

LAPORAN PRATIKUM PEMOGRAMAN WEB LANJUT

JOBSHEET 13 – Membuat Restful API Laravel



DISUSUN OLEH :

NAMA : MARDHIYAH MILLANIA

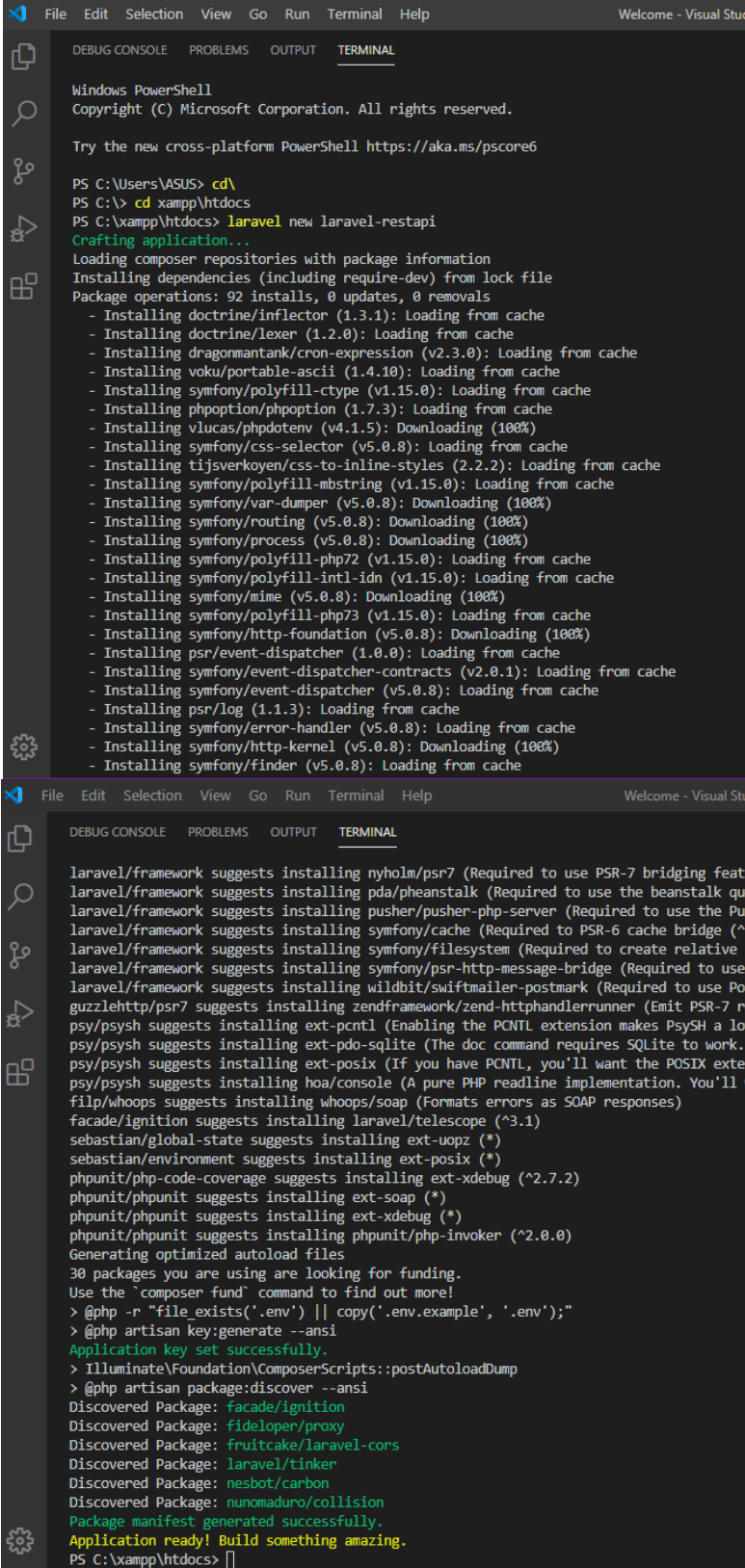
KELAS : 2A

PRODI : D-4 TEKNIK INFORMATIKA

NIM : 1841720081

ABSEN : 19

Praktikum: Membuat RESTful API di Laravel

No	Langkah
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:\Users\ASUS> cd\ PS C:\> cd xampp\htdocs 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 - Installing tijsverkoyen/css-to-inline-styles (2.2.2): Loading from cache - Installing symfony/polyfill-mbstring (v1.15.0): Loading from cache - Installing symfony/var-dumper (v5.0.8): Downloading (100%) - Installing symfony/routing (v5.0.8): Downloading (100%) - Installing symfony/process (v5.0.8): Downloading (100%) - Installing symfony/polyfill-php72 (v1.15.0): Loading from cache - Installing symfony/polyfill-intl-idn (v1.15.0): Loading from cache - Installing symfony/mime (v5.0.8): Downloading (100%) - Installing symfony/polyfill-php73 (v1.15.0): Loading from cache - Installing symfony/http-foundation (v5.0.8): Downloading (100%) - Installing psr/event-dispatcher (1.0.0): Loading from cache - Installing symfony/event-dispatcher-contracts (v2.0.1): Loading from cache - Installing symfony/event-dispatcher (v5.0.8): Loading from cache - Installing psr/log (1.1.3): Loading from cache - Installing symfony/error-handler (v5.0.8): Loading from cache - Installing symfony/http-kernel (v5.0.8): Downloading (100%) - Installing symfony/finder (v5.0.8): Loading from cache laravel/framework suggests installing nyholm/psr7 (Required to use PSR-7 bridging fea laravel/framework suggests installing pda/pheanstalk (Required to use the beanstalk qu laravel/framework suggests installing pusher/pusher-php-server (Required to use the Pu laravel/framework suggests installing symfony/cache (Required to PSR-6 cache bridge (^ laravel/framework suggests installing symfony/filesystem (Required to create relative laravel/framework suggests installing symfony/psr-http-message-bridge (Required to use laravel/framework suggests installing wilddit/swifmailer-postmark (Required to use Po guzzlehttp/psr7 suggests installing zendframework/zend-httpdierrunner (Emit PSR-7 r psy/psysh suggests installing ext-pcntl (Enabling the PCNTL extension makes PsySH a lo psy/psysh suggests installing ext-pdo-sqlite (The doc command requires SQLite to work. psy/psysh suggests installing ext-posix (If you have PCNTL, you'll want the POSIX exte psy/psysh suggests installing hoa/console (A pure PHP readline implementation. You'll t filp/whoops suggests installing whoops/soap (Formats errors as SOAP responses) facade/ignition suggests installing laravel/telescope (^3.1) sebastian/global-state suggests installing ext-uopz (*) sebastian/environment suggests installing ext-posix (*) phpunit/php-code-coverage suggests installing ext-xdebug (^2.7.2) phpunit/phpunit suggests installing ext-soap (*) phpunit/phpunit suggests installing ext-xdebug (*) phpunit/phpunit suggests installing phpunit/php-invoker (^2.0.0) Generating optimized autoload files 30 packages you are using are looking for funding. Use the `composer fund` command to find out more! > @php -r "file_exists('.env') copy('.env.example', '.env');";" > @php artisan key:generate --ansi Application key set successfully. > illuminate\Foundation\ComposerScripts::postAutoloadDump > @php artisan package:discover --ansi Discovered Package: facade/ignition Discovered Package: fideloper/proxy Discovered Package: fruitcake/laravel-cors Discovered Package: laravel/tinker Discovered Package: nesbot/carbon Discovered Package: nunomaduro/collision Package manifest generated successfully. Application ready! Build something amazing. PS C:\xampp\htdocs> </pre>
2	Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut.

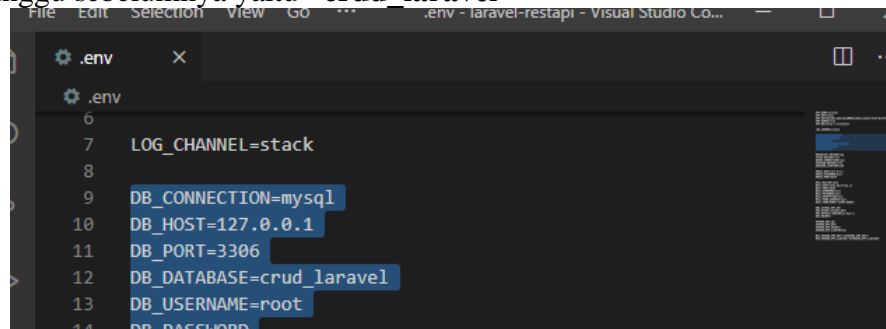
cd C:\laravel-restapi php artisan serve Akan tampil halaman default Laravel seperti di bawah ini

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



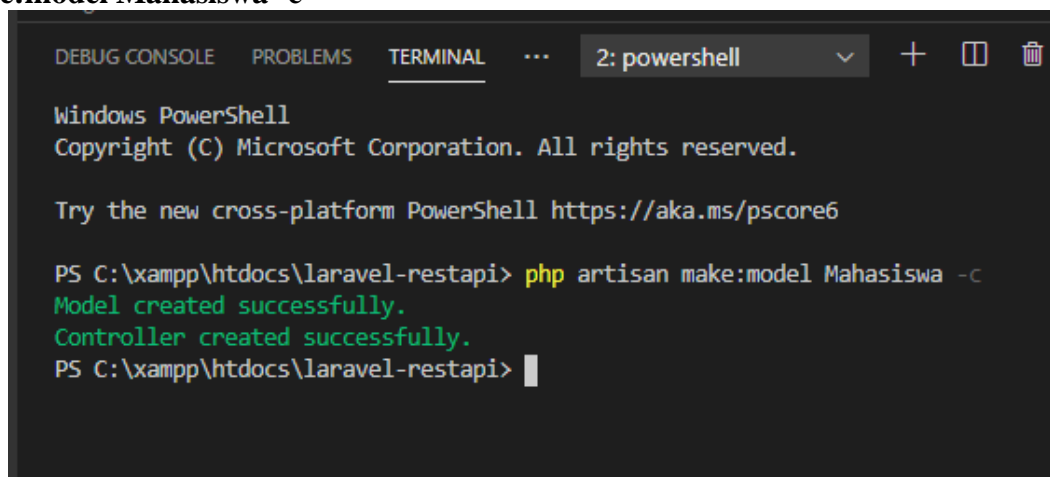
3

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 **“crud laravel”**



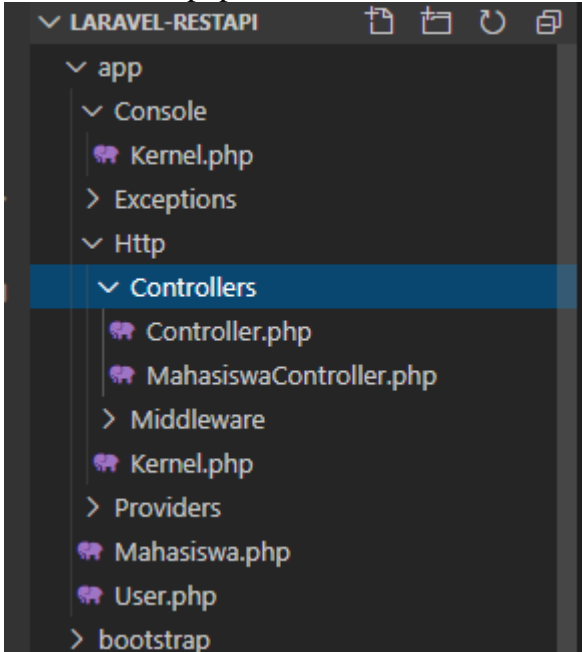
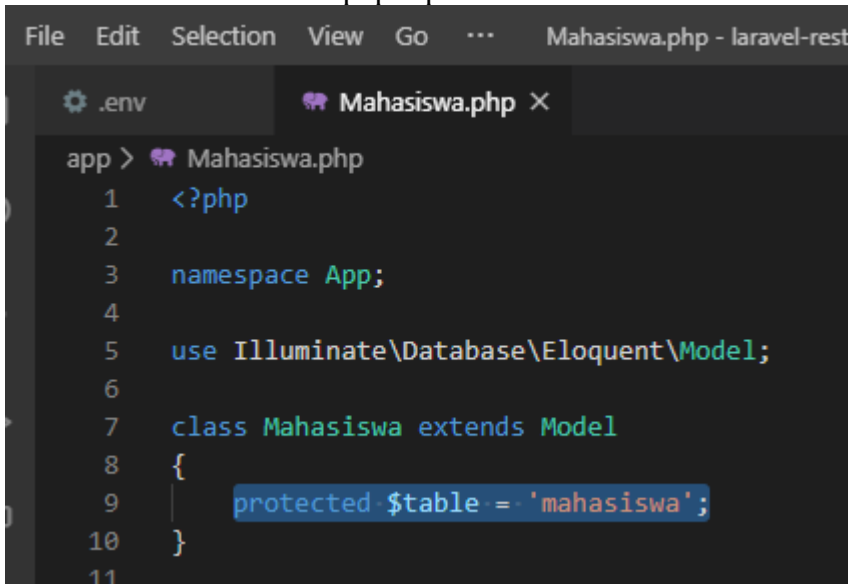
4

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

	<p>controller MahasiswaController.php.</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> 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 </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> Model ini akan mengelola tabel “mahasiswa” yang terdapat pada database crud_laravel
6	<p>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.</p>

```

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         $data = Mahasiswa::all();
13
14         //cek data tidak kosong
15         if(count($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 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

```

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

8

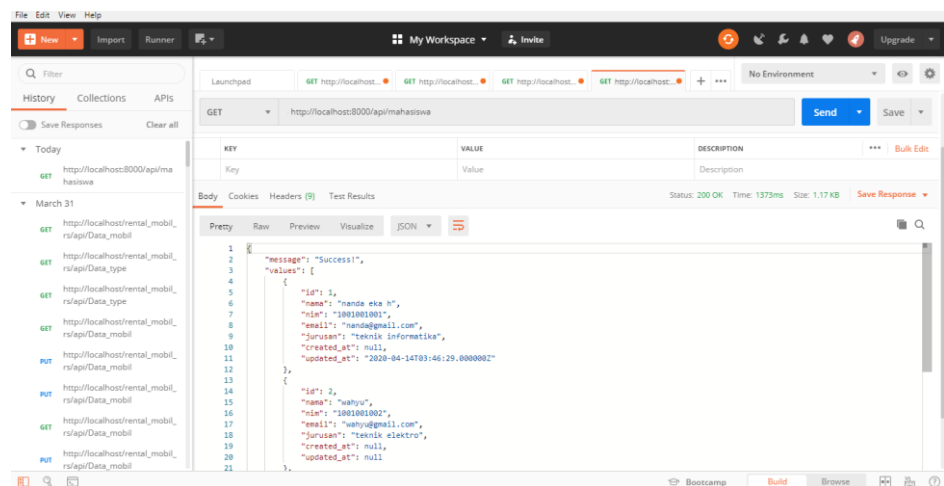
Ketikkan perintah php artisan serve pada command prompt. Lalu kita coba menguji fungsi untuk menampilkan data menggunakan aplikasi Postman. Gunakan perintah GET, isikan url : <http://localhost:8000/api/mahasiswa> Berikut adalah tampilan dari aplikasi Postman.

```

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

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

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



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.

```

app > Http > Controllers > MahasiswaController.php

25     }
26
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'] = "Success!";
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 yang dipilih

- Line 30 merupakan pemanggilan model untuk membaca data berdasarkan ID

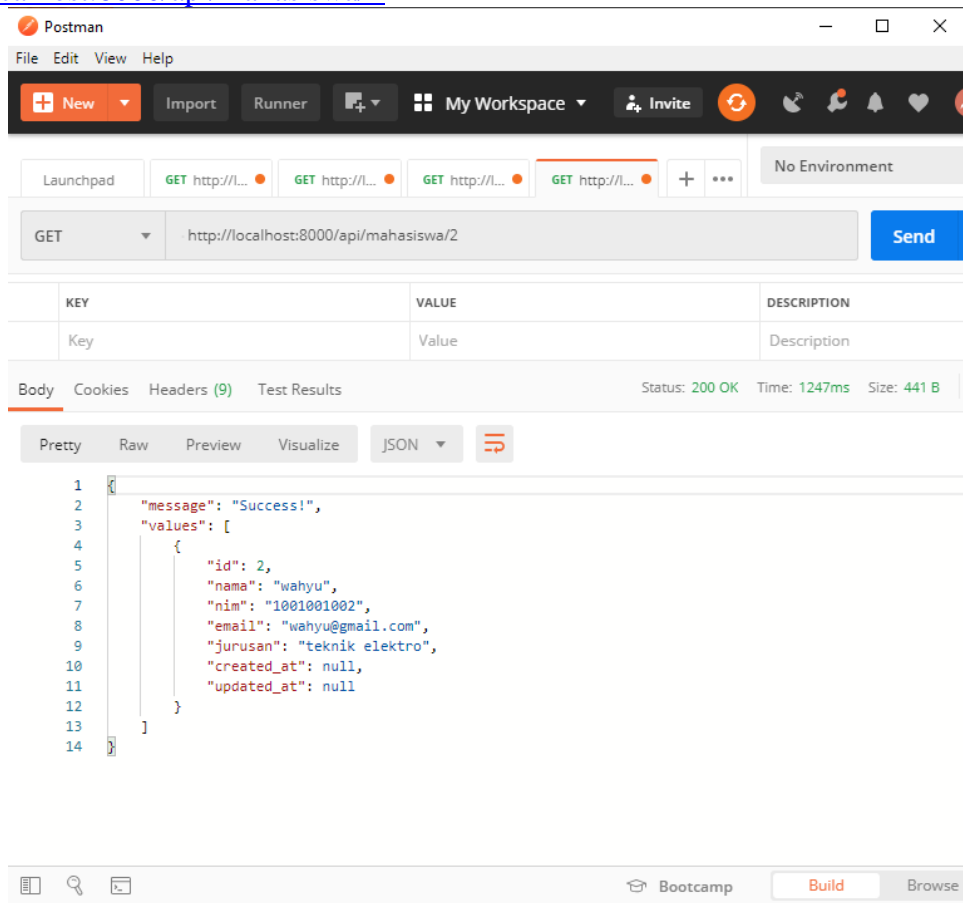
10 Tambahkan route untuk memanggil fungsi getId pada routes/api.php

```
routes > api.php
17 Route::middleware('auth:api')->get('/user', function (Request $request) {
18     return $request->user();
19 });
20
21 Route::get('mahasiswa', 'MahasiswaController@index');
22
23 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId');
24
```

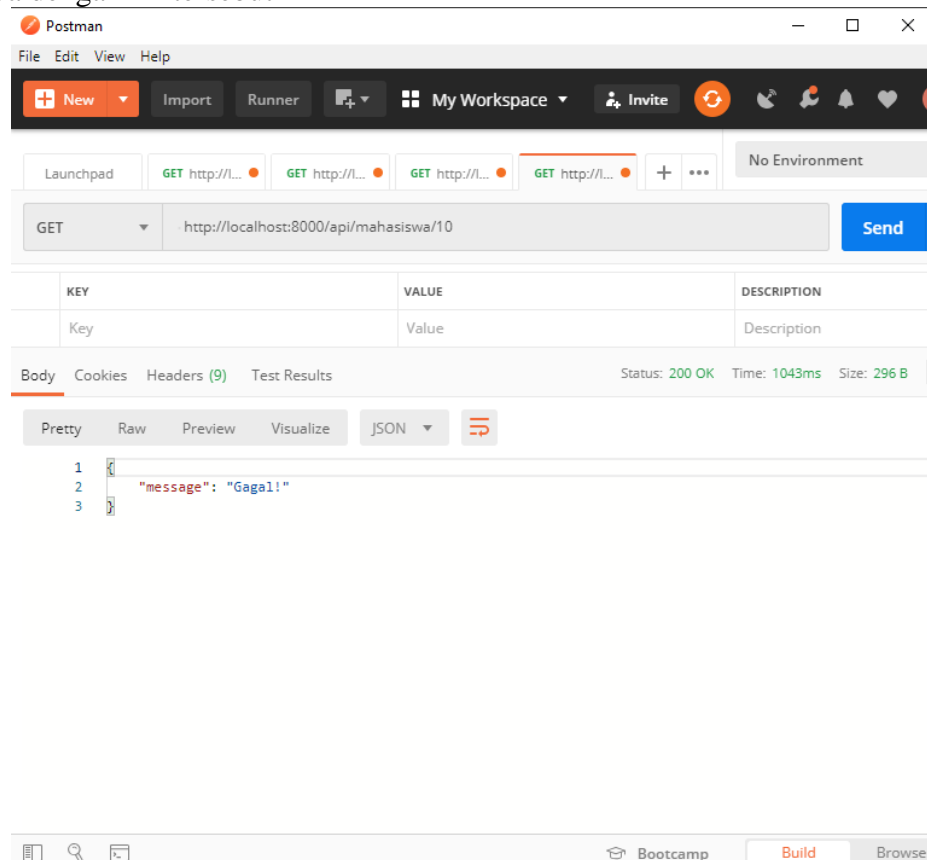
Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'

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`

```
app > Http > Controllers > MahasiswaController.php
43     }
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 }
62
```

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

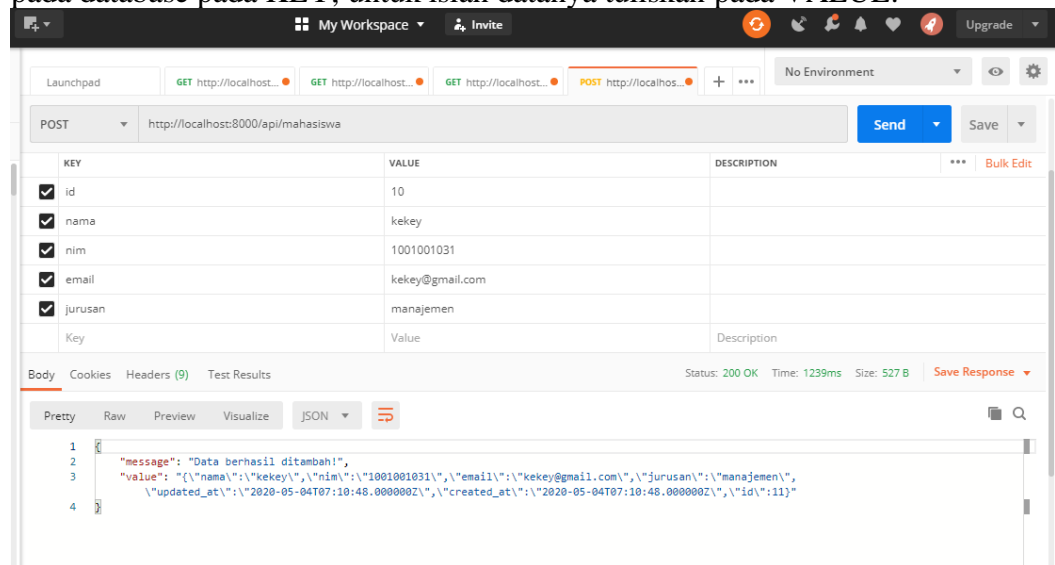
```
routes > api.php
17 Route::middleware('auth:api')->get('/user', function (Request $request) {
18     return $request->user();
19 });
20
21 Route::get('mahasiswa', 'MahasiswaController@index');
22
23 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId');
24
25 Route::post('/mahasiswa', 'MahasiswaController@create');
26
```

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

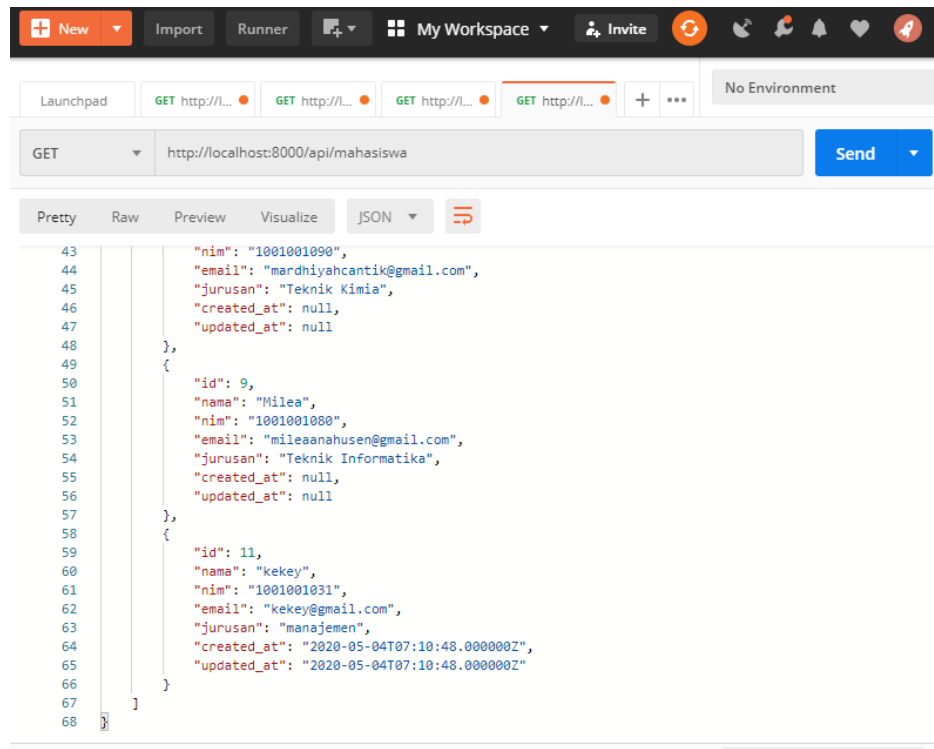
14

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



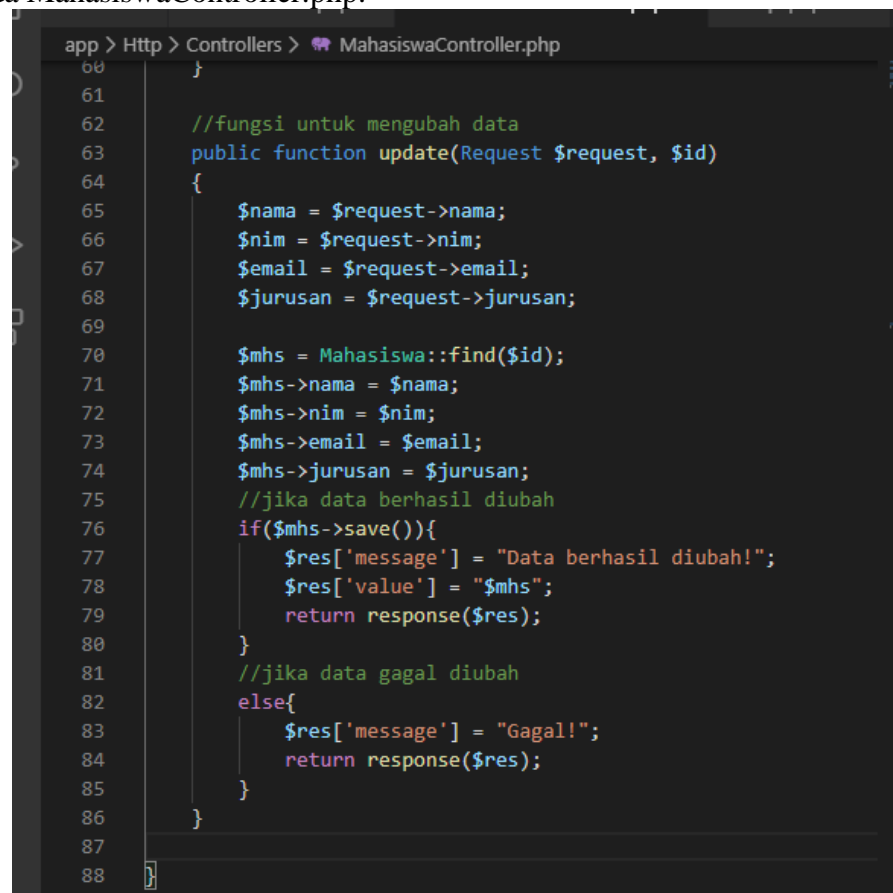
```
GET http://localhost:8000/api/mahasiswa

Pretty Raw Preview Visualize JSON

[
  {
    "nim": "1001001090",
    "email": "mardhiyhacantik@gmail.com",
    "jurusan": "Teknik Kimia",
    "created_at": null,
    "updated_at": null
  },
  {
    "id": 9,
    "nama": "Milea",
    "nim": "1001001080",
    "email": "mileaanahusen@gmail.com",
    "jurusan": "Teknik Informatika",
    "created_at": null,
    "updated_at": null
  },
  {
    "id": 11,
    "nama": "kekey",
    "nim": "1001001031",
    "email": "kekey@gmail.com",
    "jurusan": "manajemen",
    "created_at": "2020-05-04T07:10:48.000000Z",
    "updated_at": "2020-05-04T07:10:48.000000Z"
  }
]
```

15

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



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

60 }
61
62 //fungsi untuk mengubah data
63 public function update(Request $request, $id)
64 {
65     $nama = $request->nama;
66     $nim = $request->nim;
67     $email = $request->email;
68     $jurusan = $request->jurusan;
69
70     $mhs = Mahasiswa::find($id);
71     $mhs->nama = $nama;
72     $mhs->nim = $nim;
73     $mhs->email = $email;
74     $mhs->jurusan = $jurusan;
75     //jika data berhasil diubah
76     if($mhs->save()){
77         $res['message'] = "Data berhasil diubah!";
78         $res['value'] = "$mhs";
79         return response($res);
80     }
81     //jika data gagal diubah
82     else{
83         $res['message'] = "Gagal!";
84         return response($res);
85     }
86 }
87
88
```

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

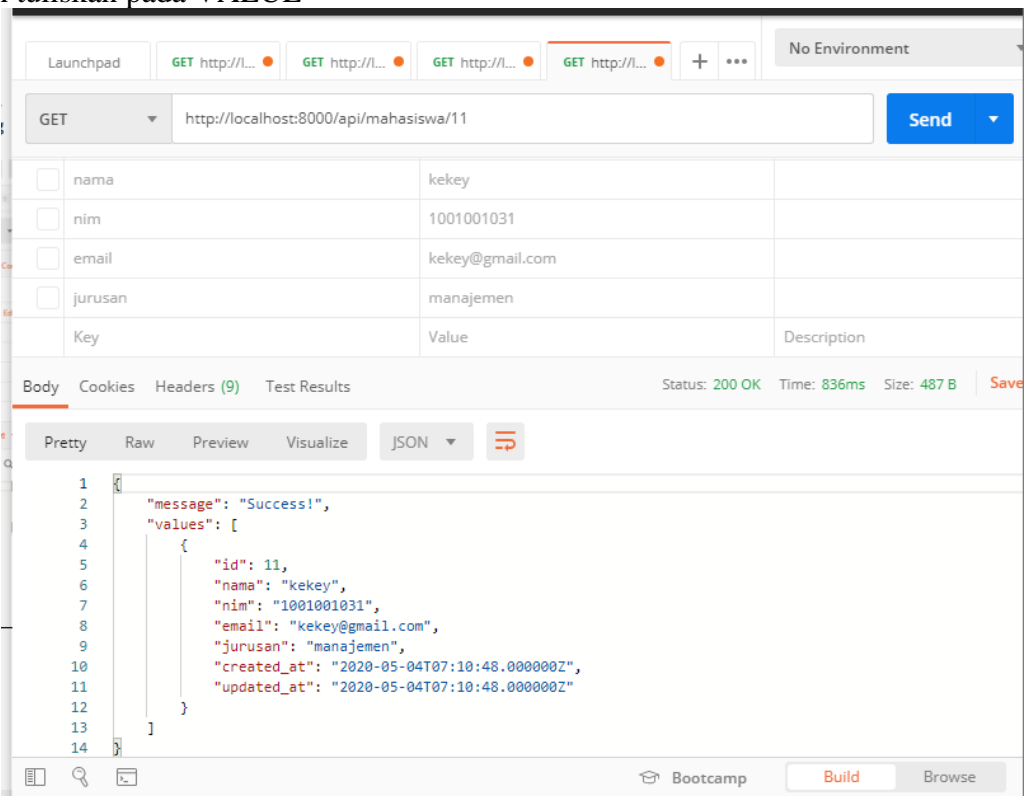
```

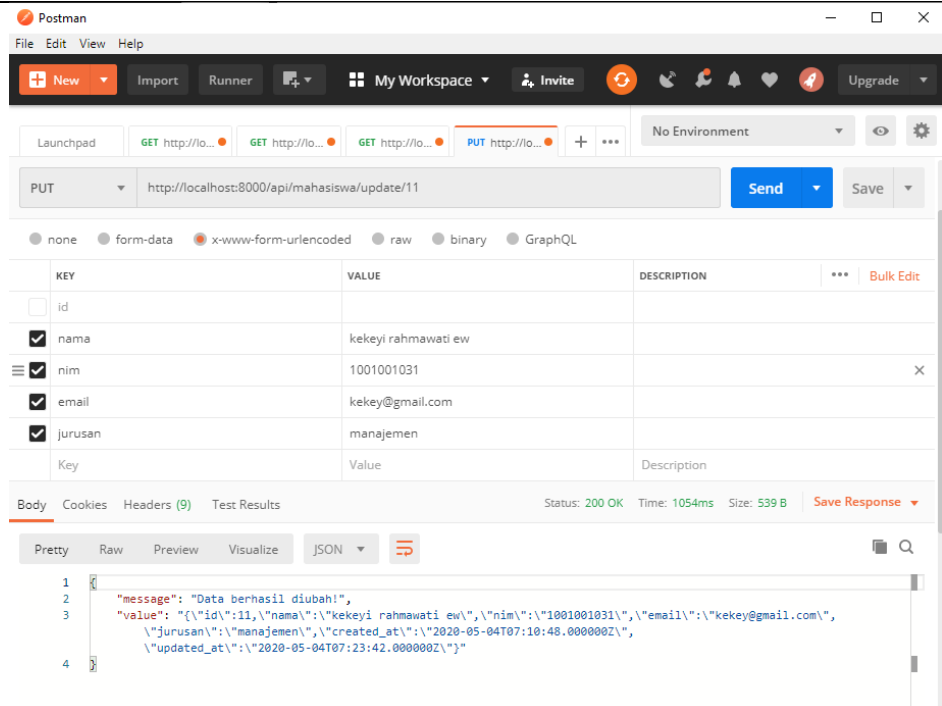
routes > api.php
17 Route::middleware('auth:api')->get('/user', function (Request $request) {
18     return $request->user();
19 });
20
21 Route::get('mahasiswa', 'MahasiswaController@index');
22
23 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId');
24
25 Route::post('/mahasiswa', 'MahasiswaController@create');
26
27 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');
28
  
```

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

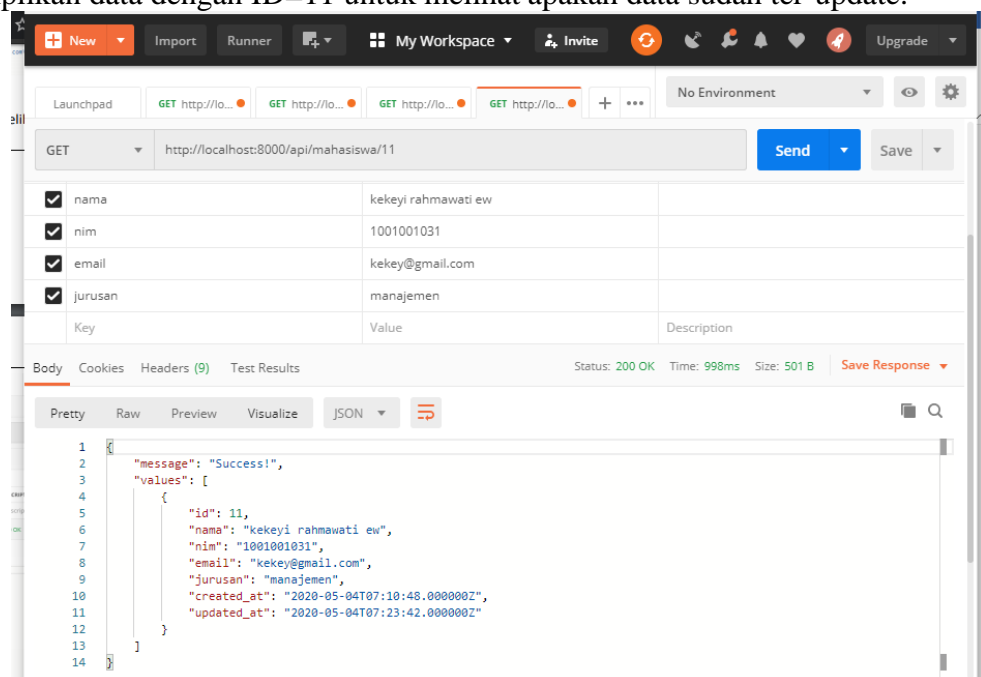
17

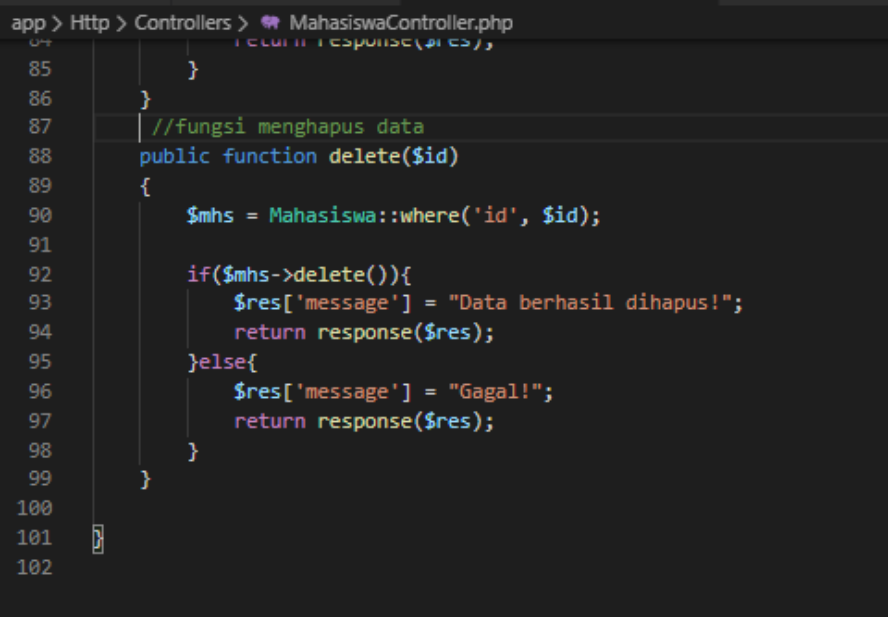

Sekarang kita coba untuk mengubah data berdasarkan ID mahasiswa yang dipilih menggunakan Postman. Gunakan perintah PUT untuk mengubah data. Berikut adalah contoh untuk mengubah data dengan ID=11, maka url diisi : <http://localhost:8000/api/mahasiswa/update/11>. Pilih tab Body dan pilih radio button x-www-form-urlencoded. Isikan nama kolom pada database pada KEY, untuk isian data yang diubah tuliskan pada VALUE

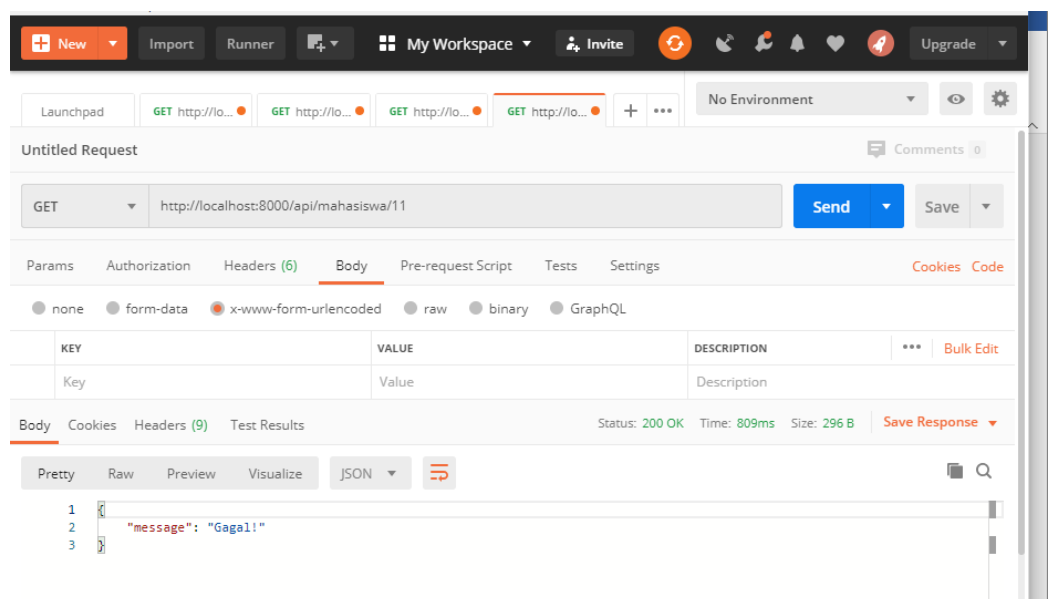
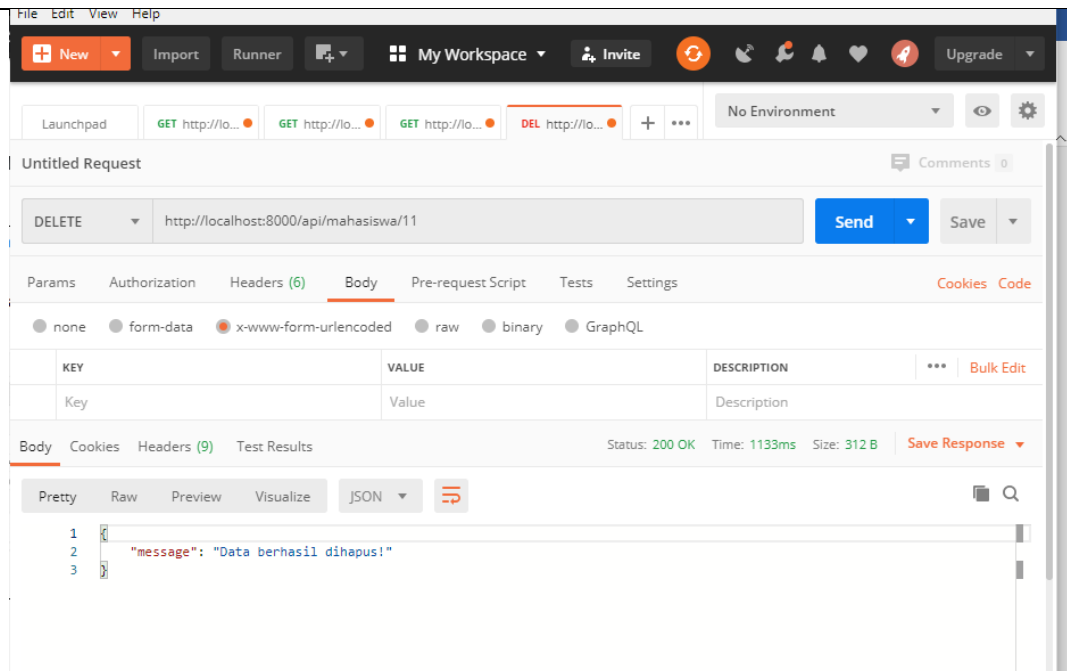




Akan muncul pesan berhasil serta perubahan data dari ID=11. Kemudian coba untuk menampilkan data dengan ID=11 untuk melihat apakah data sudah ter-update.



18	<p>Terakhir kita akan membuat fungsi untuk menghapus data dengan nama delete di MahasiswaController.php</p>  <pre> app > Http > Controllers > MahasiswaController.php 85 } 86 } 87 //fungsi 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 } 100 101 102 </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> • 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 route untuk memanggil fungsi delete pada routes/api.php</p>  <pre> routes > api.php 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 23 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId'); 24 25 Route::post('/mahasiswa', 'MahasiswaController@create'); 26 27 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update'); 28 29 Route::delete('/mahasiswa/{id}', 'MahasiswaController@delete'); </pre> <p>Karena kita ingin menghapus data, maka perintah yang dipakai adalah 'delete</p>
20	<p>Buka Postman untuk mencoba menghapus data dari database. Gunakan perintah DELETE untuk mengubah data.</p> <p>Berikut adalah contoh untuk menghapus data dengan ID=11, maka url diisi : http://localhost:8000/api/mahasiswa /101</p>



Muncul pesan berhasil ketika data terhapus dari database.