

ELOQUENT ORM

1.1 Kompetensi Utama

- Diharapkan mahasiswa dapat memahami Eloquent ORM pada laravel.

1.2 Kompetensi Penunjang

- Mengenalkan kepada mahasiswa tentang Eloquent ORM
- Membuat query SQL menggunakan Eloquent ORM pada laravel

1.3 Dasar Teori

1. Pengertian Eloquent ORM

Eloquent ORM (dari **Object-relational mapping**) adalah cara pengaksesan database dimana setiap baris tabel dianggap sebagai sebuah object. Sebagaimana yang sudah kita pahami, database terdiri dari kumpulan tabel yang saling terhubung. Di dalam setiap tabel, data disimpan dalam bentuk baris dan kolom. ORM dipakai untuk mengubah baris dan kolom ini menjadi sebuah object. Nantinya, setiap kolom akan menjadi property dari object tersebut.

Sebagai bahan praktek, kita kembali menggunakan tabel mahasiswa dan MahasiswaController. Kosongkan tabel mahasiswa kemudian buat daftar Route seperti di bawah ini :

```
routes > 📄 web.php > ...
17
18 Route::get('/', function () {
19     return view('welcome');
20 });
21
22 Route::get('/cek-objek', [MahasiswaController::class, 'cekObjek']);
23
24 Route::get('/insert', [MahasiswaController::class, 'insert']);
25 Route::get('/mass-assignment', [MahasiswaController::class, 'massAssignment']);
26
27
28 Route::get('/update', [MahasiswaController::class, 'update']);
29 Route::get('/update-where', [MahasiswaController::class, 'updateWhere']);
30 Route::get('/mass-update', [MahasiswaController::class, 'massUpdate']);
31
32 Route::get('/delete', [MahasiswaController::class, 'delete']);
33 Route::get('/destroy', [MahasiswaController::class, 'destroy']);
34 Route::get('/mass-delete', [MahasiswaController::class, 'massDelete']);
```

```

35
36 Route::get('/all', [MahasiswaController::class,'all']);
37 Route::get('/all-view', [MahasiswaController::class,'allView']);
38 Route::get('/get-where', [MahasiswaController::class,'getWhere']);
39 Route::get('/test-where', [MahasiswaController::class,'testWhere']);
40 Route::get('/first', [MahasiswaController::class,'first']);
41 Route::get('/find', [MahasiswaController::class,'find']);
42 Route::get('/latest', [MahasiswaController::class,'latest']);
43 Route::get('/limit', [MahasiswaController::class,'limit']);
44 Route::get('/skip-take', [MahasiswaController::class,'skipTake']);
45
46 Route::get('/soft-delete', [MahasiswaController::class,'softDelete']);
47 Route::get('/with-trashed', [MahasiswaController::class,'withTrashed']);
48 Route::get('/restore', [MahasiswaController::class,'restore']);
49 Route::get('/force-delete', [MahasiswaController::class,'forceDelete']);
50

```

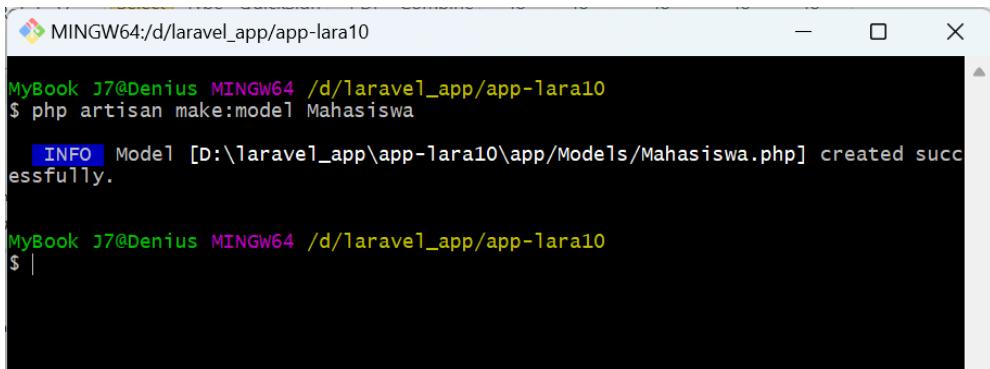
2. Membuat Model

Eloquent ORM memerlukan Model untuk proses konversi data tabel menjadi object. Object inilah yang nantinya akan kita akses dari dalam controller. Sama seperti pembuatan berbagai file di Laravel, tersedia perintah php artisan untuk membuat Model. Berikut format dasar perintah yang digunakan:

```
php artisan make:model <namaModel>
```

Nama model harus berpasangan dengan nama tabel namun dalam bentuk tunggal (*singular*). Jika kita membuat tabel mahasiswas, maka nama modelnya adalah **Mahasiswa**, atau jika memiliki tabel dosen maka nama modelnya adalah **Dosen**. Karena kita sudah memiliki tabel mahasiswas, maka perintah yang diperlukan adalah:

```
php artisan make:model Mahasiswa
```



```

MINGW64:/d/laravel_app/app-lara10
MyBook J7@Denius MINGW64 /d/laravel_app/app-lara10
$ php artisan make:model Mahasiswa
[INFO] Model [D:\laravel_app\app-lara10\app\Models\Mahasiswa.php] created successfully.

MyBook J7@Denius MINGW64 /d/laravel_app/app-lara10
$ |

```

Berikut isi dari file model Mahasiswa.php yang di sudah dibuat :

```
app > Models > Mahasiswa.php > ...
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Mahasiswa extends Model
9 {
10     use HasFactory;
11 }
12
```

3. Mengakses Model

Pembuatan kode program dengan Eloquent ORM tetap dilakukan dari controller. Hanya saja sekarang kita butuh mengakses Model, yakni class Mahasiswa yang ada di file Mahasiswa.php. Sama seperti mengakses Facade, class Mahasiswa ini harus diimport terlebih dahulu ke dalam controller. Caranya, tulis kode berikut di bagian atas file MahasiswaController:

```
use App\Models\Mahasiswa;
```

Dengan demikian, kita sudah bisa mengakses model Mahasiswa sepanjang controller.

```
app > Http > Controllers > MahasiswaController.php > cekObjek
1 <?php
2 namespace App\Http\Controllers;
3
4 use Illuminate\Support\Facades\DB;
5
6 use Illuminate\Http\Request;
7 use App\Models\Mahasiswa;
8
9 class MahasiswaController extends Controller {
10
11 public function cekObjek()
12 {
13     $mahasiswa=new Mahasiswa();
14     dd($mahasiswa);
15 }
16
```

Di baris 7 terdapat tambahan perintah `use App\Models\Mahasiswa` yang dipakai untuk proses import file model Mahasiswa. Kemudian di dalam method `cekObjek()`, kita menginstansiasi class Mahasiswa yang disimpan ke variabel `$mahasiswa`. Variabel `$mahasiswa` ini selanjutnya di `dump()` dengan hasil sebagai berikut:

4. Menginput Data

Proses menginput data menggunakan Eloquent ORM cukup sederhana. Caranya, buat object dari model Mahasiswa, input data ke dalam atribut yang bersesuaian dengan nama kolom tabel, dan terakhir jalankan method `save()`. Berikut kode program untuk input data :

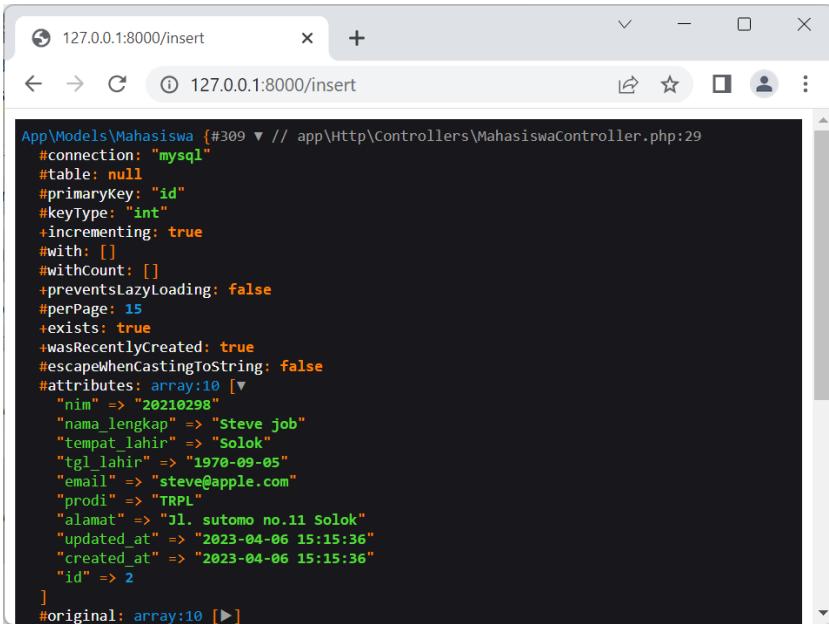
```

app > Http > Controllers > MahasiswaController.php > insert
11 public function cekObjek()
12 {
13     $mahasiswa=new Mahasiswa();
14     dd($mahasiswa);
15 }
16
17 public function insert()
18 {
19     $mahasiswa=new Mahasiswa();
20     $mahasiswa->nim='20210298';
21     $mahasiswa->nama_lengkap='Steve job';
22     $mahasiswa->tempat_lahir='Solok';
23     $mahasiswa->tgl_lahir='1970-09-05';
24     $mahasiswa->email='steve@apple.com';
25     $mahasiswa->prodi='TRPL';
26     $mahasiswa->alamat='Jl. sutomo no.11 Solok';
27     $mahasiswa->save();
28
29     dd($mahasiswa);
30 }

```

Setelah proses instansiasi class Mahasiswa ke dalam object \$mahasiswa baris 19, Kemudian di baris 20-26 kita mengisi beberapa property yang sama dengan nama kolom tabel. Tidak semua kolom harus diisi, tergantung kebutuhan dan selama sesuai dengan syarat di tabel. Pada baris 27, method \$mahasiswa->save() dipakai untuk menyimpan data-data ini ke dalam database. Dan terakhir object \$mahasiswa kita dd() untuk melihat isinya.

Jalankan method di atas dengan membuka alamat localhost:8000/insert. Dari hasil tampilan dd(), silahkan buka `#attributes`:



```

App\Models\Mahasiswa {#309 ▾ // app\Http\Controllers\MahasiswaController.php:29
  #connection: "mysql"
  #table: null
  #primaryKey: "id"
  #keyType: "int"
  +incrementing: true
  #with: []
  #withCount: []
  +preventsLazyLoading: false
  #perPage: 15
  +exists: true
  +wasRecentlyCreated: true
  #escapeWhenCastingToString: false
  #attributes: array:10 [▼
    "nim" => "20210298"
    "nama_lengkap" => "Steve job"
    "tempat_lahir" => "Solok"
    "tgl_lahir" => "1970-09-05"
    "email" => "steve@apple.com"
    "prodi" => "TRPL"
    "alamat" => "Jl. sutomo no.11 Solok"
    "updated_at" => "2023-04-06 15:15:36"
    "created_at" => "2023-04-06 15:15:36"
    "id" => 2
  ]
  #original: array:10 [▶]
}

```

The screenshot shows the MySQL Workbench interface with the 'mahasiswa' table selected. The table has columns: id, nim, nama_lengkap, tempat_lahir, tgl_lahir, email, prodi, alamat, created_at, and updated_at. Two rows of data are displayed:

	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at
1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-04-05 08:52:41	2023-04-05 08:52:41	
2	20210298	Steve job	Solok	1970-09-05	steve@apple.com	TRPL	Jl. sutomo no.11 Solok	2023-04-06 15:15:36	2023-04-06 15:15:36	

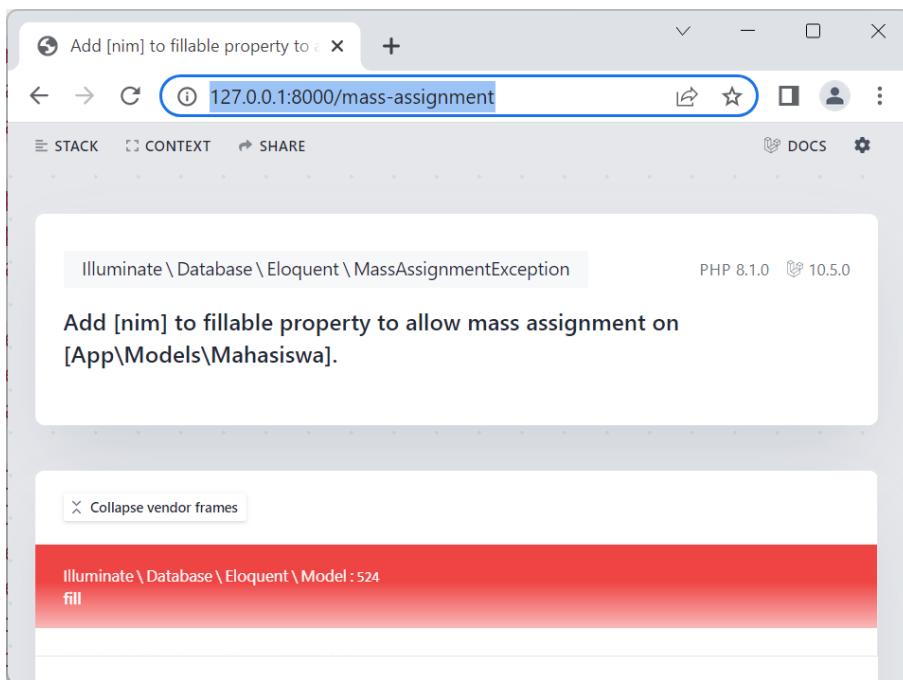
5. Input Mass Assignment

Cara lain untuk proses input menggunakan Eloquent adalah dengan teknik **mass assignment**. Disebut seperti ini karena kita bisa mengisi banyak property untuk object Mahasiswa dalam sekali proses. Cara input *mass assignment* dilakukan dengan mengakses static method `create()` dari Model object. Karena kita menggunakan model Mahasiswa, maka pemanggilan method ini adalah dengan perintah `Mahasiswa::create()`. Data yang akan diinput ditulis sebagai associative array dan menjadi argument dari method `Mahasiswa::create()`. Berikut contoh penggunaannya:

```
app > Http > Controllers > MahasiswaController.php > MahasiswaController
31
32 public function massAssignment()
33 {
34     $mahasiswa=Mahasiswa::create(
35         [
36             'nim'=>'202007890',
37             'nama_lengkap'=>'M. Yazid',
38             'tempat_lahir'=>'Padang',
39             'tgl_lahir'=>'2011-07-03',
40             'email'=>'yazid@gmail.com',
41             'prodi'=>'MI',
42             'alamat'=>'Padang',
43         ]
44     );
45     dump($mahasiswa);
46     $mahasiswa1=Mahasiswa::create(
47         [
48             'nim'=>'202007891',
49             'nama_lengkap'=>'M. Rasyid',
50             'tempat_lahir'=>'Padang',
51             'tgl_lahir'=>'2015-05-12',
52             'email'=>'rasyid@gmail.com',
53             'prodi'=>'TRPL',
54             'alamat'=>'Padang',
55         ]
56     );
57     dump($mahasiswa1);
58 }
```

Associative array yang ada di baris 36 – 42 inilah yang menjadi alasan kenapa disebut sebagai *mass assignment*. Karena seharusnya kita men-set satu per satu nilai ini ke dalam property object Mahasiswa:

Kembali ke proses input menggunakan *mass assignment*, jalankan dengan membuka halaman `localhost:8000/mass-assignment`:



Hasilnya tampil pesan error dengan keterangan *Add [nim] to fillable property to allow mass assignment on [App\Models\Mahasiswa]*. Error ini terjadi karena Laravel membatasi akses ke sebuah tabel ketika di proses menggunakan mass assignment. Pembatasan ini dibuat karena mass assignment sering dipakai dengan nilai yang langsung berasal dari form. Sehingga untuk mencegah kemungkinan terjadi 'manipulasi data', Laravel mewajibkan programmer untuk menulis nama kolom apa saja yang boleh diinput.

Dalam contoh ini, kita harus mendaftarkan kolom 'nim', 'nama_lengkap', 'tanggal_lahir' dst ke dalam property `$fillable` yang ada di model Mahasiswa.

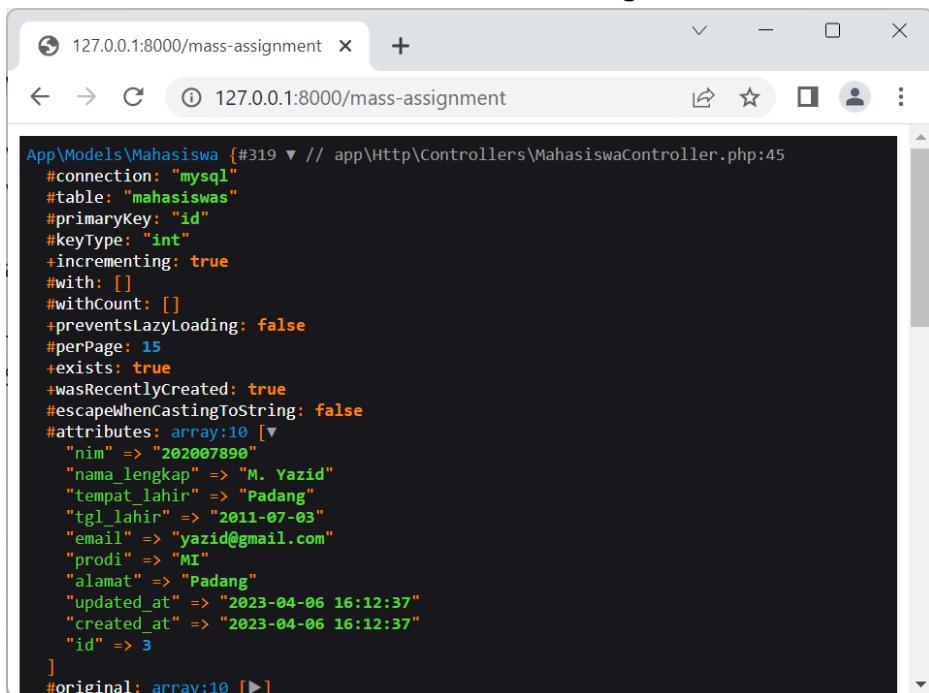
Property ini harus kita tambah manual. Silahkan buka file model Mahasiswa.php, lalu tambah 1 baris berikut ke dalam class Mahasiswa:

```
protected $fillable = ['nim', 'nama_lengkap',  
'tempat_lahir', 'tgl_lahir', 'email', 'prodi', 'alamat'];
```

Berikut kode akhir dari file model Mahasiswa.php:

```
app > Models > Mahasiswa.php > Mahasiswa > $fillable
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Mahasiswa extends Model
9 {
10     use HasFactory;
11     protected $fillable=['nim','nama_lengkap','tempat_lahir','tgl_lahir','email','prodi','alamat'];
12 }
13
```

Jalankan kembali `localhost:8000/mass-assignment`



Ketika dijalankan, tidak ada error lagi. Untuk memastikan kita cek ke database:

Server: 127.0.0.1 » Database: app_lara10 » Table: mahasiswa

[Browse](#) [Structure](#) [SQL](#) [Search](#) [Insert](#) [Export](#) [Import](#) [Privileges](#) [Operations](#) [Triggers](#)

Showing rows 0 - 3 (4 total, Query took 0.0002 seconds.)

```
SELECT * FROM `mahasiswa`
```

Profiling [Edit inline](#) [Edit](#) [Explain SQL](#) [Create PHP code](#) [Refresh](#)

<input type="checkbox"/> Show all	Number of rows:	25	Filter rows:	Search this table	Sort by key:	None																																																																							
+ Options <table border="1"> <thead> <tr> <th></th> <th>Edit</th> <th>Copy</th> <th>Delete</th> <th>id</th> <th>nim</th> <th>nama_lengkap</th> <th>tempat_lahir</th> <th>tgl_lahir</th> <th>email</th> <th>prodi</th> <th>alamat</th> <th>created_at</th> <th>updated_at</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>Edit</td> <td>Copy</td> <td>Delete</td> <td>1</td> <td>2022090909</td> <td>Linus B Torvalds</td> <td>Finlandia</td> <td>1971-08-12</td> <td>linus@linux.org</td> <td>TRPL</td> <td>Jl. Sudirman no.10 Padang</td> <td>2023-04-05 08:52:41</td> <td>2023-04-05 08:52:41</td> </tr> <tr> <td><input type="checkbox"/></td> <td>Edit</td> <td>Copy</td> <td>Delete</td> <td>2</td> <td>20210298</td> <td>Steve job</td> <td>Solok</td> <td>1970-09-05</td> <td>steve@apple.com</td> <td>TRPL</td> <td>Jl. sutomo no.11 Solok</td> <td>2023-04-06 15:15:36</td> <td>2023-04-06 15:15:36</td> </tr> <tr> <td><input type="checkbox"/></td> <td>Edit</td> <td>Copy</td> <td>Delete</td> <td>3</td> <td>202007890</td> <td>M. Yazid</td> <td>Padang</td> <td>2011-07-03</td> <td>yazid@gmail.com</td> <td>MI</td> <td>Padang</td> <td>2023-04-06 16:12:37</td> <td>2023-04-06 16:12:37</td> </tr> <tr> <td><input type="checkbox"/></td> <td>Edit</td> <td>Copy</td> <td>Delete</td> <td>4</td> <td>202007891</td> <td>M. Rasyid</td> <td>Padang</td> <td>2015-05-12</td> <td>rasyid@gmail.com</td> <td>TRPL</td> <td>Padang</td> <td>2023-04-06 16:12:37</td> <td>2023-04-06 16:12:37</td> </tr> </tbody> </table>									Edit	Copy	Delete	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at	<input type="checkbox"/>	Edit	Copy	Delete	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-04-05 08:52:41	2023-04-05 08:52:41	<input type="checkbox"/>	Edit	Copy	Delete	2	20210298	Steve job	Solok	1970-09-05	steve@apple.com	TRPL	Jl. sutomo no.11 Solok	2023-04-06 15:15:36	2023-04-06 15:15:36	<input type="checkbox"/>	Edit	Copy	Delete	3	202007890	M. Yazid	Padang	2011-07-03	yazid@gmail.com	MI	Padang	2023-04-06 16:12:37	2023-04-06 16:12:37	<input type="checkbox"/>	Edit	Copy	Delete	4	202007891	M. Rasyid	Padang	2015-05-12	rasyid@gmail.com	TRPL	Padang	2023-04-06 16:12:37	2023-04-06 16:12:37
	Edit	Copy	Delete	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at																																																																
<input type="checkbox"/>	Edit	Copy	Delete	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-04-05 08:52:41	2023-04-05 08:52:41																																																																
<input type="checkbox"/>	Edit	Copy	Delete	2	20210298	Steve job	Solok	1970-09-05	steve@apple.com	TRPL	Jl. sutomo no.11 Solok	2023-04-06 15:15:36	2023-04-06 15:15:36																																																																
<input type="checkbox"/>	Edit	Copy	Delete	3	202007890	M. Yazid	Padang	2011-07-03	yazid@gmail.com	MI	Padang	2023-04-06 16:12:37	2023-04-06 16:12:37																																																																
<input type="checkbox"/>	Edit	Copy	Delete	4	202007891	M. Rasyid	Padang	2015-05-12	rasyid@gmail.com	TRPL	Padang	2023-04-06 16:12:37	2023-04-06 16:12:37																																																																
↑ <input type="checkbox"/> Check all With selected: Edit Copy Delete Export																																																																													
<input type="checkbox"/> Show all	Number of rows:	25	Filter rows:	Search this table	Sort by key:	None																																																																							

Data berhasil ditambahkan ke database. Cara lain untuk mengizinkan nama kolom bisa diakses dari *mass assignment* adalah menggunakan property **\$guarded**. Berbeda dengan **\$fillable**, **\$guarded** dipakai untuk menulis nama kolom apa saja yang **tidak boleh diisi**. Sebagai contoh, jika kita memutuskan kolom prodi tidak boleh diisi menggunakan mass assignment, bisa menulis kode berikut:

```
protected $guarded = ['prodi'];
```

Akibatnya, pada saat proses input dilakukan, nilai untuk kolom prodi akan terhalang dan tidak bisa sampai ke database meskipun sudah tertulis dalam controller. Salah satu teknik yang sering dipakai adalah mengisi array kosong ke dalam **\$guarded**:

```
protected $guarded = [];
```

```
app > Models > Mahasiswa.php > Mahasiswa
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Mahasiswa extends Model
9 {
10     use HasFactory;
11     protected $guarded=[];
12 }
13
```

6. Update Data

Eloquent ORM memproses data tabel menggunakan object. Untuk proses update, caranya adalah kita harus cari Model object dari tabel, lalu ubah beberapa property dan simpan kembali. Berikut contoh prakteknya:

```
app > Http > Controllers > MahasiswaController.php > update
52     'prodi'=>'TRPL',
53     'alamat'=>'Padang',
54 ];
55 );
56 dump($mahasiswa1);
57 }
58
59 public function update()
60 {
61     $mahasiswa=Mahasiswa::find(2);
62
63     $mahasiswa->tempat_lahir='California';
64     $mahasiswa->prodi='Tekom';
65     $mahasiswa->save();
66
67
68 dd($mahasiswa);
69 }
```

Di baris 62 terdapat perintah `Mahasiswa::find(2)`. Method `find()` dipakai untuk mencari data tabel berdasarkan kolom `id`. Artinya, perintah `Mahasiswa::find(2)` akan mengambil data di tabel mahasiswa yang memiliki `id = 2`, lalu membuat object dari data tersebut. Object ini selanjutnya disimpan ke dalam variabel `$mahasiswa`. Kemudian di baris 64 dan 65, kita mengisi data ke dalam

`$mahasiswa->tempat_lahir` dan ke dalam `$mahasiswa->prodi`.

Kedua nilai ini akan menimpa data yang ada sebelumnya, yakni data dari mahasiswa dengan `id = 2`. Agar perubahan tersimpan ke database, akhiri dengan perintah `$mahasiswa->save()`.

Jalankan kode ini dengan mengakses halaman `localhost:8000/update`:

```
#with: []
#withCount: []
+preventsLazyLoading: false
#perPage: 15
+exists: true
+wasRecentlyCreated: false
#escapeWhenCastingToString: false
#attributes: array:10 [▼
    "id" => 2
    "nim" => "20210298"
    "nama_lengkap" => "Steve job"
    "tempat_lahir" => "California"
    "tgl_lahir" => "1970-09-05"
    "email" => "steve@apple.com"
    "prodi" => "Tekom"
    "alamat" => "Jl. sutomo no.11 Solok"
    "created_at" => "2023-04-06 15:15:36"
    "updated_at" => "2023-04-06 16:35:09"
]
#original: array:10 [▶]
#changes: array:3 [▶]
```

	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at			
<input type="checkbox"/>	Edit	Copy	Delete	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-04-05 08:52:41	2023-04-05 08:52:41
<input type="checkbox"/>	Edit	Copy	Delete	2	20210298	Steve job	California	1970-09-05	steve@apple.com	Tekom	Jl. sutomo no.11 Solok	2023-04-06 15:15:36	2023-04-06 16:35:09
<input type="checkbox"/>	Edit	Copy	Delete	3	202007890	M. Yazid	Padang	2011-07-03	yazid@gmail.com	MI	Padang	2023-04-06 16:12:37	2023-04-06 16:12:37
<input type="checkbox"/>	Edit	Copy	Delete	4	202007891	M. Rasyid	Padang	2015-05-12	rasyid@gmail.com	TRPL	Padang	2023-04-06 16:12:37	2023-04-06 16:12:37

Ok, perubahan dari proses update sudah sampai ke database.

7. Update Data Menggunakan Where

Selain menggunakan method find(), terdapat berbagai method lain yang bisa dipakai untuk mencari data tabel. Mayoritas method ini mirip seperti yang kita pakai di bab

Query Builder. Salah satunya adalah method where():

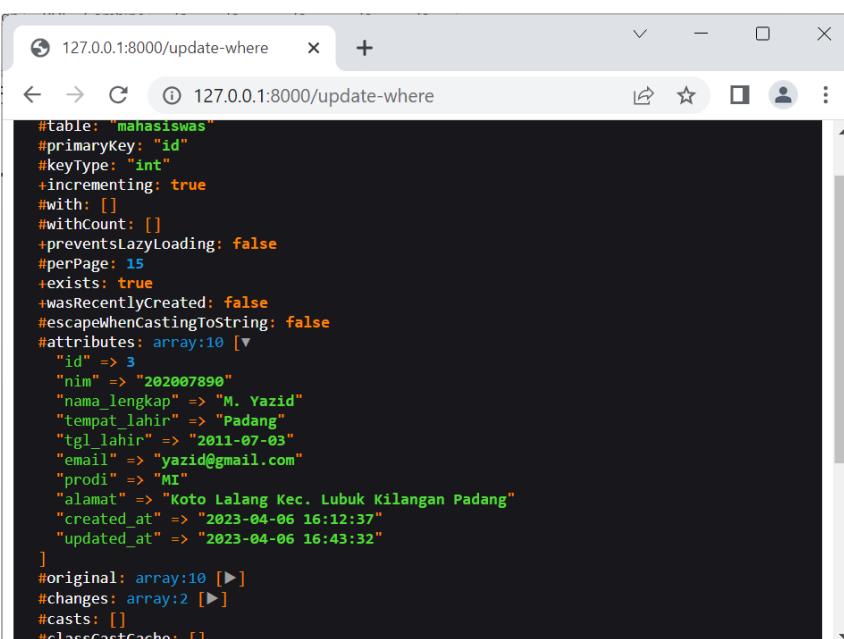
```
app > Http > Controllers > MahasiswaController.php > updateWhere()
66     $mahasiswa->save();
67
68     dd($mahasiswa);
69 }
70
71 public function updateWhere()
72 {
73     $mahasiswa=Mahasiswa::where('nim','202007890')->first();
74     $mahasiswa->alamat='Koto Lalang Kec. Lubuk Kilangan Padang';
75     $mahasiswa->save();
76
77 }
78
79
```

Di baris 73 kita menjalankan method

`Mahasiswa::where('nim','202007890')->first();`

Perintah ini akan mencari mahasiswa dengan kolom nim yang berisi nilai 202007890.

Tambahan method first() perlu ditulis karena method where() akan menghasilkan sebuah collection. Setelah object di dapat, update alamat mahasiswa tersebut dan simpan kembali dengan method save(). Jalankan dengan mengakses halaman `localhost:8000/update-where`:



```
#table: "mahasiswa"
#primaryKey: "id"
#keyType: "int"
+incrementing: true
#with: []
#withCount: []
+preventsLazyLoading: false
#perPage: 15
+exists: true
+wasRecentlyCreated: false
#escapeWhenCastingToString: false
#attributes: array:10 [▼
    "id" => 3
    "nim" => "202007890"
    "nama_lengkap" => "M. Yazid"
    "tempat_lahir" => "Padang"
    "tgl_lahir" => "2011-07-03"
    "email" => "yazid@gmail.com"
    "prodi" => "MI"
    "alamat" => "Koto Lalang Kec. Lubuk Kilangan Padang"
    "created_at" => "2023-04-06 16:12:37"
    "updated_at" => "2023-04-06 16:43:32"
]
#original: array:10 [▶]
#changes: array:2 [▶]
#casts: []
#classCastCache: []
```

The screenshot shows the MySQL Workbench interface with the 'mahasiswa' table selected. The table has columns: id, nim, nama_lengkap, tempat_lahir, tgl_lahir, email, prodi, alamat, created_at, and updated_at. The data is as follows:

	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at
1	Edit Copy Delete	202209009	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-04-05 08:52:41	2023-04-05 08:52:41
2	Edit Copy Delete	20210298	Steve job	California	1970-09-05	steve@apple.com	Tekom	Jl. sutomo no.11 Solok	2023-04-06 15:15:36	2023-04-06 16:35:09
3	Edit Copy Delete	202007890	M. Yazid	Padang	2011-07-03	yazid@gmail.com	MI	Koto Lalang Kec. Lubuk Kilangan Padang	2023-04-06 16:12:37	2023-04-06 16:43:32
4	Edit Copy Delete	202007891	M. Rasyid	Padang	2015-05-12	rasyid@gmail.com	TRPL	Padang	2023-04-06 16:12:37	2023-04-06 16:12:37

8. Mass Update

Proses update juga bisa dilakukan menggunakan teknik *mass update*. Caranya mirip seperti *mass assignment*, tapi kali ini kita menggunakan method `update()`:

```
app > Http > Controllers > MahasiswaController.php > massUpdate
72 {
73     $mahasiswa=Mahasiswa::where('nim','202007890')->first();
74     $mahasiswa->alamat='Koto Lalang Kec. Lubuk Kilangan Padang';
75     $mahasiswa->save();
76
77     dd($mahasiswa);
78 }
79
80 public function massUpdate()
81 {
82     $mahasiswa=Mahasiswa::where('nim','202007890')->first()->update(
83     [
84         'tempat_lahir'=>'Payakumbuh',
85         'prodi'=>'Teknik Komputer'
86     ]);
87
88     dd($mahasiswa);
89 }
90
```

Di baris 82 kembali menggunakan method

`Mahasiswa::where('nim','19003036')->first()` untuk mencari mahasiswa yang ingin di update, kemudian langsung di-chain dengan method `update()` yang berisi associative array dari data yang akan di ubah.

Jalankan method `massUpdate()` ini dengan membuka localhost:8000/mass-update. Hasilnya, kolom `tempat_lahir`, `prodi` dan `updated_at` sudah berubah:

	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-04-05 08:52:41	2023-04-05 08:52:41
	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	2	20210298	Steve job	California	1970-09-05	steve@apple.com	Tekom	Jl. sutomo no.11 Solok	2023-04-06 15:15:36	2023-04-06 16:35:09
	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	3	202007890	M. Yazid	Payakumbuh	2011-07-03	yazid@gmail.com	Teknik Komputer	Koto Lalang Kec. Lubuk Kilangan Padang	2023-04-06 16:12:37	2023-04-06 16:54:01
	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	4	202007891	M. Rasyid	Padang	2015-05-12	rasyid@gmail.com	TRPL	Padang	2023-04-06 16:12:37	2023-04-06 16:12:37

9. Menghapus Data menggunakan Method delete()

Proses menghapus data dengan Eloquent sangat *simple*, cukup dengan mengakses method `delete()` dari Model object. Namun sebelum itu kita harus cari terlebih dahulu Model object yang ingin dihapus. Berikut contoh menghapus data menggunakan Eloquent ORM:

```

89 }
90
91 public function delete()
92 {
93     $mahasiswa=Mahasiswa::find(4);
94     $mahasiswa->delete();
95
96     dd($mahasiswa);
97 }
98
--
```

Di baris 93 kita gunakan method `find(4)` untuk mencari data mahasiswa yang memiliki `id=4`. Setelah object di dapat, tinggal jalankan method `delete()`.

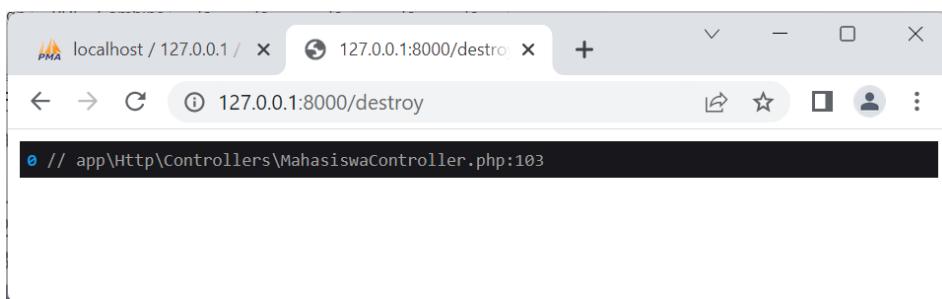
Jalankan kode di atas dengan membuka halaman `localhost:8000/delete`. Hasilnya, data Mahasiswa dengan `id=4` sudah tidak ada lagi.

	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-04-05 08:52:41	2023-04-05 08:52:41
	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	2	20210298	Steve job	California	1970-09-05	steve@apple.com	Tekom	Jl. sutomo no.11 Solok	2023-04-06 15:15:36	2023-04-06 16:35:09
	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	3	202007890	M. Yazid	Payakumbuh	2011-07-03	yazid@gmail.com	Teknik Komputer	Koto Lalang Kec. Lubuk Kilangan Padang	2023-04-06 16:12:37	2023-04-06 16:54:01

10. Menghapus Data menggunakan destroy()

Untuk menghapus data juga bisa menggunakan *static method* Model::destroy(). Method destroy() ini butuh sebuah argument berupa id dari data yang ingin dihapus:

```
90
91 public function delete()
92 {
93     $mahasiswa=Mahasiswa::find(4);
94     $mahasiswa->delete();
95
96     dd($mahasiswa);
97 }
98
99 public function destroy()
100 {
101     $mahasiswa=Mahasiswa::destroy(4);
102
103     dd($mahasiswa);
104 }
```



Perintah Mahasiswa::destroy(4) akan menghapus data dari tabel mahasiswas yang memiliki kolom id = 4. Jika tidak ditemukan, method ini mengembalikan nilai 0 dan tidak terjadi error. Method destroy() juga bisa diisi dengan array atau *collection* yang terdiri dari kumpulan id seperti contoh berikut:

```
Mahasiswa::destroy([3, 9, 10]);
Mahasiswa::destroy(collect([3, 9, 10]));
```

Jika method di atas dijalankan, maka data id 3, 9 dan 10 akan dihapus dari tabel mahasiswas.

11. Menghapus Data Menggunakan Mass Delete

Mass delete adalah sebutan untuk cara menghapus data dari kumpulan Model object. Kumpulan model object ini didapat dari hasil pencarian method where() seperti contoh berikut:

```

103     |     dd($mahasiswa);
104 }
105
106 public function massDelete()
107 {
108     $mahasiswa=Mahasiswa::where('prodi','Teknik Komputer')->delete();
109
110    dd($mahasiswa);
111 }
112

```

Di baris 108 kita menjalankan method `Mahasiswa::where('prodi','Teknik Komputer')` yang dipakai untuk mencari semua mahasiswa dengan prodi=Teknik Komputer. Untuk semua data ini, langsung di *chaining* dengan method `delete()`. Hasilnya, setelah halaman `localhost:8000/mass-delete` di akses, isi tabel mahasiswas akan Menghapus semua mahasiswa dengan prodi=Teknik Komputer.

	Edit	Copy	Delete	1	2022090909	Linus B Torvalds	Finlandia	1971-08-12	linus@linux.org	TRPL	Jl. Sudirman no.10 Padang	2023-04-05 08:52:41	2023-04-05 08:52:41
	Edit	Copy	Delete	2	20210298	Steve job	California	1970-09-05	steve@apple.com	Tekom	Jl. sutomo no.11 Solok	2023-04-06 15:15:36	2023-04-06 16:35:09
Check all	Edit	Copy	Delete							Export			

12. Menampilkan Data dengan Method All()

Method `all()`ini berfungsi untuk mengambil semua data dari dalam tabel:

```

111 }
112
113 public function all()
114 {
115     $mahasiswa=Mahasiswa::all();
116
117     dd($mahasiswa);
118 }
119

```

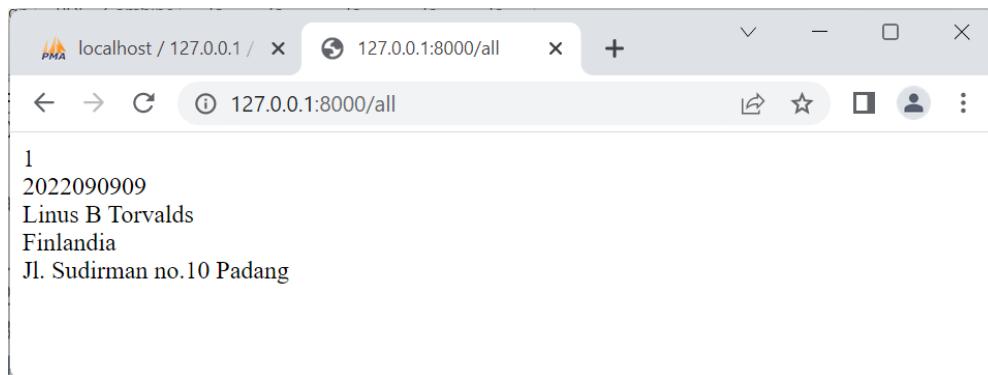
```

Illuminate\Database\Eloquent\Collection {#657 ▼ // app\Http\Controllers\MahasiswaController.php:117
#items: array:2 [▼
  0 => App\Models\Mahasiswa {#1272 ▶}
  1 => App\Models\Mahasiswa {#1273 ▶}
]
#escapeWhenCastingToString: false
}

```

Sedangkan untuk menampilkan data dari 1 mahasiswa hasil dari method all(), sebagai berikut :

```
105
106 public function massDelete()
107 {
108     $mahasiswa=Mahasiswa::where('prodi','Teknik Komputer')->delete();
109
110     dd($mahasiswa);
111 }
112
113 public function all()
114 {
115     $mahasiswa=Mahasiswa::all();
116     echo $mahasiswa[0]->id . '<br>';
117     echo $mahasiswa[0]->nim . '<br>';
118     echo $mahasiswa[0]->nama_lengkap . '<br>';
119     echo $mahasiswa[0]->tempat_lahir . '<br>';
120     echo $mahasiswa[0]->alamat;
121     //dd($mahasiswa);
122 }
```



Dan untuk menampilkan semua data menggunakan foreach..

```
app > Http > Controllers > MahasiswaController.php > MahasiswaController > all
105
106 public function massDelete()
107 {
108     $mahasiswa=Mahasiswa::where('prodi','Teknik Komputer')->delete();
109
110     dd($mahasiswa);
111 }
112
113 public function all()
114 {
115     $mahasiswa=Mahasiswa::all();
116     foreach ($mahasiswa as $mhs) {
117         echo $mhs->id . '<br>';
118         echo $mhs->nim . '<br>';
119         echo $mhs->nama_lengkap . '<br>';
120         echo $mhs->tempat_lahir . '<br>';
121         echo $mhs->alamat;
122         echo '<br>';
123     }
124 }
```

```

1
2022090909
Linus B Torvalds
Finlandia
Jl. Sudirman no.10 Padang

2
20210298
Steve job
California
Jl. sutomo no.11 Solok

```

Untuk mengirim data ke view

```

app > Http > Controllers > MahasiswaController.php > MahasiswaController > all
119     echo $mhs->nim . "<br>";
120     echo $mhs->nama_lengkap . '<br>';
121     echo $mhs->tempat_lahir . '<br>';
122     echo $mhs->alamat;
123     echo '<hr>';
124 }
125
126
127 public function allView()
128 {
129     $mahasiswas=Mahasiswa::all();
130     return view('akademik.mahasiswa',[ 'mahasiswas'=>$mahasiswas]);
131 }
132
133

```

No	Nim	Nama	Email	Alamat
1	2022090909	Linus B Torvalds	linus@linux.org	Jl. Sudirman no.10 Padang
2	20210298	Steve job	steve@apple.com	Jl. sutomo no.11 Solok

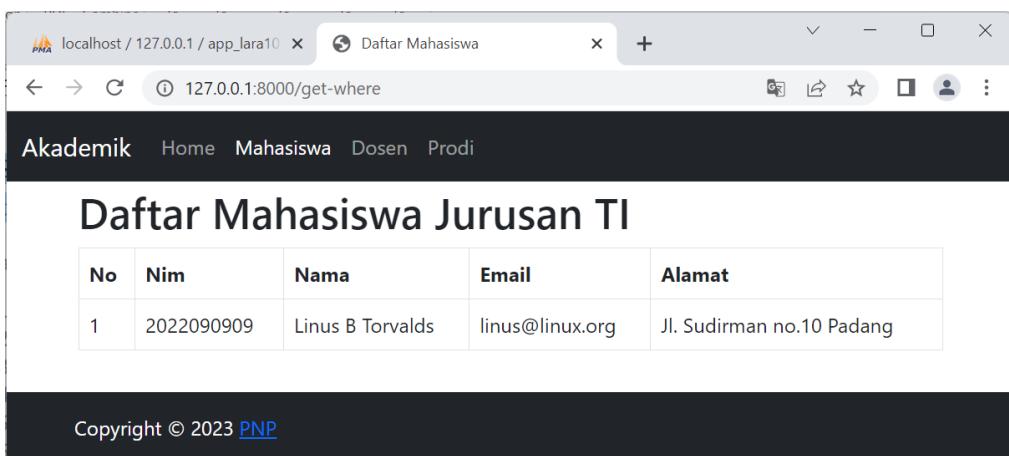
Copyright © 2023 PNP

13. Menampilkan Data dengan Method Where()

Untuk mengambil sebagian data dari tabel, kita bisa memakai method where(). Hasil dari method ini bisa di *chaining* untuk kombinasi pembatasan lebih lanjut.

Sebagai contoh, kita ingin menampilkan semua mahasiswa dengan prodi='TRPL', dan diurutkan berdasarkan nama secara asc. Berikut kode programnya:

```
132
133 public function getWhere()
134 {
135     $mahasiswa=Mahasiswa::where('prodi','TRPL')
136         ->orderBy('nama_lengkap','asc')
137         ->get();
138     return view('akademik.mahasiswa',['mahasiswa'=>$mahasiswa]);
139 }
140
```



A screenshot of a web browser window titled "Daftar Mahasiswa". The URL in the address bar is "localhost / 127.0.0.1 / app_lara10" and the page title is "Daftar Mahasiswa". Below the title, there is a navigation bar with links: Akademik, Home, Mahasiswa, Dosen, Prodi. The main content area has a heading "Daftar Mahasiswa Jurusan TI". Below the heading is a table with the following data:

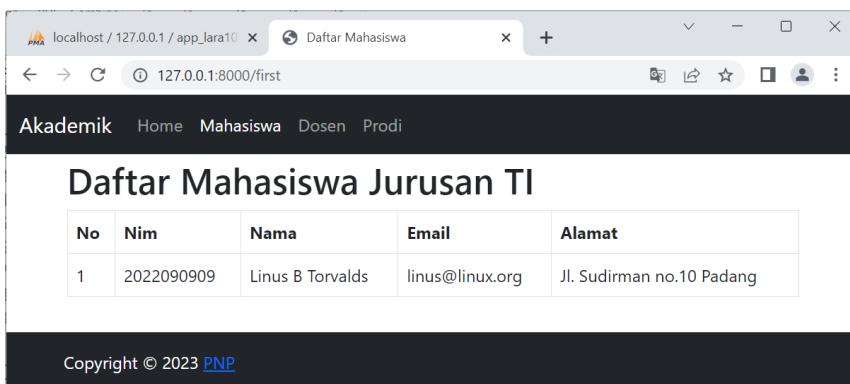
No	Nim	Nama	Email	Alamat
1	2022090909	Linus B Torvalds	linus@linux.org	Jl. Sudirman no.10 Padang

At the bottom of the page, there is a dark footer bar with the text "Copyright © 2023 PNP".

14. Menampilkan Data dengan method First()

Jika yang kita inginkan adalah 1 object saja, bisa memakai method first() sebagai penganti get():

```
139 }
140
141 public function first()
142 {
143     $mahasiswa=Mahasiswa::where('prodi','TRPL')->first();
144     return view('akademik.mahasiswa',['mahasiswa'=>$mahasiswa]);
145 }
```



A screenshot of a web browser window titled "Daftar Mahasiswa". The URL in the address bar is "localhost / 127.0.0.1 / app_lara10" and the page title is "Daftar Mahasiswa". Below the title, there is a navigation bar with links: Akademik, Home, Mahasiswa, Dosen, Prodi. The main content area has a heading "Daftar Mahasiswa Jurusan TI". Below the heading is a table with the following data:

No	Nim	Nama	Email	Alamat
1	2022090909	Linus B Torvalds	linus@linux.org	Jl. Sudirman no.10 Padang

At the bottom of the page, there is a dark footer bar with the text "Copyright © 2023 PNP".

15. Menampilkan Data dengan method find()

Method find() bisa dipakai untuk mencari data Model berdasarkan kolom id. Hasil dari method ini langsung berbentuk object, yang sama seperti method first(). Berikut contoh penggunaannya:

```
146
147 public function find()
148 {
149     $mahasiswa=Mahasiswa::find(2);
150     return view('akademik.mahasiswa',['mahasiswa'=>$mahasiswa]);
151 }
152
```

No	Nim	Nama	Email	Alamat
2	20210298	Steve job	steve@apple.com	Jl. sutomo no.11 Solok

Copyright © 2023 PNP

16. Menampilkan Data dengan Method Latest()

Method latest() berguna untuk mengambil collection yang sudah di urutkan berdasarkan tanggal pembuatan secara menurun, data paling akhir yang diinput akan ada di urutan pertama. Berikut contoh penggunaannya:

```
152
153 public function latest()
154 {
155     $mahasiswa=Mahasiswa::latest()->get();
156     return view('akademik.mahasiswa',['mahasiswa'=>$mahasiswa]);
157 }
158
```

No	Nim	Nama	Email	Alamat
5	202007890	M. Yazid	yazid@gmail.com	Padang
6	202007891	M. Rasyid	rasyid@gmail.com	Padang
2	20210298	Steve job	steve@apple.com	Jl. sutomo no.11 Solok
1	2022090909	Linus B Torvalds	linus@linux.org	Jl. Sudirman no.10 Padang

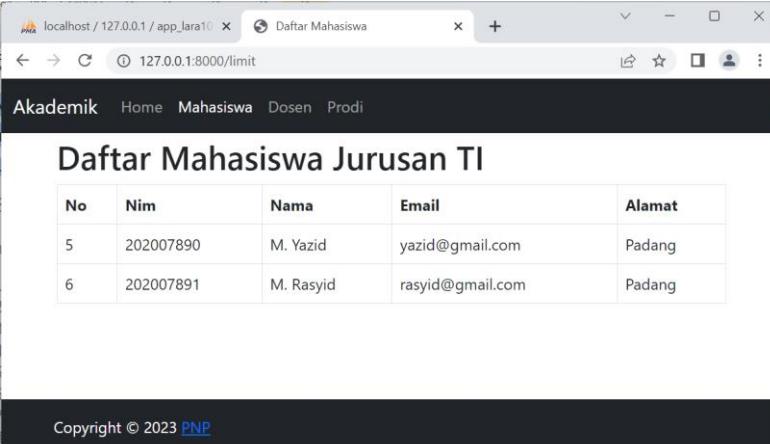
Copyright © 2023 PNP

17. Menampilkan Data dengan Method Limit()

Method limit() bisa dipakai untuk membatasi hasil collection. Method ini butuh sebuah argument berupa angka. Sebagai contoh, limit(1) hanya akan mengambil 1 object, sedangkan limit(3) akan mengambil 3 object teratas dari collection.

```
158
159 public function limit()
160 {
161     $mahasiswa=Mahasiswa::latest()->limit(2)->get();
162     return view('akademik.mahasiswa',['mahasiswa'=>$mahasiswa]);
163 }
164
```

Perintah `Mahasiswa::latest()->limit(2)->get()` dipakai untuk menampilkan 2 nama mahasiswa yang terakhir diinput.



The screenshot shows a web browser window with the URL `localhost / 127.0.0.1 / app_lara10` and the path `127.0.0.1:8000/limit`. The page title is "Daftar Mahasiswa Jurusan TI". The header includes links for Akademik, Home, Mahasiswa, Dosen, and Prodi. The main content is a table with the following data:

No	Nim	Nama	Email	Alamat
5	202007890	M. Yazid	yazid@gmail.com	Padang
6	202007891	M. Rasyid	rasyid@gmail.com	Padang

At the bottom of the page, there is a footer bar with the text "Copyright © 2023 PNP".

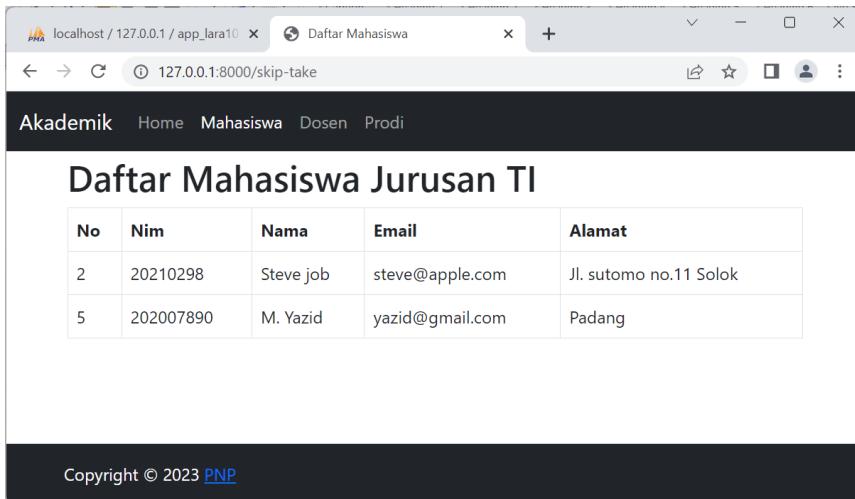
18. Menampilkan Data dengan Skip() dan Take()

Kedua method ini sudah pernah kita bahas pada materi menampilkan data menggunakan query builder. Method skip() berfungsi untuk melompati beberapa data, sedangkan method take() dipakai untuk mengambil sejumlah data tabel. Kedua method ini butuh argument berupa angka. Berikut contoh penggunaan dari `skip()` dan `take()`:

```
164
165 public function skipTake()
166 {
167     $mahasiswa=Mahasiswa::orderBy('id')->skip(1)->take(2)->get();
168     return view('akademik.mahasiswa',['mahasiswa'=>$mahasiswa]);
169 }
170
```

Perintah di baris 167 bisa dibaca: Urutkan tabel mahasiswa berdasarkan id (dari nilai terendah), lewatkan 1 data paling atas, lalu ambil 2 data setelahnya. Jika menggunakan

perintah SQL, ini sama dengan perintah `SELECT * FROM mahasiswa ORDER BY id LIMIT 1,2`



A screenshot of a web browser window titled "Daftar Mahasiswa Jurusan TI". The URL is "127.0.0.1:8000/skip-take". The page header includes "Akademik", "Home", "Mahasiswa", "Dosen", and "Prodi". Below the header is a table with the following data:

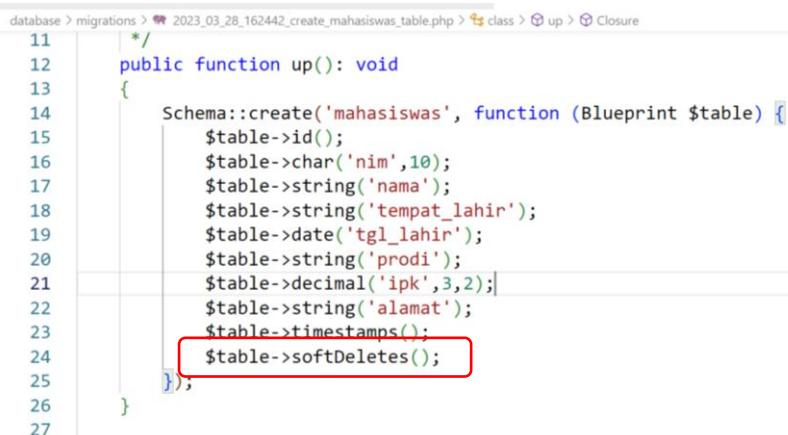
No	Nim	Nama	Email	Alamat
2	20210298	Steve job	steve@apple.com	Jl. sutomo no.11 Solok
5	202007890	M. Yazid	yazid@gmail.com	Padang

At the bottom of the page, there is a copyright notice: "Copyright © 2023 PNP".

19. Soft Delete

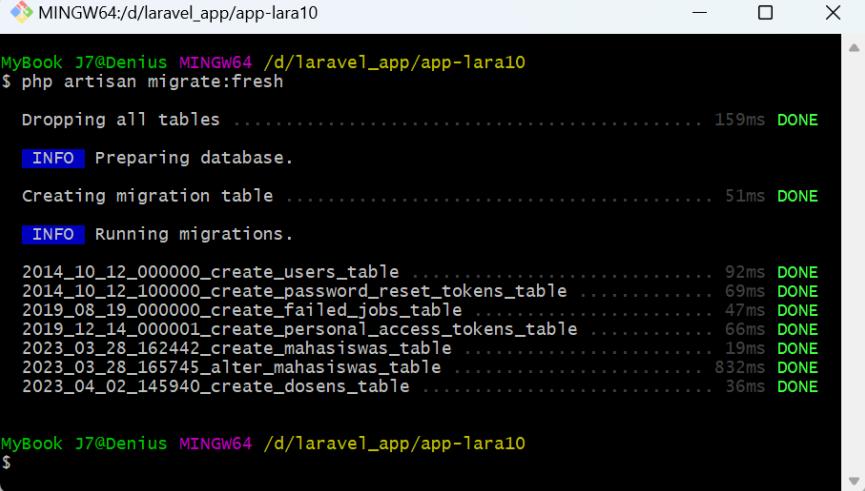
Bawaan dari MySQL, data yang dihapus dengan query `DELETE` tidak bisa dikembalikan lagi. Ini kadang jadi masalah jika user pengguna aplikasi kita tidak sengaja menghapus data penting. Eloquent memberikan solusi untuk hal ini, yang disebut sebagai **Soft Delete**. Teknik yang digunakan adalah, eloquent tidak benar-benar menghapus data dari dalam tabel, tapi menambah satu kolom khusus sebagai penanda data yang telah dihapus. Ketika ditampilkan, misalnya dengan method `all()`, data ini tidak akan terlihat. Jika di lain waktu kita berubah pikiran, data tersebut bisa dikembalikan ke kondisi semula, yakni dengan cara menghapus kolom penanda tadi.

Ada beberapa syarat agar bisa menggunakan **soft delete**. Pertama, sebuah tabel harus memiliki kolom `delete_at`. Untuk membuatnya silahkan buka kembali file migration dari tabel `mahasiswa` lalu tambah perintah `$table->softDeletes()` ke dalam struktur tabel:



```
database > migrations > 2023_03_28_162442_create_mahasiswa_table.php > class > up > Closure
11  /*
12  public function up(): void
13  {
14      Schema::create('mahasiswa', function (Blueprint $table) {
15          $table->id();
16          $table->char('nim', 10);
17          $table->string('nama');
18          $table->string('tempat_lahir');
19          $table->date('tgl_lahir');
20          $table->string('prodi');
21          $table->decimal('ipk', 3, 2);
22          $table->string('alamat');
23          $table->timestamps();
24          $table->softDeletes(); // This line is highlighted with a red box
25      });
26  }
27 }
```

Method `$table->softDeletes()` adalah method khusus yang disediakan Laravel untuk membuat kolom `delete_at`. Pada kode di atas, perintah ini ada di baris 24. Supaya struktur tabel mahasiswa berubah, lakukan migrasi dan rollback atau cukup dengan perintah `php artisan migrate:fresh`.

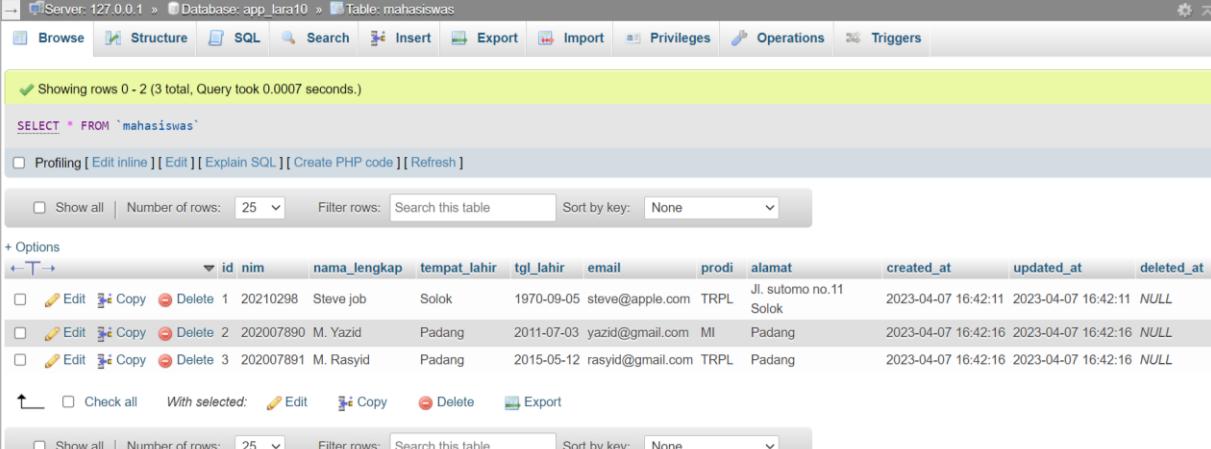


```
MINGW64:/d/laravel_app/app-lara10
$ php artisan migrate:fresh
Dropping all tables ..... 159ms DONE
INFO Preparing database.
Creating migration table ..... 51ms DONE
INFO Running migrations.

2014_10_12_000000_create_users_table ..... 92ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 69ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 47ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 66ms DONE
2023_03_28_162442_create_mahasiswa_table ..... 19ms DONE
2023_03_28_165745_alter_mahasiswa_table ..... 832ms DONE
2023_04_02_145940_create_dosens_table ..... 36ms DONE

MyBook J7@Denius MINGW64 /d/laravel_app/app-lara10
$
```

Selanjutnya tambahkan lagi data mahasiswa dengan menjalankan `localhost:8000/insert`, `localhost:8000/mass-assignment`. Setelah itu, cek isi tabel:



	Edit	Copy	Delete	1	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at	deleted_at
Edit	Copy	Delete	1	20210298	Steve job	Solok	1970-09-05	steve@apple.com	TRPL	Jl. sutomo no.11 Solok			2023-04-07 16:42:11	2023-04-07 16:42:11	NULL
Edit	Copy	Delete	2	202007890	M. Yazid	Padang	2011-07-03	yazid@gmail.com	MI	Padang			2023-04-07 16:42:16	2023-04-07 16:42:16	NULL
Edit	Copy	Delete	3	202007891	M. Rasyid	Padang	2015-05-12	rasyid@gmail.com	TRPL	Padang			2023-04-07 16:42:16	2023-04-07 16:42:16	NULL

Di sisi paling kanan, terdapat kolom `delete_at`. Inilah kolom yang di-generate oleh method `$table->softDeletes()` dari file migration. Saat ini nilainya NULL untuk semua baris tabel.

Syarat kedua agar bisa menggunakan **soft delete** adalah, import class `Illuminate\Database\Eloquent\SoftDeletes` ke dalam Model.

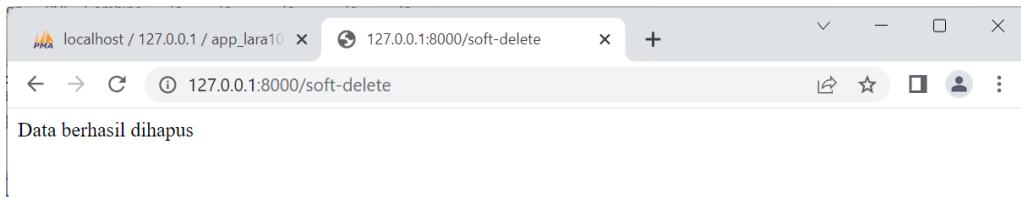
Berikut isi model Mahasiswa dengan penambahan perintah ini:

```
app > Models > Mahasiswa.php > Mahasiswa
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7 use Illuminate\Database\Eloquent\SoftDeletes;
8
9 class Mahasiswa extends Model
10 {
11     use HasFactory;
12     protected $guarded = [];
13     use SoftDeletes;
14 }
```

Terdapat 2 kode tambahan, yakni di baris 7 untuk import class SoftDeletes, dan di baris 13 untuk menggunakan class ini. Persiapan kita sudah selesai, sekarang kita coba menghapus salah satu data mahasiswa:

```
170 public function softDelete()
171 {
172     Mahasiswa::where('id','3')->delete();
173     return ('Data berhasil dihapus');
174 }
175
```

Data yang akan dihapus adalah mahasiswa dengan id = 3. Jalankan method ini dengan mengakses route `localhost:8000/soft-delete`.



Proses penghapusan berhasil. Mari kita cek ke database:

Server: 127.0.0.1 » Database: app_lara10 » Table: mahasiswa

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 2 (3 total, Query took 0.00007 seconds.)

SELECT * FROM `mahasiswa`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	id	nim	nama_lengkap	tempat_lahir	tgl_lahir	email	prodi	alamat	created_at	updated_at	deleted_at
<input type="checkbox"/>	1	20210298	Steve job	Solok	1970-09-05	steve@apple.com	TRPL	Jl. sutomo no.11 Solok	2023-04-07 16:42:11	2023-04-07 16:42:11	NULL
<input type="checkbox"/>	2	202007890	M. Yazid	Padang	2011-07-03	yazid@gmail.com	MI	Padang	2023-04-07 16:42:16	2023-04-07 16:42:16	NULL
<input type="checkbox"/>	3	202007891	M. Rasyid	Padang	2015-05-12	rasyid@gmail.com	TRPL	Padang	2023-04-07 16:42:16	2023-04-07 16:47:27	2023-04-07 16:47:27

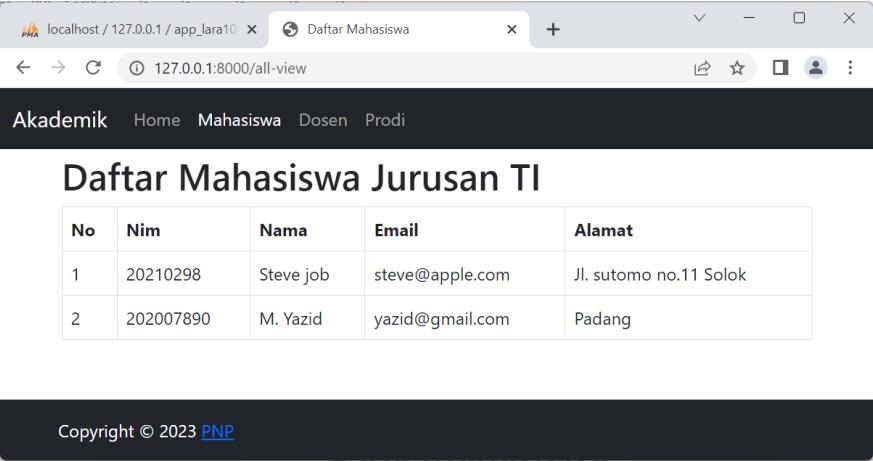
Check all With selected: Edit Copy Delete Export

Show all Number of rows: 25 Filter rows: Search this table Sort by key: None

Mahasiswa yang memiliki id=3 adalah M. Rasyid, setelah dilakukan proses delete menggunakan Eloquent, data ini tetap masih ada di database, namun kolom delete_at berisi nilai *timestamp* yang berisi tanggal dan waktu saat proses delete di lakukan.

Sekarang kita akan coba akses isi tabel mahasiswas menggunakan method all(). Kode program untuk proses ini sudah kita buat sebelumnya dan bisa diakses dari halaman

localhost:8000/all-view:



No	Nim	Nama	Email	Alamat
1	20210298	Steve job	steve@apple.com	Jl. sutomo no.11 Solok
2	202007890	M. Yazid	yazid@gmail.com	Padang

Copyright © 2023 PNP

Terlihat data mahasiswa M. RAsyid tidak tampil, padahal method all() dipakai untuk menampilkan seluruh isi tabel. Inilah hasil dari proses **soft delete**, dimana data seolah-olah telah terhapus dan tidak bisa diakses lagi, termasuk dengan berbagai method eloquent yang sudah kita bahas.

Syaratnya tentu saja data tabel harus diakses menggunakan Eloquent. Jika isi tabel diakses menggunakan *raw query* maupun *query builder*, data yang sudah di soft-delete tetap akan tampil. Namun bagaimana jika kita tetap ingin menampilkan data yang sudah di soft delete dari Eloquent? Caranya, gunakan method `Model::withTrashed()`, seperti contoh berikut:

```
172 |     $mahasiswa = Mahasiswa::withTrashed()->get();
173 |     return ('Data berhasil dihapus');
174 }
175
176 public function withTrashed()
177 {
178     $mahasiswa = Mahasiswa::withTrashed()->get();
179     return view('akademik.mahasiswa', ['mahasiswa' => $mahasiswa]);
180 }
```

Daftar Mahasiswa Jurusan TI

No	Nim	Nama	Email	Alamat
1	20210298	Steve job	steve@apple.com	Jl. sutomo no.11 Solok
2	202007890	M. Yazid	yazid@gmail.com	Padang
3	202007891	M. Rasyid	rasyid@gmail.com	Padang

Copyright © 2023 PNP

Dengan perintah `Mahasiswa::withTrashed()->get()`, kita bisa mengakses semua data termasuk yang sudah di soft delete.

20. Restore Soft Delete

Tujuan dari soft delete adalah agar data yang sudah terhapus bisa diakses kembali. Untuk keperluan ini, Eloquent menyediakan method `restore()`. Berikut contoh penggunaannya:

```
178     $mahasiswas=Mahasiswa::withTrashed()->get();
179     return view('akademik.mahasiswa',[ 'mahasiswas'=>$mahasiswas]);
180 }
181
182 public function restore()
183 {
184     Mahasiswa::withTrashed()->where('id','3')->restore();
185     return 'Berhasil di restore';
186 }
187
```

Jalankan `localhost:8000/restore`

127.0.0.1:8000/restore

Berhasil di restore

	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	1	20210298	Steve job	Solok	1970-09-05	steve@apple.com	TRPL	Jl. sutomo no.11 Solok	2023-04-07 16:42:11	2023-04-07 16:42:11	NULL
	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	2	202007890	M. Yazid	Padang	2011-07-03	yazid@gmail.com	MI	Padang	2023-04-07 16:42:16	2023-04-07 16:42:16	NULL
	<input type="checkbox"/> Edit	<input type="checkbox"/> Copy	<input type="checkbox"/> Delete	3	202007891	M. Rasyid	Padang	2015-05-12	rasyid@gmail.com	TRPL	Padang	2023-04-07 16:42:16	2023-04-07 16:54:16	NULL

Hasilnya, kolom `delete_at` untuk M. Rasyid kembali menjadi NULL, yang berarti sudah bisa diakses kembali seperti biasa.

21. Menghapus Data secara Permanen

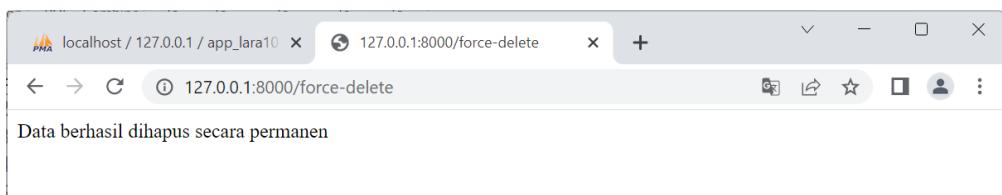
Dalam beberapa situasi, kita ingin menghapus data secara permanen dari database, bukan lagi sekedar soft delete. Namun method `delete()` sudah tidak bisa dipakai karena itu hanya menghapus data sebagai soft delete. Untuk keperluan ini, Eloquent menyediakan method `forceDelete()`. Berikut cara penggunaan dari method `forceDelete()`:

```

186 }
187
188 public function forceDelete()
189 {
190     Mahasiswa::where('id','3')->forceDelete();
191     return ('Data berhasil dihapus secara permanen');
192 }
193

```

Dengan perintah `Mahasiswa::where('id','3')->forceDelete()`, maka data mahasiswa dengan `id=3` akan dihapus secara permanen, yakni sama seperti method `delete()` jika tanpa menggunakan soft delete. Jalankan kode program di atas dengan mengakses halaman `localhost:8000/force-delete`:



Data yang sudah dihapus secara permanen tidak bisa dikembalikan lagi, karena memang sudah tidak ada di database.