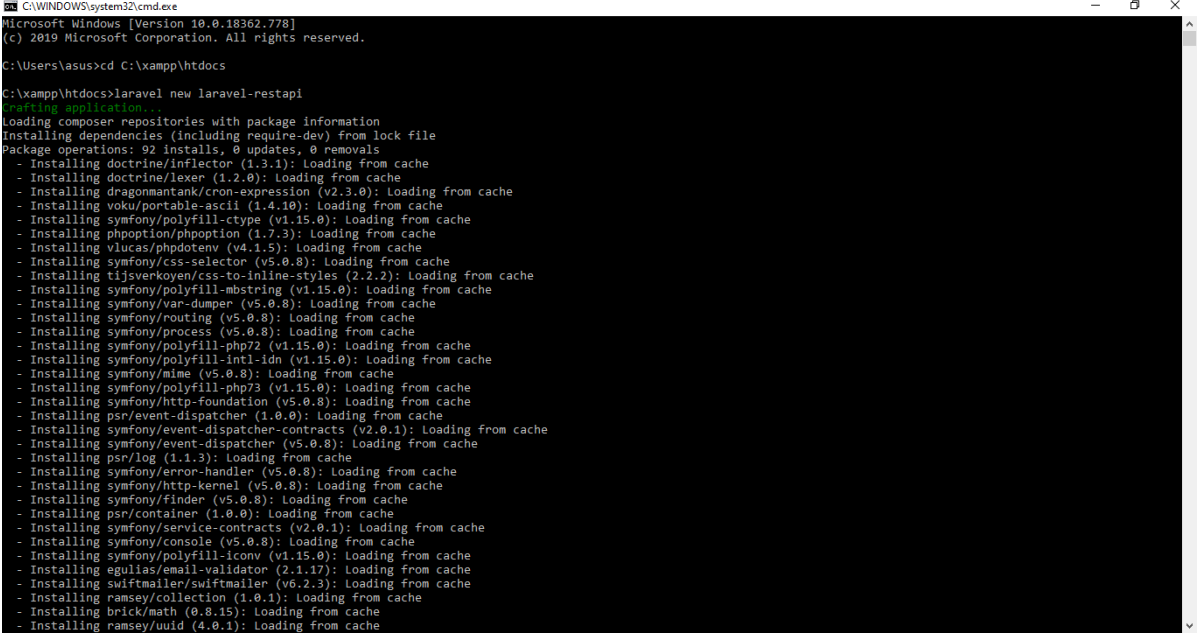




Jurusan Teknologi Informasi Politeknik Negeri Malang  
**Jobsheet-14: Membuat RESTful API Laravel**  
**Mata Kuliah Pemrograman Web Lanjut**  
Pengampu: Tim Ajar Pemrograman Web Lanjut  
Mei 2020

Nama : Septianda Reza Maulana  
Kelas : TI-2A  
NIM : 1841720135

**Praktikum: Membuat RESTful API di Laravel**

Langkah	Keterangan
1	<p>Buat project baru dengan nama “<b>laravel-restapi</b>”. Buka command prompt, tuliskan perintah berikut.</p> <pre>cd C:\xampp\htdocs laravel new laravel-restapi</pre> 

```
- Installing ramsey/collection (1.0.1): Loading from cache
- Installing brick/math (0.8.15): Loading from cache
- Installing ramsey/uuid (4.0.1): Loading from cache
- Installing psr/simple-cache (1.0.1): Loading from cache
- Installing opis/closure (3.5.1): Loading from cache
- Installing symfony/translation-contracts (v2.0.1): Loading from cache
- Installing symfony/translation (v5.0.3): Loading from cache
- Installing nesbot/carbon (2.33.0): Loading from cache
- Installing monolog/monolog (2.0.2): Loading from cache
- Installing league/flysystem (1.0.67): Loading from cache
- Installing league/commonmark (1.4.2): Loading from cache
- Installing laravel/framework (v7.9.2): Loading from cache
- Installing fidoalloy/proxy (4.3.0): Loading from cache
- Installing asm89/stack-cors (1.3.0): Loading from cache
- Installing fruitcake/laravel-cors (v1.0.6): Loading from cache
- Installing ralouphie/getallheaders (3.0.3): Loading from cache
- Installing psr/http-message (1.0.1): Loading from cache
- Installing guzzlehttp/psr7 (1.6.1): Loading from cache
- Installing guzzlehttp/promises (v1.3.1): Loading from cache
- Installing guzzlehttp/guzzle (6.5.3): Loading from cache
- Installing nikic/php-parser (v4.4.0): Loading from cache
- Installing dnoegel/php-xdg-base-dir (v0.1.1): Loading from cache
- Installing psy/psysh (v0.10.4): Loading from cache
- Installing laravel/tinker (v2.4.0): Loading from cache
- Installing scrivo/highlight.php (v9.18.1.1): Loading from cache
- Installing filp/whoops (2.7.1): Loading from cache
- Installing facade/contracts (1.0.0): Loading from cache
- Installing facade/flame-client-php (1.3.2): Loading from cache
- Installing facade/ignition (2.0.2): Loading from cache
- Installing fzaninotto/faker (v1.9.1): Loading from cache
- Installing hamcrest/hamcrest-php (v2.0.0): Loading from cache
- Installing mockery/mockery (1.3.1): Loading from cache
- Installing nunomaduro/collision (v4.2.0): Loading from cache
- Installing webmozart/assert (1.8.0): Loading from cache
- Installing phpdocumentor/reflection-common (2.2.0): Loading from cache
- Installing phpdocumentor/type-resolver (1.1.0): Loading from cache
- Installing phpdocumentor/reflection-docblock (5.1.0): Loading from cache
- Installing phpunit/php-token-stream (3.1.1): Loading from cache
- Installing sebastian/version (2.0.1): Loading from cache
- Installing sebastian/type (1.1.3): Loading from cache
- Installing sebastian/resource-operations (2.0.1): Loading from cache
- Installing sebastian/recursion-context (3.0.0): Loading from cache
- Installing sebastian/object-reflector (1.1.1): Loading from cache
- Installing sebastian/object-enumerator (3.0.3): Loading from cache
```

```
- Installing sebastian/object-enumerator (3.0.3): Loading from cache
- Installing sebastian/global-state (3.0.0): Loading from cache
- Installing sebastian/exporter (3.1.2): Loading from cache
- Installing sebastian/environment (4.2.3): Loading from cache
- Installing sebastian/diff (3.0.2): Loading from cache
- Installing sebastian/comparator (3.0.2): Loading from cache
- Installing phpunit/php-timer (2.1.2): Loading from cache
- Installing phpunit/php-text-template (1.2.1): Loading from cache
- Installing phpunit/php-file-iterator (2.0.2): Loading from cache
- Installing theseer/tokenizer (1.1.3): Loading from cache
- Installing sebastian/code-unit-reverse-lookup (1.0.1): Loading from cache
- Installing phpunit/php-code-coverage (7.0.10): Loading from cache
- Installing doctrine/instantiator (1.3.0): Loading from cache
- Installing phpspec/prophecy (v1.0.3): Loading from cache
- Installing phar-io/version (2.0.1): Loading from cache
- Installing phar-io/manifest (1.0.3): Loading from cache
- Installing myclabs/deep-copy (1.9.5): Loading from cache
- Installing phpunit/phpunit (8.5.4): Loading from cache
voku/portable-ascii suggests installing ext-intl (Use Intl for transliterator_transliterate() support)
symfony/var-dumper suggests installing ext-intl (To show region name in time zone dump)
symfony/routing suggests installing doctrine/annotations (For using the annotation loader)
symfony/routing suggests installing symfony/config (For using the all-in-one router or any loader)
symfony/routing suggests installing symfony/expression-language (For using expression matching)
symfony/routing suggests installing symfony/yaml (For using the YAML loader)
symfony/polyfill-intl-idn suggests installing ext-intl (For best performance)
symfony/event-dispatcher suggests installing symfony/dependency-injection
symfony/http-kernel suggests installing symfony/browser-kit
symfony/http-kernel suggests installing symfony/config
symfony/http-kernel suggests installing symfony/dependency-injection
symfony/service-contracts suggests installing symfony/service-implementation
symfony/console suggests installing symfony/lock
egulias/email-validator suggests installing ext-intl (PHP Internationalization Libraries are required to use the SpoofChecking validation)
swiftmailer/swiftmailer suggests installing ext-intl (Needed to support internationalized email addresses)
swiftmailer/swiftmailer suggests installing true/punycode (Needed to support internationalized email addresses, if ext-intl is not installed)
ramsey/uuid suggests installing ext-gmp (Enables faster math with arbitrary-precision integers using GMP.)
ramsey/uuid suggests installing ext-uuid (Enables the use of ReUuidTimeGenerator and PeclUuidRandomGenerator.)
ramsey/uuid suggests installing paragonie/random-lib (Provides RandomLib for use with the RandomLibAdapter)
ramsey/uuid suggests installing ramsey/uuid-doctrine (Allows the use of Ramsey\Uuid\Uuid as Doctrine field type.)
symfony/translation suggests installing symfony/config
symfony/translation suggests installing symfony/yaml
monolog/monolog suggests installing aws/aws-sdk-php (Allow sending log messages to AWS services like DynamoDB)
monolog/monolog suggests installing doctrine/couchdb (Allow sending log messages to a CouchDB server)
monolog/monolog suggests installing elasticsearch/elasticsearch (Allow sending log messages to an Elasticsearch server via official client)
monolog/monolog suggests installing ext-sockets (Allow sending log messages to an AWS SQS server. (1.0.0, required))
```

```

C:\WINDOWS\system32\cmd.exe
monolog/monolog suggests installing ext-amqp (Allow sending log messages to an AMQP server (1.0+ required))
monolog/monolog suggests installing ext-mongodb (Allow sending log messages to a MongoDB server (via driver))
monolog/monolog suggests installing graylog2/gelf-php (Allow sending log messages to a Graylog2 server)
monolog/monolog suggests installing mongodb/mongodb (Allow sending log messages to a MongoDB server (via library))
monolog/monolog suggests installing php-amqp/php-amqp (Allow sending log messages to an AMQP server using php-amqp)
monolog/monolog suggests installing php-console/php-console (Allow sending log messages to Google Chrome)
monolog/monolog suggests installing rollbar/rollbar (Allow sending log messages to Rollbar)
monolog/monolog suggests installing rufin/elastic (Allow sending log messages to an Elastic Search server)
monolog/monolog suggests installing league/flysystem-aws-s3-v2 (Allows you to use S3 storage with AWS SDK v2)
league/flysystem suggests installing league/flysystem-aws-s3-v3 (Allows you to use S3 storage with AWS SDK v3)
league/flysystem suggests installing league/flysystem-azure (Allows you to use Windows Azure Blob storage)
league/flysystem suggests installing league/flysystem-cached-adapter (Flysystem adapter decorator for metadata caching)
league/flysystem suggests installing league/flysystem-eventable-flysystem (Allows you to use EventableFlysystem)
league/flysystem suggests installing league/flysystem-filesystem (Allows you to use Rackspace Cloud Files)
league/flysystem suggests installing league/flysystem-sftp (Allows you to use SFTP server storage via phpseclib)
league/flysystem suggests installing league/flysystem-webdav (Allows you to use WebDAV storage)
league/flysystem suggests installing league/flysystem-ziparchive (Allows you to use ZipArchive adapter)
league/flysystem suggests installing spatie/flysystem-dropbox (Allows you to use Dropbox storage)
league/flysystem suggests installing srmlive/flysystem-dropbox-v2 (Allows you to use Dropbox storage for PHP 5 applications)
laravel/framework suggests installing aws/aws-sdk-php (Required to use the SQS queue driver, DynamoDB failed job storage and SES mail driver (^3.0).)
laravel/framework suggests installing doctrine/dbal (Required to rename columns and drop SQLite columns (^2.6).)
laravel/framework suggests installing ext-memcached (Required to use the memcached cache driver.)
laravel/framework suggests installing ext-pcntl (Required to use all features of the queue worker.)
laravel/framework suggests installing ext-posix (Required to use all features of the queue worker.)
laravel/framework suggests installing ext-redis (Required to use the Redis cache and queue drivers (^4.0|^5.0).)
laravel/framework suggests installing league/flysystem-aws-s3-v3 (Required to use the Flysystem S3 driver (^1.0).)
laravel/framework suggests installing league/flysystem-cached-adapter (Required to use the Flysystem cache (^1.0).)
laravel/framework suggests installing league/flysystem-sftp (Required to use the Flysystem SFTP driver (^1.0).)
laravel/framework suggests installing montonator/montonator (Required to use the ordered storage driver (^1.1).)
laravel/framework suggests installing nyholm/psr7 (Required to use PSR-7 bridging features (^1.2).)
laravel/framework suggests installing pda/phanstalk (Required to use the beanstalk queue driver (^4.0).)
laravel/framework suggests installing pusher/pusher-php-server (Required to use the Pusher broadcast driver (^4.0).)
laravel/framework suggests installing symfony/cache (Required to PSR-6 cache bridge (^5.0).)
laravel/framework suggests installing symfony/filesystem (Required to create relative storage directory symbolic links (^5.0).)
laravel/framework suggests installing symfony/http-message (Required to use PSR-7 bridge (^3.0).)
laravel/framework suggests installing wildbit/swiftmailer-postmark (Required to use Postmark mail driver (^3.0).)
guzzlehttp/psr7 suggests installing zendframework/zend-httphandlerrunner (Emit PSR-7 responses)
psr/psrsh suggests installing ext-pcntl (Enabling the PCNTL extension makes PsrSh a lot happier :))
psr/psrsh suggests installing ext-pdo-sqlite (The doc command requires SQLite to work.)
psr/psrsh suggests installing ext-posix (If you have PCNTL, you'll want the POSIX extension as well.)
psr/psrsh suggests installing hoa/console (A pure PHP readline implementation. You'll want this if your PHP install doesn't already provide readline or libedit.)
laravel/framework suggests installing whoops/whoops (Formats errors as SOAP responses)
facade/ignition suggests installing laravel/telescope (^3.1)
sebastian/global-state suggests installing ext-woor (^1.0)

```

```
C:\WINDOWS\system32\cmd.exe
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.
C:\xampp\htdocs>
```

This PC > Local Disk (C:) > xampp > htdocs

Name	Date modified	Type	Size
belajarjavascript	10-Nov-19 16:54	File folder	
Code-Kedelapan	07-Apr-20 8:10	File folder	
dasarWeb	09-Dec-19 16:14	File folder	
dashboard	10-Nov-19 16:36	File folder	
DOM	10-Nov-19 16:56	File folder	
img	10-Nov-19 16:36	File folder	
Kelompok-2	03-May-20 20:49	File folder	
laravelapp	07-Apr-20 10:21	File folder	
laravel-crud	27-Apr-20 23:48	File folder	
laravel-crud-kedua	14-Apr-20 10:59	File folder	
laravel-restapi	04-May-20 20:32	File folder	

Kita coba jalankan dulu project tersebut. Pada command prompt tulis perintah berikut.

**Cd C:\laravel-restapi php  
artisan serve**

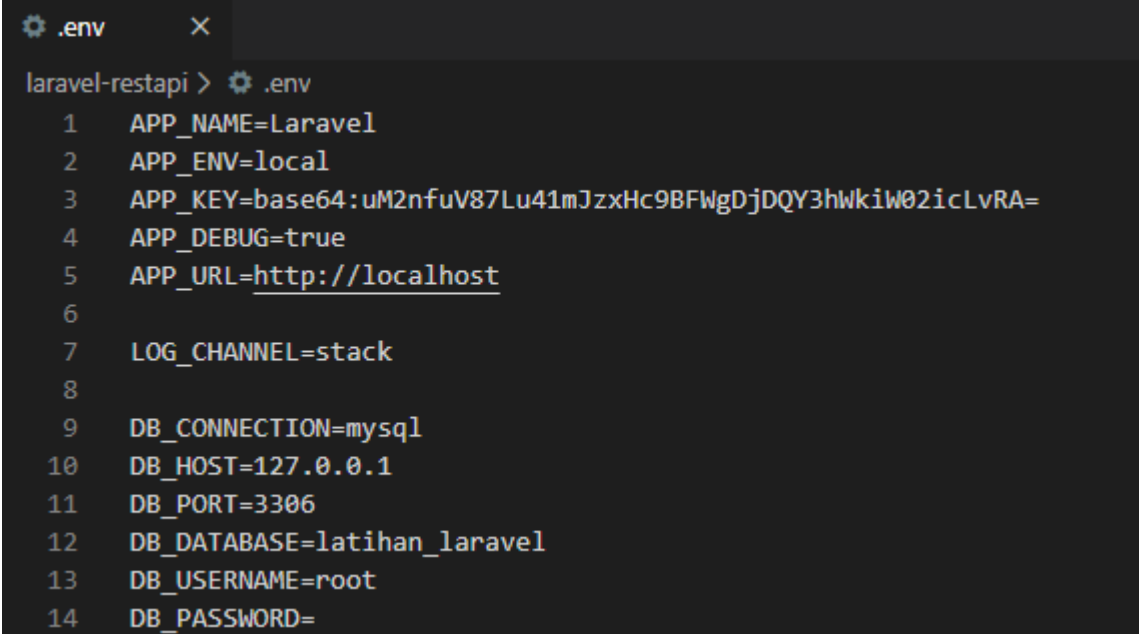

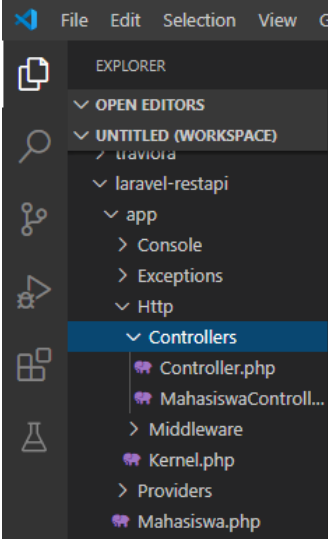
```
C:\WINDOWS\system32\cmd.exe - php artisan serve
Application ready! Build something amazing.
C:\xampp\htdocs>cd C:\xampp\htdocs\laravel-restapi
C:\xampp\htdocs\laravel-restapi>php artisan serve
Laravel development server started: http://127.0.0.1:8000
```

Akan tampil halaman default Laravel seperti di bawah ini.



# Laravel

[DOCS](#) [LARACASTS](#) [NEWS](#) [BLOG](#) [NOVA](#) [FORGE](#) [VAPOR](#) [GITHUB](#)

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> 1 APP_NAME=Laravel 2 APP_ENV=local 3 APP_KEY=base64:uM2nfuV87Lu41mJzxHc9BFWgDjDQY3hWkiW02icLvRA= 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= </pre>
4	<p>Buat <b>model</b> dengan nama <b>Mahasiswa</b>, buat juga <b>controllernya</b>. Untuk membuat model dan <b>controllernya</b> sekaligus tuliskan perintah berikut pada <i>command prompt</i> (terlebih dahulu keluar dari php artisan serve dengan mengetik ctrl+C pada keyboard)</p> <p><b>php artisan make:model Mahasiswa -c</b></p> <p>Keterangan :</p> <ul style="list-style-type: none"> <li>• -c merupakan perintah untuk menyertakan pembuatan <i>controller</i></li> </ul>  <p>Sehingga pada project laravel-restapi akan bertambah dua file yaitu <b>model Mahasiswa.php</b> serta <b>controller MahasiswaController.php</b>.</p> 

Selanjutnya ubah isi model Mahasiswa.php seperti berikut ini.

5

```
Mahasiswa.php X
laravel-restapi > 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
```

Keterangan:

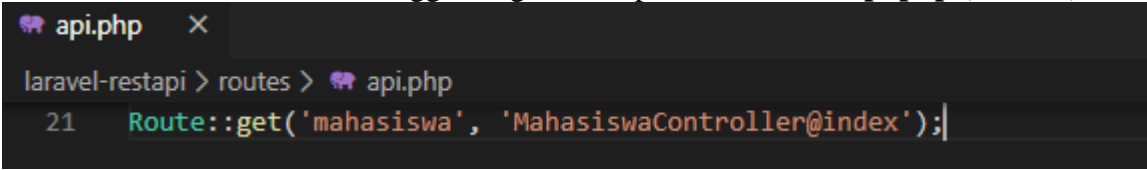
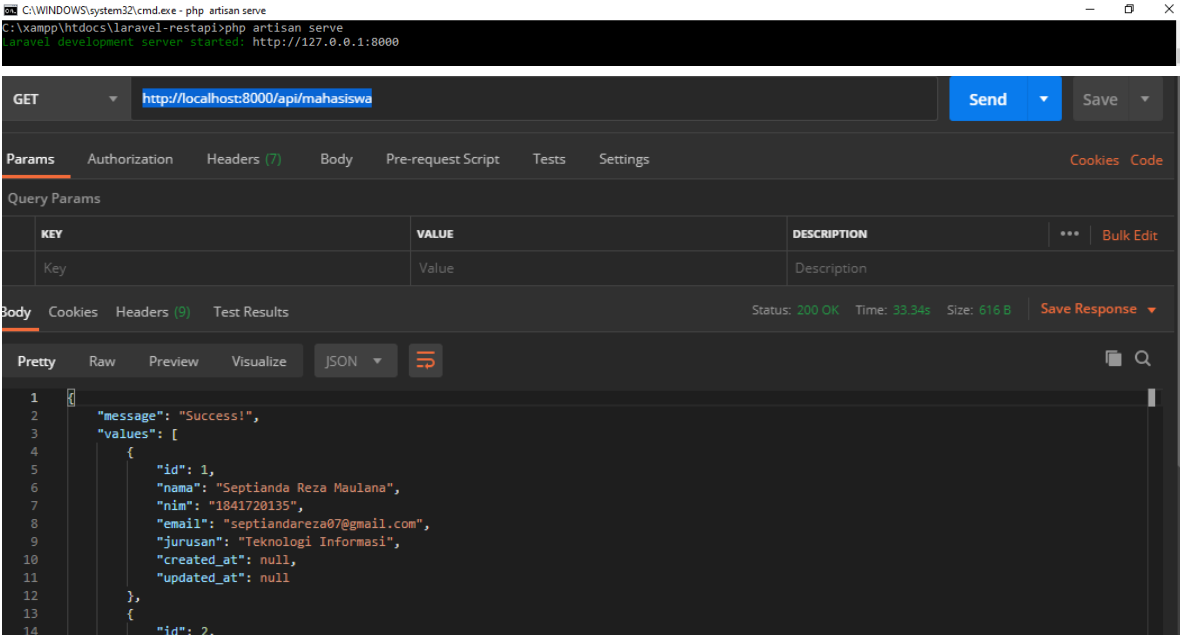
- Model ini akan mengelola tabel “mahasiswa” yang terdapat pada database latihan\_laravel

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.

6

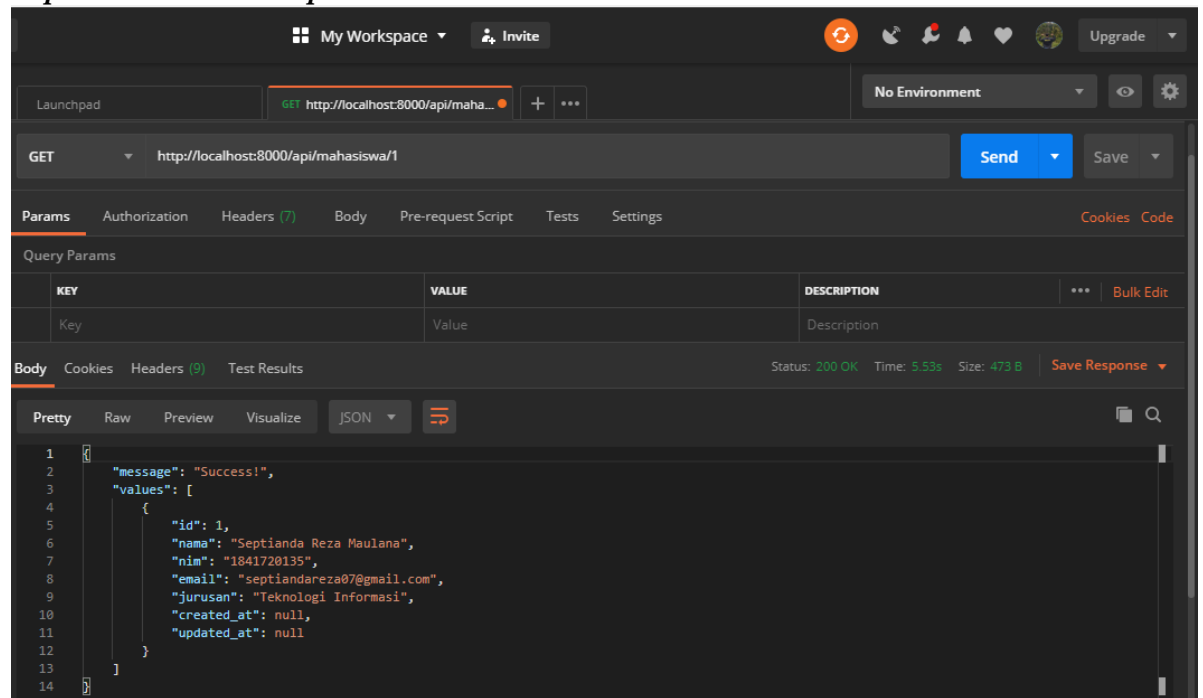
```
MahasiswaController.php X
laravel-restapi > app > Http > Controllers > MahasiswaController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Mahasiswa;
7
8  class MahasiswaController extends Controller
9  {
10     //fungsi index digunakan untuk menampilkan 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             $res['message'] = "Success!";
23             return response($res);
24         }
25     }
26 }
```

	<p>Keterangan:</p> <ul style="list-style-type: none"> <li>• Tambahkan line 6 agar model Mahasiswa dapat digunakan pada MahasiswaController</li> <li>• Line 15-19 digunakan untuk memeriksa apakah data &gt; 0 atau data tidak kosong</li> <li>• Variabel \$res[message] digunakan untuk menampilkan pesan apakah ada data atau tidak ada data di tabel mahasiswa</li> <li>• Variabel \$array[values] akan menyimpan semua baris data pada tabel mahasiswa</li> </ul>
7	<p>Tambahkan route untuk memanggil fungsi index pada file <b>routes/api.php</b> (Line 21).</p>  <p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'.</p>
8	<p>Ketikkan perintah <b>php artisan serve</b> pada <i>command prompt</i>. Lalu kita coba menguji fungsi untuk menampilkan data menggunakan aplikasi <b>Postman</b>. Gunakan perintah <b>GET</b>, isikan url : <b>http://localhost:8000/api/mahasiswa</b></p> <p>Berikut adalah tampilan dari aplikasi Postman.</p>  <p>Semua data pada tabel mahasiswa akan tampil, ditampilkan juga pesan sukses.</p>

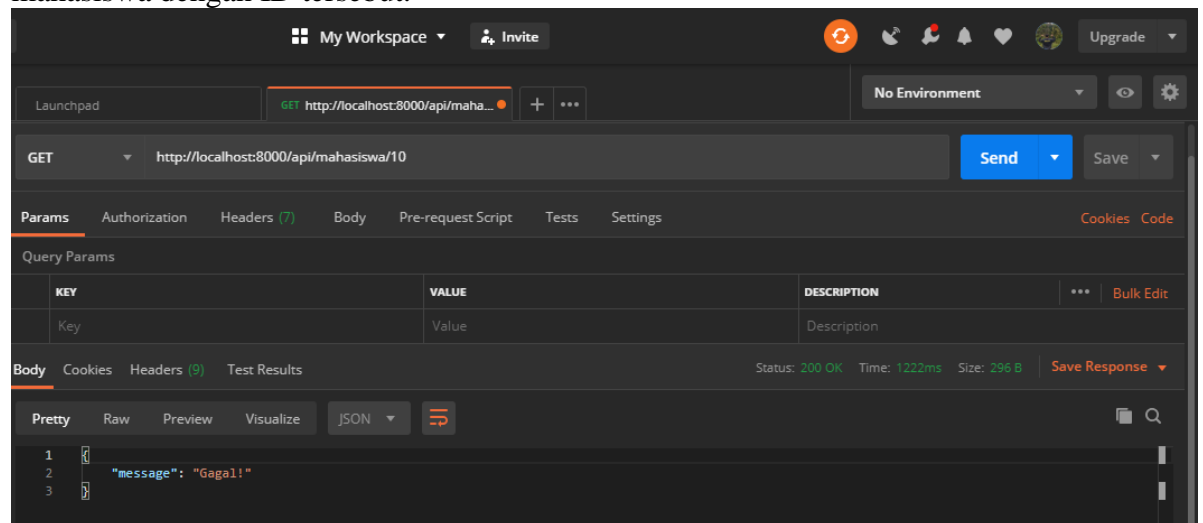
9	<p>Selanjutnya kita akan menambahkan fungsi untuk melihat suatu data ketika dipilih ID tertentu. Buat fungsi baru yaitu <b>getId</b> pada <b>MahasiswaController.php</b>.</p> <p>Keterangan:</p> <ul style="list-style-type: none"> <li>• Fungsi getId menerima parameter \$id yang menunjukkan ID mahasiswa yang dipilih</li> <li>• Line 28 merupakan pemanggilan model untuk membaca data berdasarkan ID</li> </ul> <pre> MahasiswaController.php X laravel-restapi &gt; app &gt; Http &gt; Controllers &gt; MahasiswaController.php 26 //fungsi untuk menampilkan data dari sebuah ID 27 public function getId(\$id) { 28     \$data = Mahasiswa::where('id',\$id)-&gt;get(); 29 30     //cek jika data ditemukan 31     if(count(\$data) &gt; 0) { 32         \$res['message'] = "Success!"; 33         \$res['values'] = \$data; 34         return response(\$res); 35     } 36     //jika data tidak ditemukan 37     else { 38         \$res['message'] = "Gagal!"; 39         return response(\$res); 40     } 41 } 42 </pre>
10	<p>Tambahkan <i>route</i> untuk memanggil fungsi getId pada <b>routes/api.php</b></p> <pre> api.php X laravel-restapi &gt; routes &gt; api.php 23 Route::get('/mahasiswa/{id}', 'MahasiswaController@getId'); </pre> <p>Karena kita ingin menampilkan data, maka perintah yang dipakai adalah 'get'</p>

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***



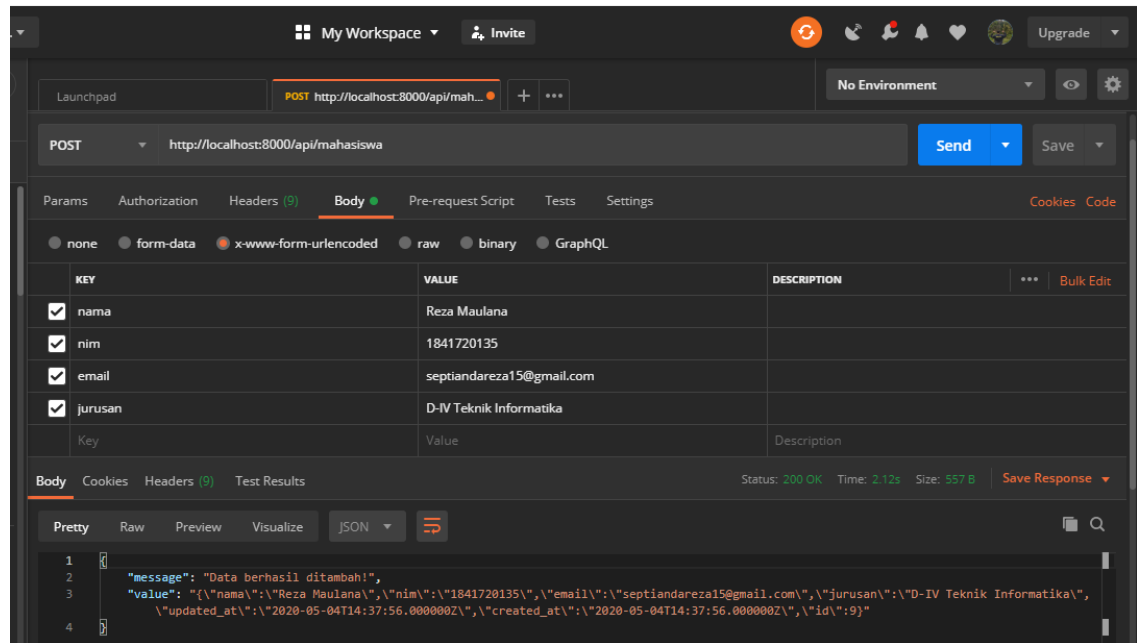
Ketika mencoba menampilkan ID=10 akan muncul pesan “Gagal”, karena tidak ada data mahasiswa dengan ID tersebut.





12	<p>Setelah dapat menampilkan data, kita akan membuat fungsi untuk menambahkan data baru ke database dengan nama <b>create</b> pada <b>MahasiswaController.php</b>.</p> <pre> MahasiswaController.php X laravel-restapi &gt; app &gt; Http &gt; Controllers &gt; MahasiswaController.php  42 //fungsi tambah data 43 public function create(Request \$request) { 44     \$mhs          = new Mahasiswa(); 45     \$mhs-&gt;nama     = \$request-&gt;nama; 46     \$mhs-&gt;nim      = \$request-&gt;nim; 47     \$mhs-&gt;email    = \$request-&gt;email; 48     \$mhs-&gt;jurusan  = \$request-&gt;jurusan; 49 50     //jika data berhasil tersimpan 51     if(\$mhs-&gt;save()) { 52         \$res['message'] = "Data berhasil ditambah!"; 53         \$res['value']   = "\$mhs"; 54         return response(\$res); 55     } 56 } 57 </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> <li>• Fungsi <b>create</b> menerima parameter <b>Request</b> yang menampung isian data mahasiswa yang akan ditambahkan ke database.</li> <li>• Line 51-55 : <code>\$mhs-&gt;save()</code> digunakan untuk menyimpan data ke database, apabila <code>save()</code> berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang ditambah.</li> </ul>
13	<p>Tambahkan <i>route</i> untuk memanggil fungsi <b>create</b> pada <b>routes/api.php</b></p> <pre> api.php X laravel-restapi &gt; routes &gt; api.php  25 Route::post('/mahasiswa', 'MahasiswaController@create');  </pre> <p>Karena kita ingin menambah data, maka perintah yang dipakai adalah 'post'.</p>

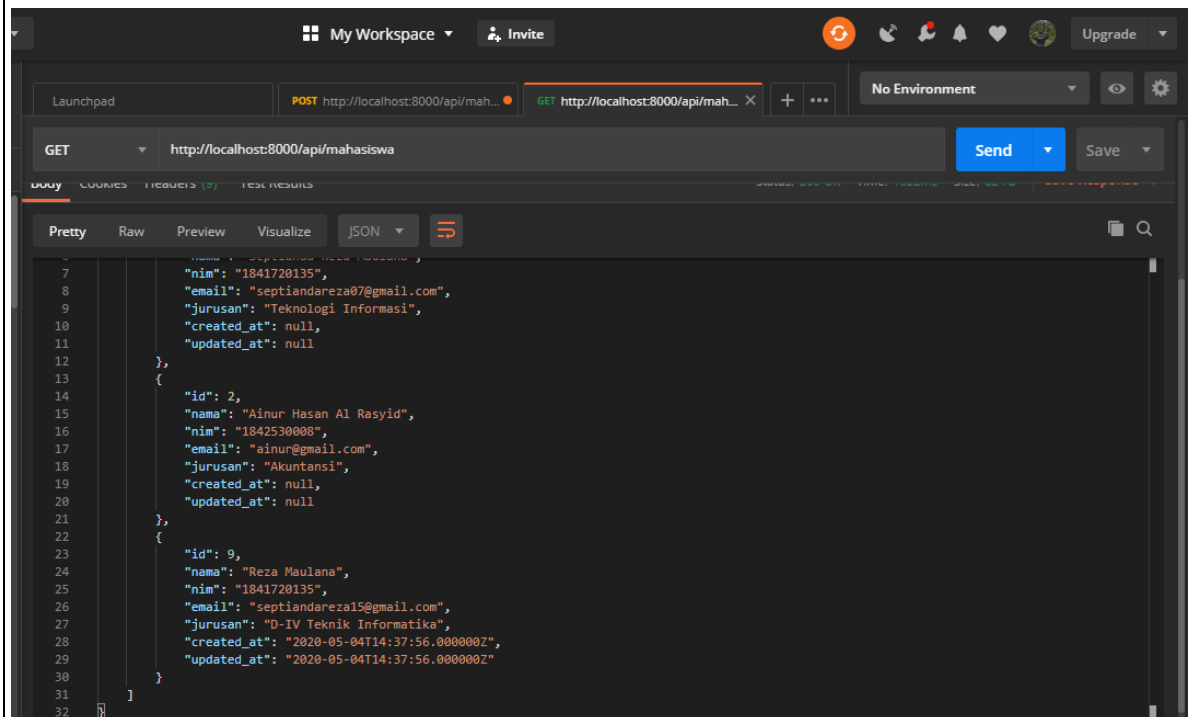
Kita coba untuk menambahkan data melalui Postman.



14

- 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.



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

```
MahasiswaController.php X
laravel-restapi > app > Http > Controllers > MahasiswaController.php

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

15

Keterangan:

- Fungsi **update** menerima parameter **Request** yang menampung isian data mahasiswa yang akan diubah dan parameter **id** yang menunjukkan ID yang dipilih.
- Line 64 : `Mahasiswa::find($id)` digunakan untuk pencarian data pada tabel mahasiswa berdasarkan \$id.
- Line 70-74 : `$mhs->save()` digunakan untuk menyimpan perubahan data ke database, apabila `save()` berhasil dijalankan maka akan ditampilkan pesan berhasil serta data yang diubah.

Tambahkan *route* untuk memanggil fungsi update pada **routes/api.php**

```
api.php X
laravel-restapi > routes > api.php

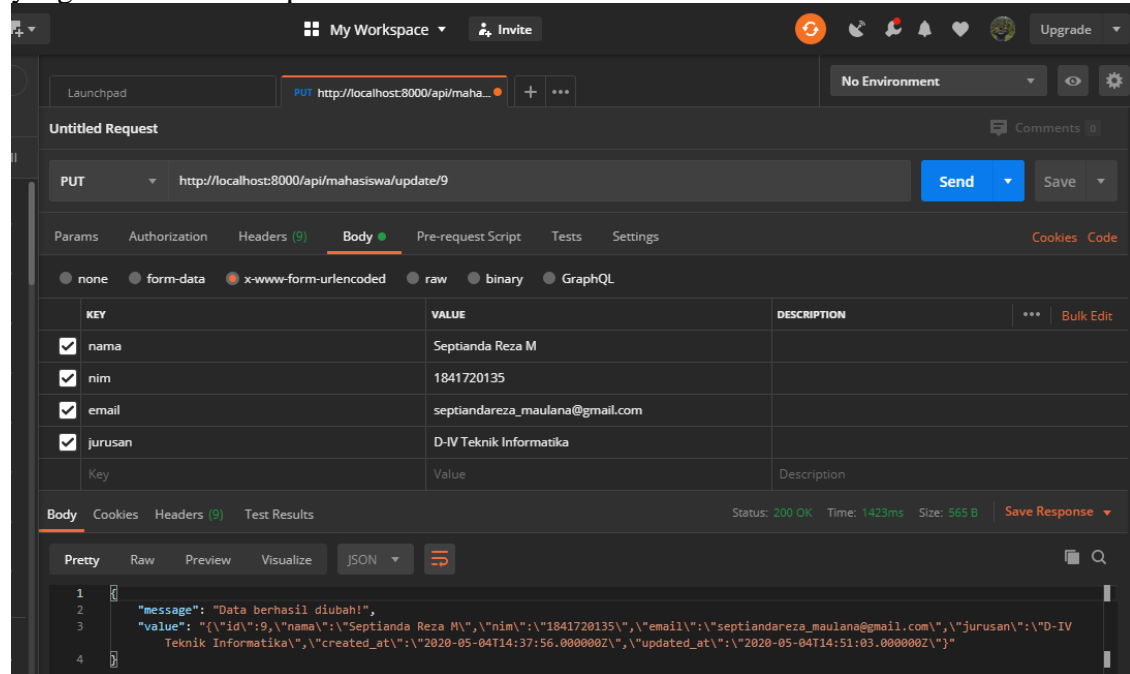
27 Route::put('/mahasiswa/update/{id}', 'MahasiswaController@update');
```

16

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

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

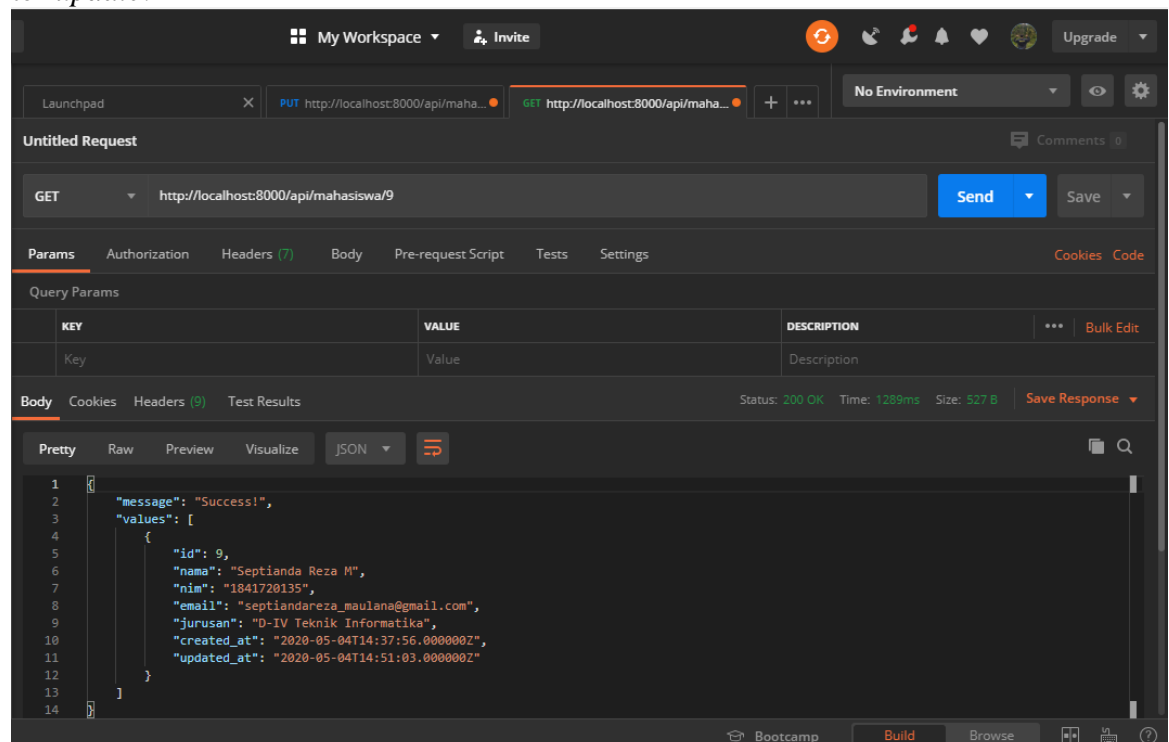
Berikut adalah contoh untuk mengubah data dengan ID=9, maka url diisi : ***http://localhost:8000/api/mahasiswa/update/9***. Pilih tab **Body** dan pilih radio button **x-www-form-urlencoded**. Isikan nama kolom pada database pada **KEY**, untuk isian data yang diubah tuliskan pada **VALUE**.



17

Akan muncul pesan berhasil serta perubahan data dari ID=9.

Kemudian coba untuk menampilkan data dengan ID=9 untuk melihat apakah data sudah ter-update.

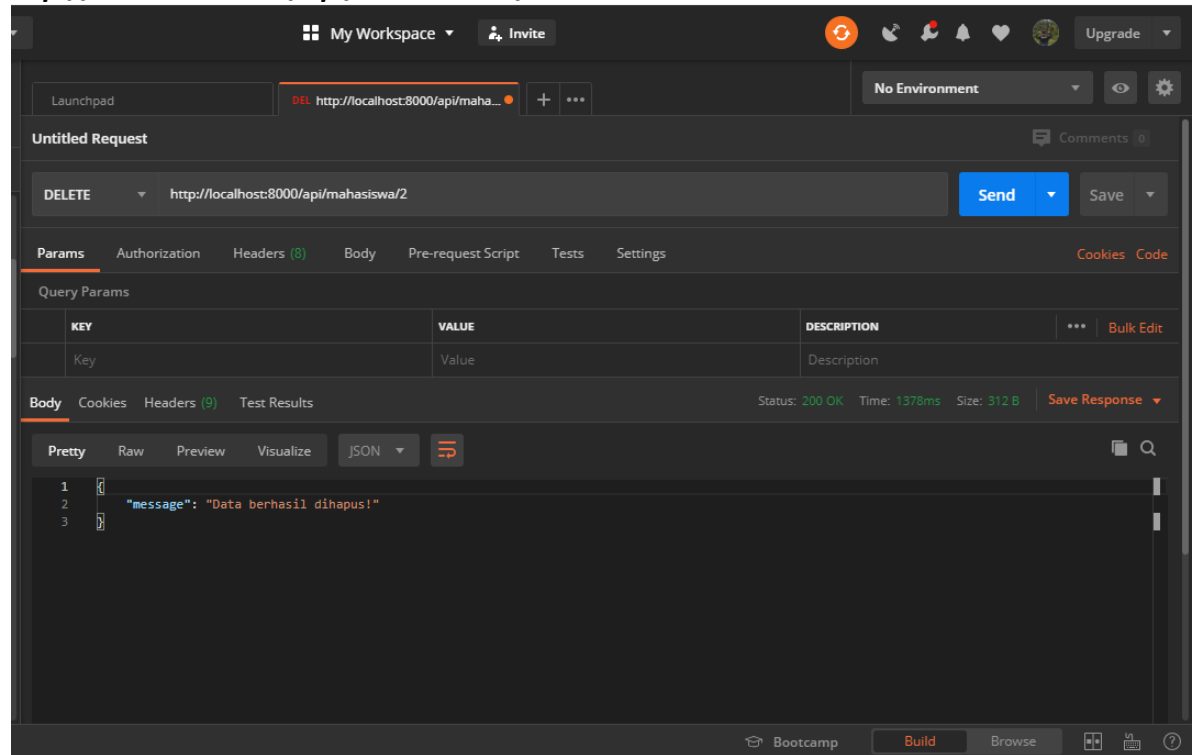


18	<p>Terakhir kita akan membuat fungsi untuk menghapus data dengan nama <b>delete</b> di <b>MahasiswaController.php</b>.</p> <pre> MahasiswaController.php × laravel-restapi &gt; app &gt; Http &gt; Controllers &gt; MahasiswaController.php  79      //fungsi untuk menghapus data 80      public function delete(\$id) { 81          \$mhs = Mahasiswa::where('id', \$id); 82 83          if(\$mhs-&gt;delete()) { 84              \$res['message'] = "Data berhasil dihapus!"; 85              return response(\$res); 86          }else { 87              \$res['message'] = "Gagal!"; 88              return response(\$res); 89          } 90      } 91  </pre> <p>Keterangan:</p> <ul style="list-style-type: none"> <li>• Fungsi <b>delete</b> menerima parameter <b>id</b> yang menunjukkan ID yang dipilih.</li> <li>• Line 83-89 : <code>\$mhs-&gt;delete()</code> digunakan untuk menghapus data dari database, apabila <code>delete()</code> berhasil dijalankan maka akan ditampilkan pesan berhasil.</li> </ul>
19	<p>Tambahkan <i>route</i> untuk memanggil fungsi delete pada <b>routes/api.php</b></p> <pre> api.php × laravel-restapi &gt; routes &gt; api.php 29      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.