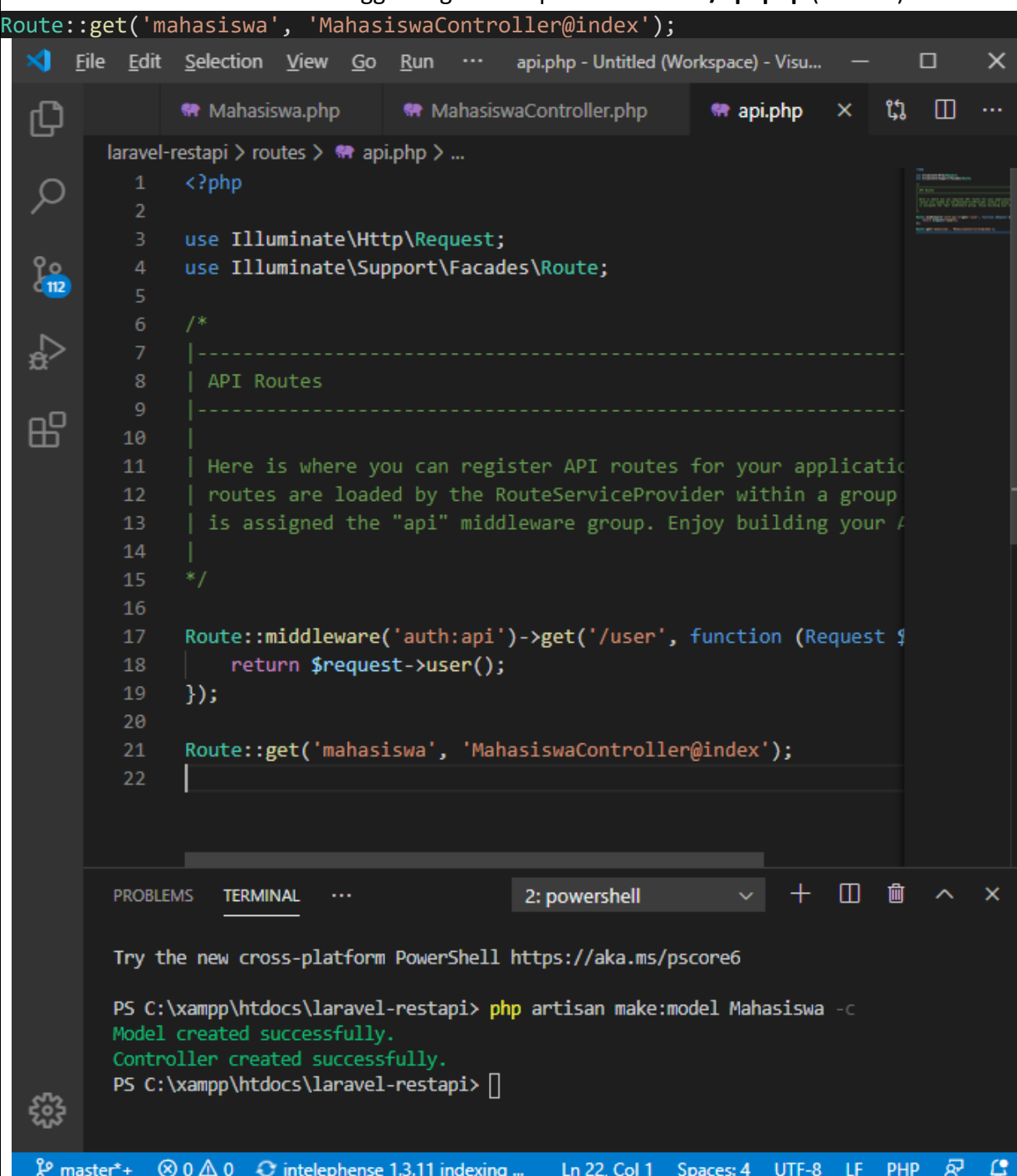
```
File Edit Selection View Go ... MahasiswaController.php - Untitled (Worksp...
.env Mahasiswa.php MahasiswaController.php x
http > Controllers > MahasiswaController.php > PHP Intelephense > MahasiswaController > index
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Mahasiswa;
7
8 class MahasiswaController extends Controller
9 {
10     //fungsi index digunakan untuk menampilkan semua data mahasiswa
11     public function index()
12     {
13         $data = Mahasiswa::all();
14
15         //cek data apakah kosong atau tidak
16         if (count($data) > 0) {
17             $res['message'] = "Success";
18             $res['values'] = $data;
19             return response($res);
20         } else {
21             $res['message'] = "Data is Empty";
22             return response($data);
23         }
24     }
25 }
```

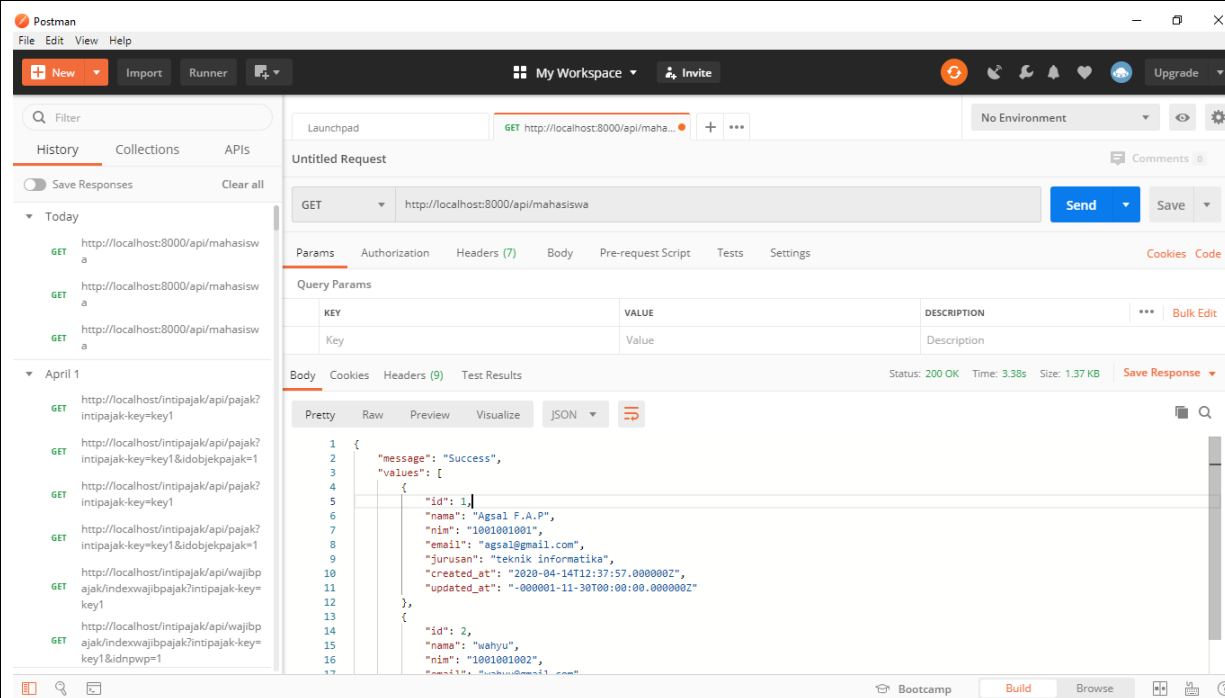
PROBLEMS TERMINAL ... 2: powershell

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

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

^ [] [] IND 19.28

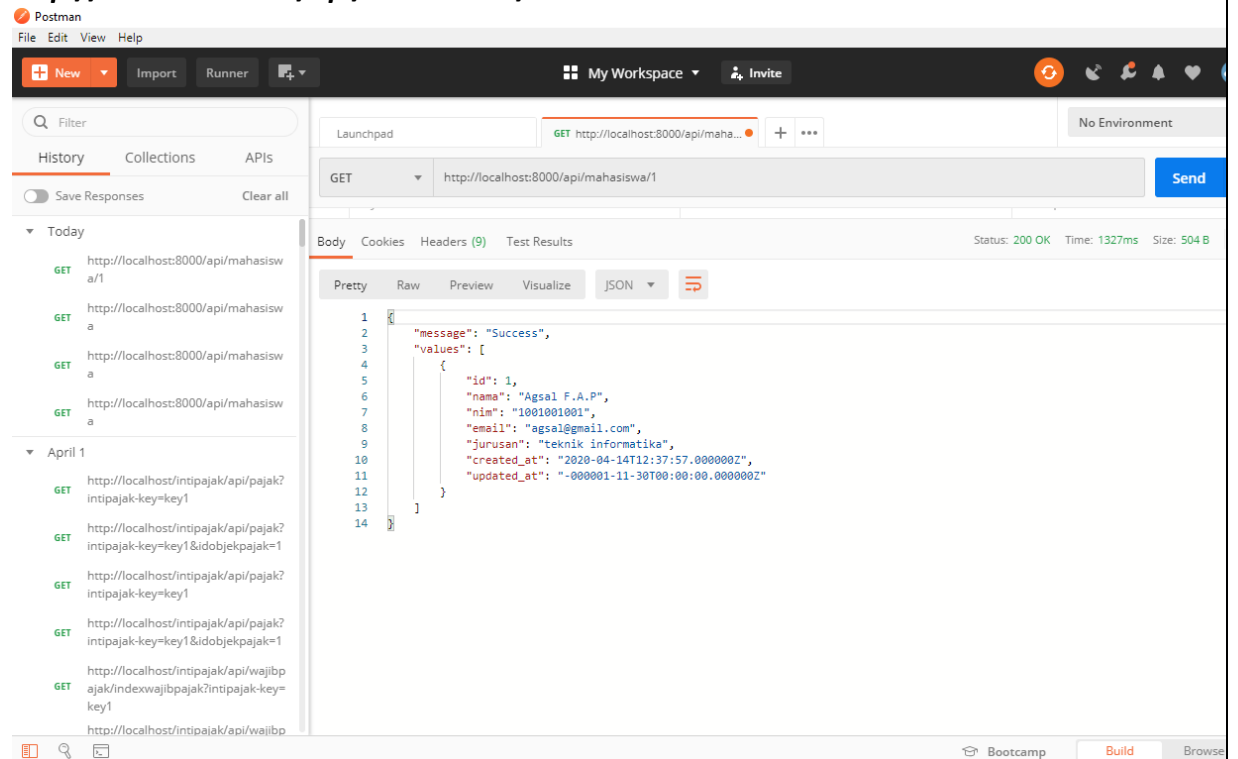
	<p>Keterangan:</p> <ul style="list-style-type: none"> • Tambahkan line 6 agar model Mahasiswa dapat digunakan pada MahasiswaController • Line 15-19 digunakan untuk memeriksa apakah data > 0 atau data tidak kosong • Variabel \$res[message] digunakan untuk menampilkan pesan apakah ada data atau tidak ada data di tabel mahasiswa • Variabel \$array[values] akan menyimpan semua baris data pada tabel mahasiswa
7	<p>Tambahkan route untuk memanggil fungsi index pada file routes/api.php (Line 21).</p>  <pre>Route::get('mahasiswa', 'MahasiswaController@index');</pre> <pre> 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');</pre> <pre> PS C:\xampp\htdocs\laravel-restapi> php artisan make:model Mahasiswa -c Model created successfully. Controller created successfully. PS C:\xampp\htdocs\laravel-restapi> </pre>
	<p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'.</p>
8	<p>Ketikkan perintah php artisan serve pada <i>command prompt</i>. 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.</p>

	 <p>Semua data pada tabel mahasiswa akan tampil, ditampilkan juga pesan sukses.</p>
9	<p>Selanjutnya kita akan menambahkan fungsi untuk melihat suatu data ketika dipilih ID tertentu. Buat fungsi baru yaitu getId pada MahasiswaController.php.</p> <pre> public function getId(\$id) { \$data = Mahasiswa::where('id', \$id)->get(); //cek jika data ditemukan if (count(\$data) > 0) { \$res['message'] = "Success"; \$res['values'] = \$data; return response(\$res); } else { \$res['message'] = "Data is Empty"; return response(\$res); } } </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> • Fungsi getId menerima parameter \$id yang menunjukkan ID mahasiswa yang dipilih • Line 30 merupakan pemanggilan model untuk membaca data berdasarkan ID
10	<p>Tambahkan <i>route</i> untuk memanggil fungsi getId pada routes/api.php</p> <pre> Route::get('/mahasiswa/{id}', 'MahasiswaController@getId'); </pre>
	<p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'</p>

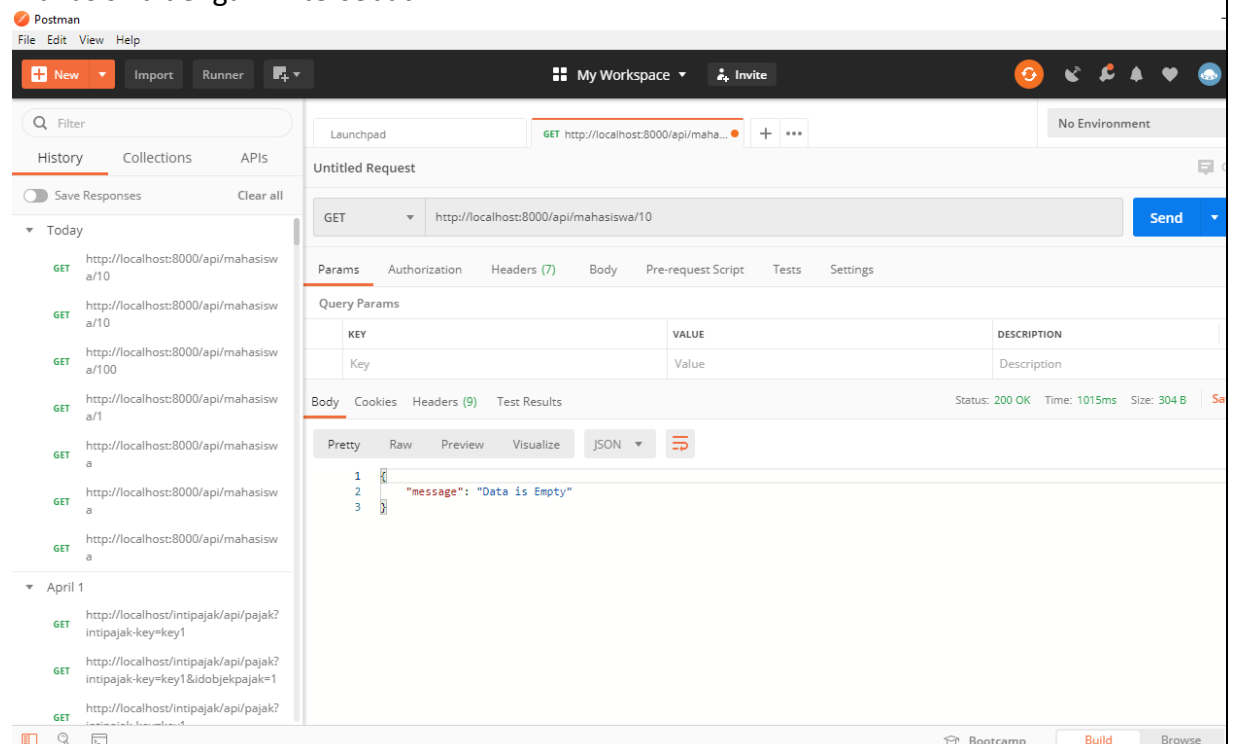
11

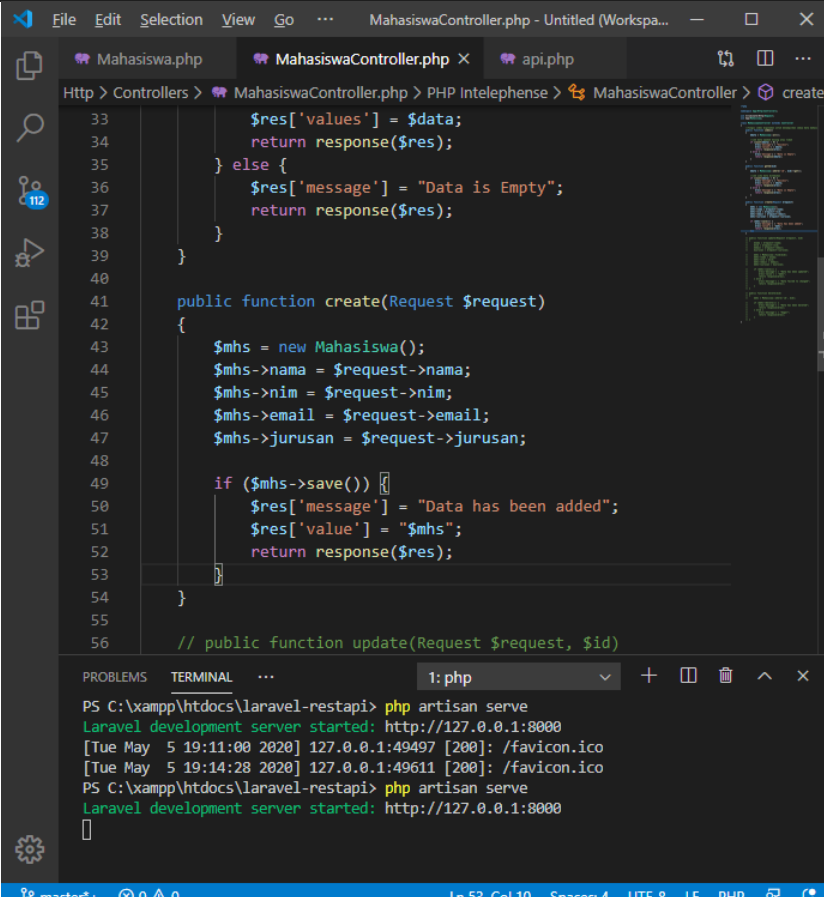
Sekarang kita coba untuk menampilkan data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah **GET** untuk menampilkan data.

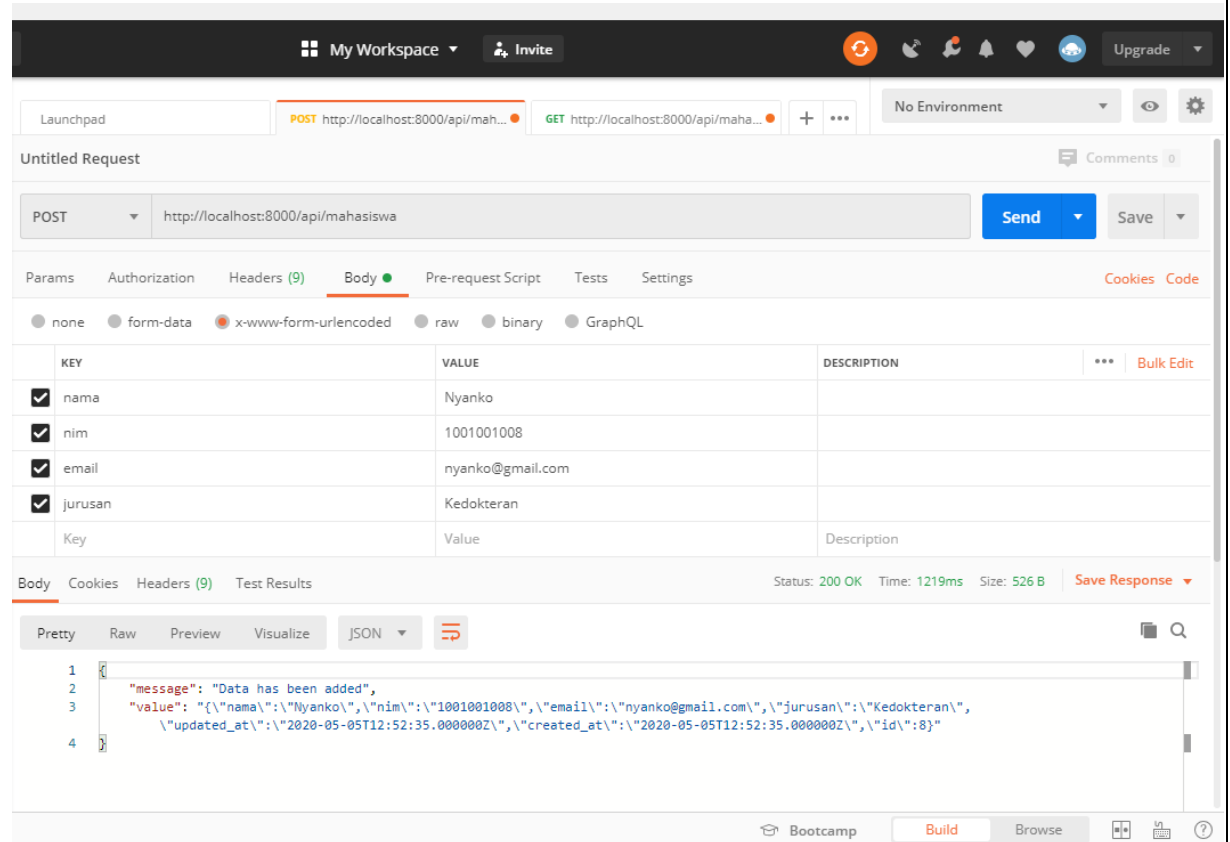
Di bawah ini adalah contoh untuk menampilkan data dengan ID=1, maka url diisi :
http://localhost:8000/api/mahasiswa/1



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

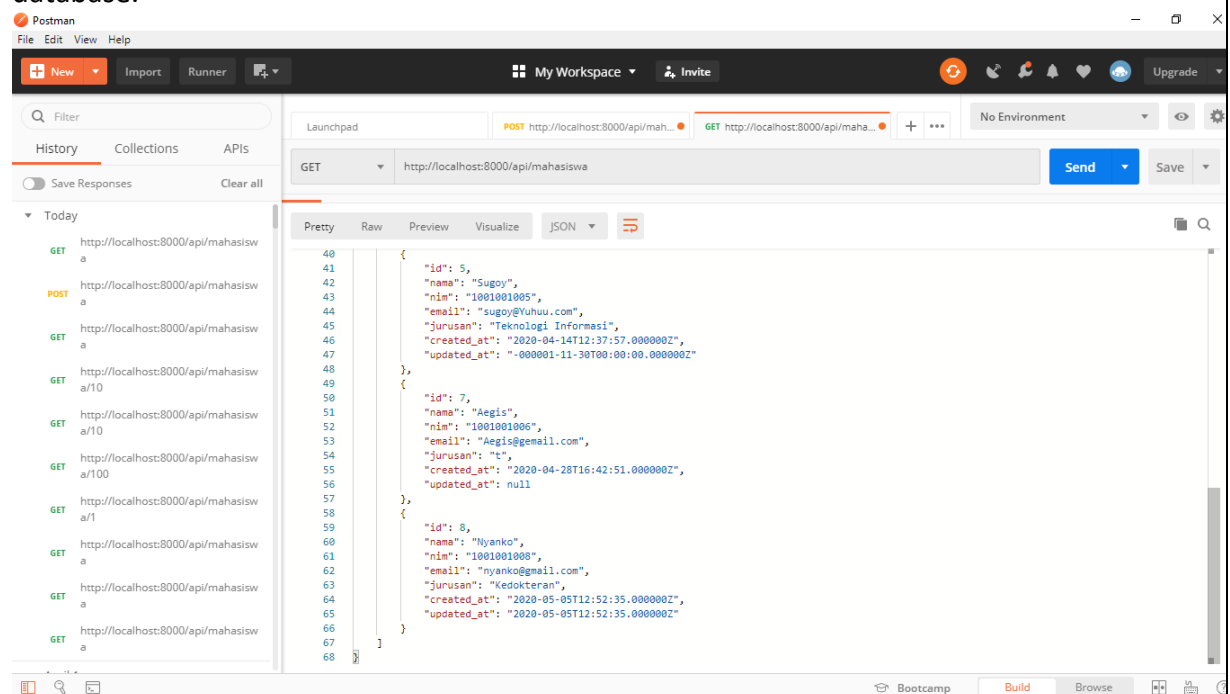


12	<p>Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama create pada MahasiswaController.php.</p>
	<pre> public function create(Request \$request) { \$mhs = new Mahasiswa(); \$mhs->nama = \$request->nama; \$mhs->nim = \$request->nim; \$mhs->email = \$request->email; \$mhs->jurusan = \$request->jurusan; if (\$mhs->save()) { \$res['message'] = "Data has been added"; \$res['value'] = "\$mhs"; return response(\$res); } } </pre>  <p>Keterangan:</p> <ul style="list-style-type: none"> • Fungsi create menerima parameter Request yang menampung isian data mahasiswa yang akan ditambahkan ke database. • Line 55-59 : <code>\$mhs->save()</code> digunakan untuk menyimpan data ke database, apabila <code>save()</code> berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang ditambah.
13	<p>Tambahkan <i>route</i> untuk memanggil fungsi <code>create</code> pada routes/api.php</p> <pre>Route::post('/mahasiswa', 'MahasiswaController@Create');</pre> <p>Karena kita ingin menambah data, maka perintah yang dipakai adalah 'post'.</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.



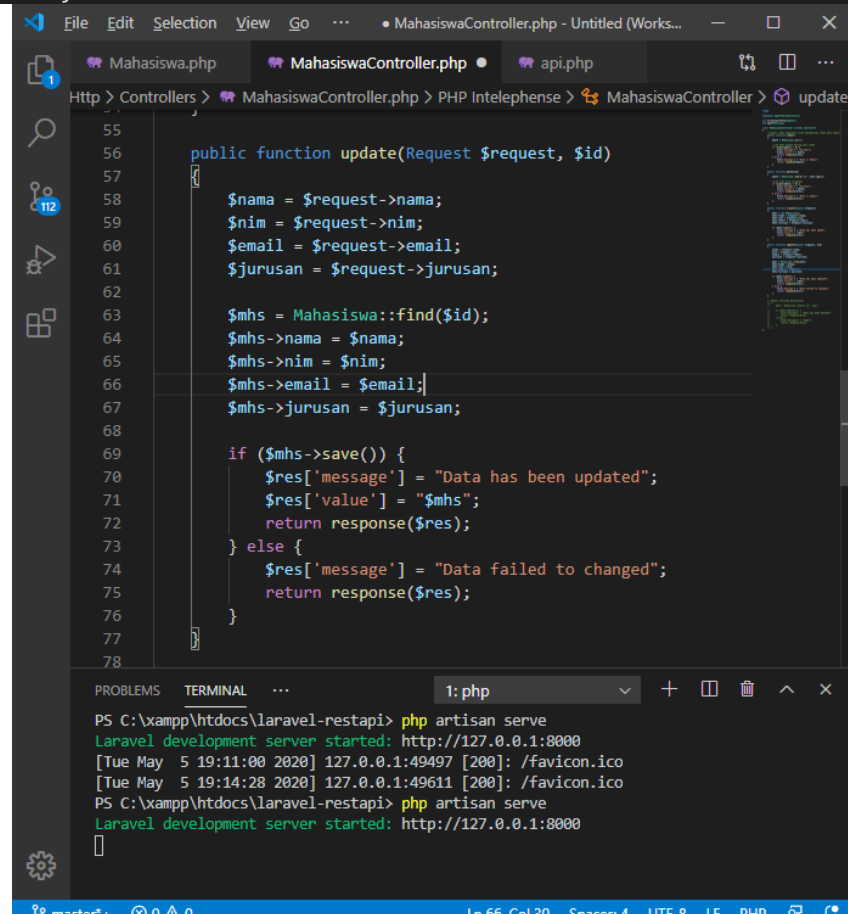
15

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

```
public function update(Request $request, $id)
{
    $nama = $request->nama;
    $nim = $request->nim;
    $email = $request->email;
    $jurusan = $request->jurusan;

    $mhs = Mahasiswa::find($id);
    $mhs->nama = $nama;
    $mhs->nim = $nim;
    $mhs->email = $email;
    $mhs->jurusan = $jurusan;

    if ($mhs->save()) {
        $res['message'] = "Data has been updated";
        $res['value'] = "$mhs";
        return response($res);
    } else {
        $res['message'] = "Data failed to changed";
        return response($res);
    }
}
```



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**

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

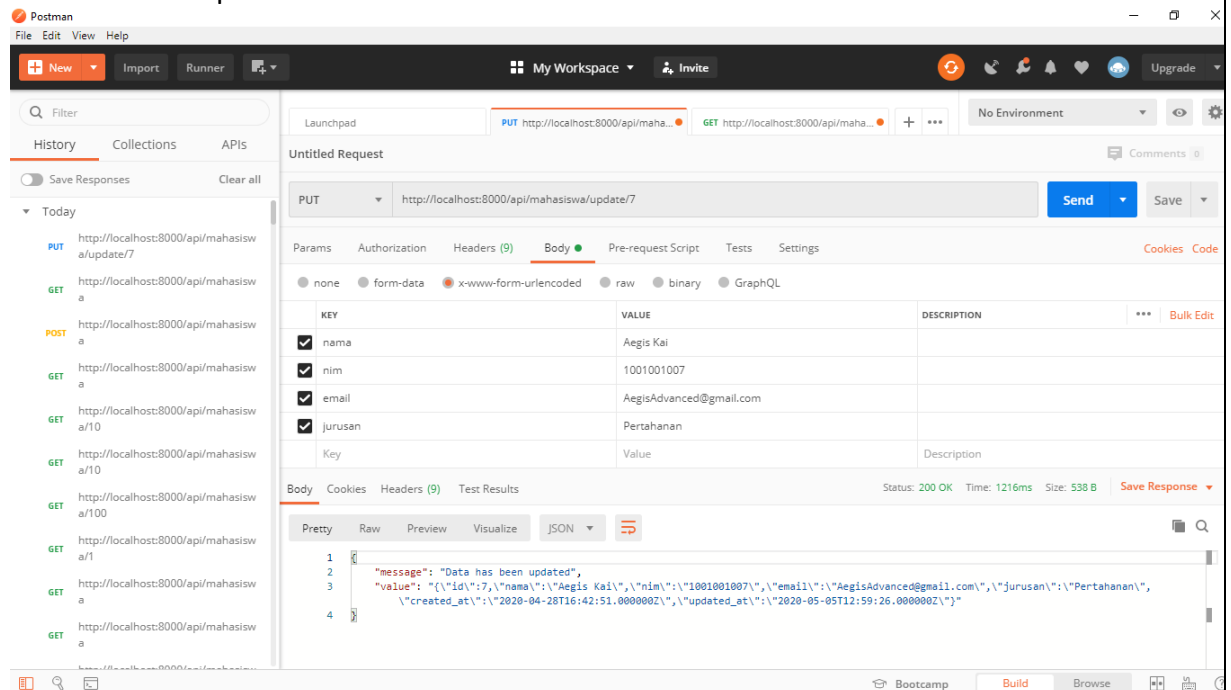
Karena kita ingin memasukkan perubahan data, maka perintah yang dipakai adalah 'put'.

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=7, maka url diisi :

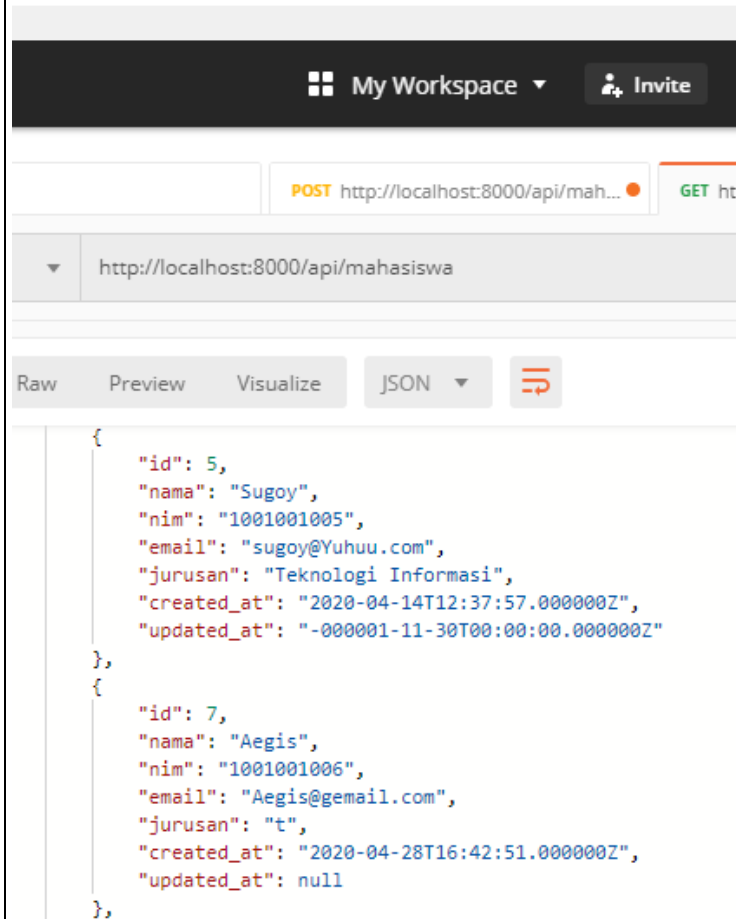
http://localhost:8000/api/mahasiswa/update/7. Pilih tab **Body** dan pilih radio button **x-www-form-urlencoded**. Isikan nama kolom pada database pada **KEY**, untuk isian data yang diubah tuliskan pada **VALUE**.



Akan muncul pesan berhasil serta perubahan data dari ID=7.

Kemudian coba untuk menampilkan data dengan ID=7 untuk melihat apakah data sudah *ter-update*.

Data sebelum diubah



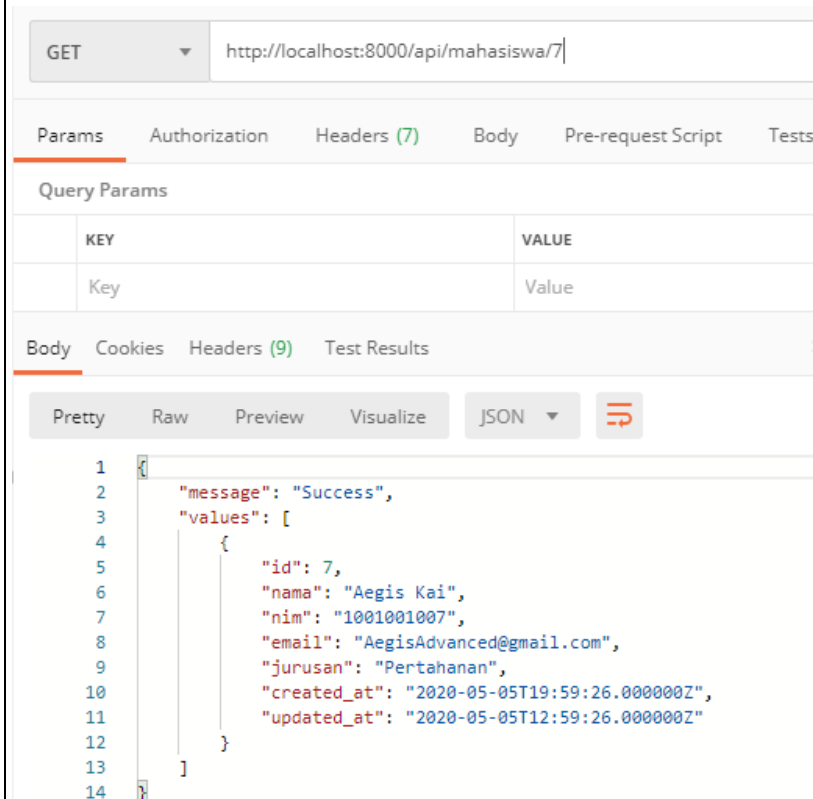
My Workspace ▼ Invite

POST http://localhost:8000/api/mah... GET http://localhost:8000/api/mahasiswa

Raw Preview Visualize JSON ⌵

```
{
  "id": 5,
  "nama": "Sugoy",
  "nim": "1001001005",
  "email": "sugoy@Yuhuu.com",
  "jurusan": "Teknologi Informasi",
  "created_at": "2020-04-14T12:37:57.000000Z",
  "updated_at": "-000001-11-30T00:00:00.000000Z"
},
{
  "id": 7,
  "nama": "Aegis",
  "nim": "1001001006",
  "email": "Aegis@gmail.com",
  "jurusan": "t",
  "created_at": "2020-04-28T16:42:51.000000Z",
  "updated_at": null
},
}
```

Data sesudah diubah



GET http://localhost:8000/api/mahasiswa/7

Params Authorization Headers (7) Body Pre-request Script Tests

Query Params

KEY	VALUE
Key	Value

Body Cookies Headers (9) Test Results

Pretty Raw Preview Visualize JSON ⌵

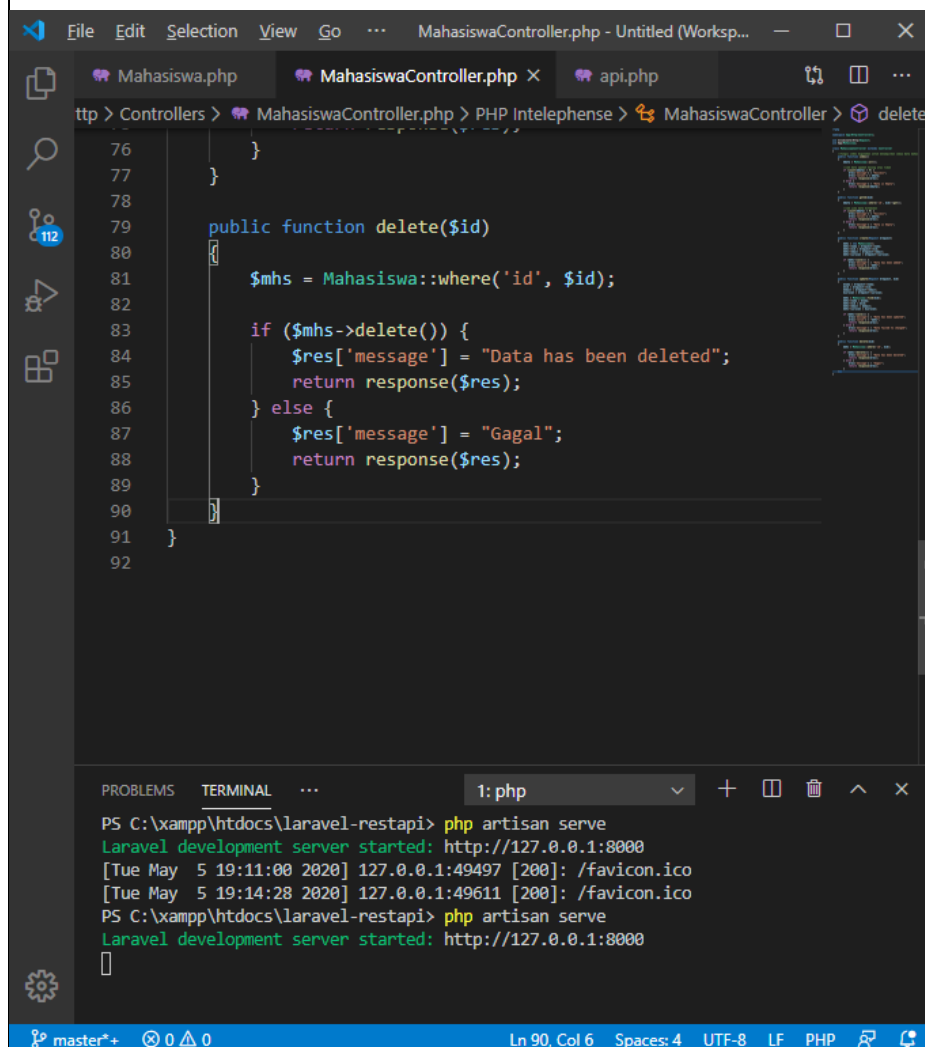
```
1 {
2   "message": "Success",
3   "values": [
4     {
5       "id": 7,
6       "nama": "Aegis Kai",
7       "nim": "1001001007",
8       "email": "AegisAdvanced@gmail.com",
9       "jurusan": "Pertahanan",
10      "created_at": "2020-05-05T19:59:26.000000Z",
11      "updated_at": "2020-05-05T12:59:26.000000Z"
12    }
13  ]
14 }
```

18

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

```
public function delete($id)
{
    $mhs = Mahasiswa::where('id', $id);

    if ($mhs->delete()) {
        $res['message'] = "Data has been deleted";
        return response($res);
    } else {
        $res['message'] = "Gagal";
        return response($res);
    }
}
```



The screenshot shows a code editor with the following content:

```
File Edit Selection View Go ... MahasiswaController.php - Untitled (Worksp...
Mahasiswa.php MahasiswaController.php X api.php
ttp > Controllers > MahasiswaController.php > PHP Intelephense > MahasiswaController > delete
76 }
77 }
78
79 public function delete($id)
80 {
81     $mhs = Mahasiswa::where('id', $id);
82
83     if ($mhs->delete()) {
84         $res['message'] = "Data has been deleted";
85         return response($res);
86     } else {
87         $res['message'] = "Gagal";
88         return response($res);
89     }
90 }
91
92
```

The terminal window at the bottom shows the following output:

```
PROBLEMS TERMINAL ... 1: php
PS C:\xampp\htdocs\laravel-restapi> php artisan serve
Laravel development server started: http://127.0.0.1:8000
[Tue May 5 19:11:00 2020] 127.0.0.1:49497 [200]: /favicon.ico
[Tue May 5 19:14:28 2020] 127.0.0.1:49611 [200]: /favicon.ico
PS C:\xampp\htdocs\laravel-restapi> php artisan serve
Laravel development server started: http://127.0.0.1:8000
```

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	<p>Tambahkan <i>route</i> untuk memanggil fungsi delete pada routes/api.php</p> <pre data-bbox="321 178 1546 220">Route::delete('/mahasiswa/{id}', 'MahasiswaController@delete');</pre> <p>Karena kita ingin menghapus data, maka perintah yang dipakai adalah 'delete.</p>
----	--

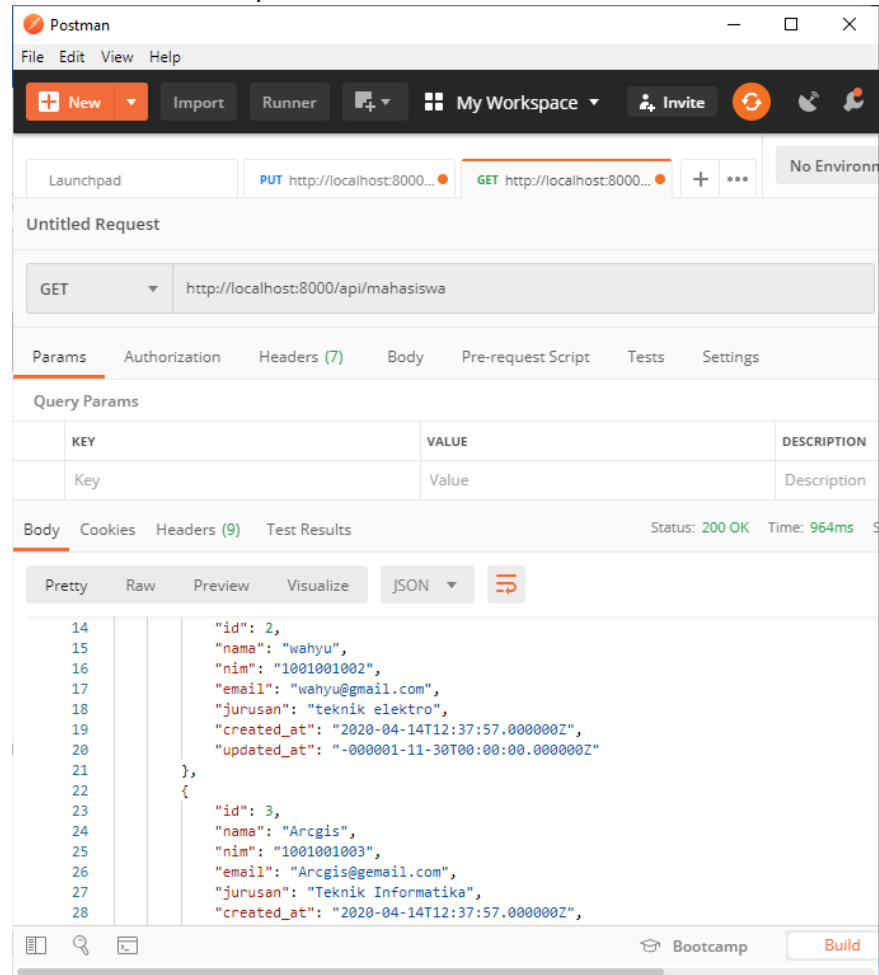
20

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

Berikut adalah contoh untuk menghapus data dengan ID=2, maka url diisi :
http://localhost:8000/api/mahasiswa /2

Muncul pesan berhasil ketika data terhapus dari database.

Data sebelum dihapus



Setelah dihapus

```
1 {
2   "message": "Data has been deleted"
3 }
```

Cek kembali

Postman

File Edit View Help

New

Import

Runner

My Workspace

Invite

Launchpad

DEL http://localhost:8000...

GET http://localhost:8000...

No Environn

Untitled Request

GET

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

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Settings

Query Params

	KEY	VALUE	DESCRIPTION
	Key	Value	Description

Body

Cookies

Headers (9)

Test Results

Status: 200 OK Time: 928ms

Pretty

Raw

Preview

Visualize

JSON

1

2

3

"message": "Data is Empty"

Bootcamp

Build

Data telah terhapus

-- Selamat Mengerjakan --