KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
http://www.polinema.ac.id

Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 1 (satu)

Nama : Dea Putri Nastiti
NIM : 2241720117
Kelas : TI 2H

JOBSHEET 04

MODEL dan ELOQUENT ORM

Sebelumnya kita sudah membahas mengenai *Migration, Seeder*, *DB Façade, Query Builder,* dan sedikit tentang *Eloquent ORM* yang ada di Laravel. Sebelum kita masuk pada pembuatan aplikasi berbasis website, alangkah baiknya kita perlu menyiapkan Basis data sebagai tempat menyimpan data-data pada aplikasi kita nanti. Selain itu, umumnya kita perlu menyiapkan juga data awal yang kita gunakan sebelum membuat aplikasi, seperti data user administrator, data pengaturan sistem, dll.

Dalam pertemuang kali ini kita akan memahami tentang bagainama cara menampilkan data, mengubah data, dan menghapus data menggunakan teknik Eloquent.

Sesuai dengan `Studi Kasus PWL.pdf`.
Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL_POS.**

Project **PWL_POS** akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajarai

Jabarkan apa itu Eloquent ORM dan Hubungannya dengan file Model di laravel

Jawab: ORM (Object Relational Mapping) adalah fitur laravel yang memungkinkan suatu tabel dalam database menjadi objek. Dengan Eloquent kita bisa melakukan operasi database (CRUD) secara langsung tanpa menulis query SQL. Eloquent akan mengubah operasi yang dilakukan di php menjadi query SQL.

Eloquent ORM memerlukan Model untuk proses konversi data pada tabel menjadi object (object akan di akses dari dalam controller). Sehingga membuat Model pada Laravel berarti menggunakan Eloquent ORM.

A. PROPERTI `$fillable` DAN `$guarded`

1. `$fillable`

Variable `$fillable` berguna untuk mendaftarkan atribut (nama kolom) yang bisa kita isi ketika melakukan insert atau update ke database. Sebelumnya kita sudah memahami menambahkan record baru ke database. Untuk langkah menambahkan Variable `$fillable` bisa dengan menambahkan *script* seperti di bawah ini pada file model

```
protected $fillable = ['level_id', 'username'];
```

Praktikum 1 - $fillable:

1. Buka file model dengan nama `UserModel.php` dan tambahkan `$fillable` seperti gambar di bawah ini

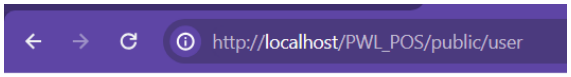
app > Models > UserModel.php > UserModel

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user'; //Mendefinisikan nama tabel yang digunakan oleh model ini
    protected $primarykey = 'user_id'; //mendefinisikan primary key dari tabel yang dignakan
    protected $fillable = ['level_id', 'username', 'nama', 'password'];
}
```

2. Buka file controller dengan nama `UserController.php` dan ubah *script* untuk menambahkan data baru seperti gambar di bawah ini

```php
class UserController extends Controller
    public function index(){

        $data = [
            'level_id' => 2,
            'username' => 'manager_dua',
            'nama' => 'Manager 2',
            'password' => Hash::make('12345')
        ];
        UserModel::create($data);
        //coba akses model UserModel
        $user = userModel::all(); //ambil semua
        return view('user', ['data'=>$user]);
    }

}
```

3. Simpan kode program Langkah 1 dan 2, dan jalankan perintah web server. Kemudian jalankan link `localhost/PWL_POS/public/user` pada *browser* dan amati apa yang terjadi



Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
6	manager_tiga	Manager ke-3	2
7	manager_dua	Manager 2	2

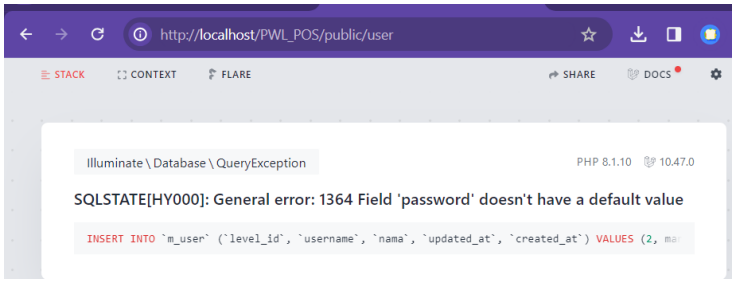
4. Ubah file model `UserModel.php` seperti pada gambar di bawah ini pada bagian `$fillable`

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user'; //Mendefinisikan nama tabel
    protected $primarykey = 'user_id'; //mendefinisikan prim
    //protected $fillable = ['level_id', 'username', 'nama',
    protected $fillable = ['level_id', 'username', 'nama'];
}
```

5. Ubah kembali file controller `UserController.php` seperti pada gambar di bawah hanya bagian array pada *$data*

```
app > Http > Controllers > UserController.php > UserController > index
 9   class UserController extends Controller
12       public function index(){
26           $data = [
27               'level_id' => 2,
28               //'username' => 'manager_dua',
29               //'nama' => 'Manager 2',
30               'username' => 'manager_empat',
31               'nama' => 'Manager 3    ',
32               'password' => Hash::make('12345')
33           ];
34           UserModel::create($data);
35           //coba akses model UserModel
36           $user = userModel::all(); //ambil semua data
37           return view('user', ['data'=>$user]);
38       }
```

6. Simpan kode program Langkah 4 dan 5. Kemudian jalankan pada *browser* dan amati apa yang terjadi

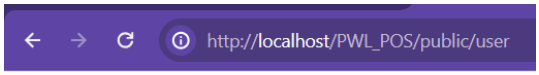


Akan terjadi error karena pada UserModel kolom password tidak dituliskan dan kolom password tidak memiliki default value. Sehingga saat di UserController menginputkan kolom username, kolom username tidak dikenali.

Perbaikan

```
protected $fillable = ['level_id', 'username', 'nama', 'password'];
```

Hasil



Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
6	manager_tiga	Manager ke-3	2
7	manager_dua	Manager 2	2
8	manager_empat	Manager 3	2

7. Laporkan hasil Praktikum-1 ini dan *commit* perubahan pada *git*.

2. `$guarded`

Kebalikan dari `$fillable` adalah `$guarded`. Semua kolom yang kita tambahkan ke `$guarded` akan diabaikan oleh Eloquent ketika kita melakukan insert/update. Secara default `$guarded` isinya `array("*")`, yang berarti semua atribut tidak bisa diset melalui ***mass assignment*** (jabarkan istilah ini).

Jawab: mass assigment memudahkan kita untuk menetapkan nilai ke beberapa atribut model sekaligus dari array datat input. Penggunaan mass assigment dapat menghemat waktu dan tenaga. $guarded digunakan untuk mendefinisikan atribut mana saja yang tidak diizinkan untuk diisi menggunakan mass assigment.

B. RETRIEVING SINGLE MODELS

Selain mengambil semua rekaman yang cocok dengan kueri tertentu, Anda juga dapat mengambil rekaman tunggal menggunakan metode `find`, `first`, atau `firstWhere`. Daripada mengembalikan kumpulan model, metode ini mengembalikan satu contoh model dan dilakukan pada controller:

```php
// Ambil model dengan kunci utamanya...
$user = UserModel::find(1);

// Ambil model pertama yang cocok dengan batasan kueri...
$user = UserModel::where('active', 1)->first();

// Alternatif untuk mengambil model pertama yang cocok dengan batasan kueri...
$user = UserModel::firstWhere('active', 1);
```

Praktikum 2.1 – Retrieving Single Models

`UserController.php` *script* seperti gambar

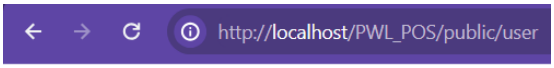
1. Buka file controller dengan nama dan ubah di bawah ini

```
app > Http > Controllers > UserController.php > UserController > index
 9  class UserController extends Controller
12      public function index(){
35
36          $user = UserModel::find(1);
37          //coba akses model UserModel
38          //$user = userModel::all(); //ambil semua da
39          return view('user', ['data'=>$user]);
40      }
```

2. Buka file *view* dengan nama `user.blade.php` dan ubah *script* seperti gambar di bawah ini

```
 8      <table border="1" cellpadding="2" cellspacing="0">
 9          <tr>
10              <th>ID</th>
11              <th>Username</th>
12              <th>Nama</th>
13              <th>ID Level Pengguna</th>
14          </tr>
15          {{-- @foreach ($data as $d)
16          <tr>
17              <td>{{ $d->user_id }}</td>
18              <td>{{ $d->username }}</td>
19              <td>{{ $d->nama }}</td>
20              <td>{{ $d->level_id }}</td>
21          </tr>
22          @endforeach --}}
23          <tr>
24              <td>{{ $data->user_id }}</td>
25              <td>{{ $data->username }}</td>
26              <td>{{ $data->nama }}</td>
27              <td>{{ $data->level_id }}</td>
28          </tr>
```

3. Simpan kode program Langkah 1 dan 2. Kemudian jalankan link http://localhost:8000/user pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



Data User

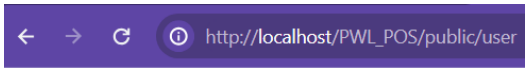
ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Find(1) digunakan untuk mencari dan mengambil record dari database berdasarkan primary key. Pada praktikum dicari primary key = 1, maka akan menampilkan record dengan id 1.

4. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
37      $user = UserModel::where('level_id', 1)->first();
```

5. Simpan kode program Langkah 4. Kemudian jalankan link http://localhost:8000/user pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



Data User

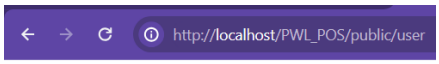
ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Where('level_id', 1)->first() → digunakan unutk mencari record pertama yang memiliki level_id = 1

6. Ubah file controller dengan nama dan ubah di bawah ini

```
38      $user = UserModel::firstWhere('level_id', 1);
```

7. Simpan kode program Langkah 6. Kemudian jalankan link http://localhost:8000/user pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

firstWhere('level_id', 1) → memiliki arti yang sama dengan Where('level_id', 1)->first() yaitu digunakan unutk mencari record pertama yang memiliki level_id = 1

Terkadang Anda mungkin ingin melakukan beberapa tindakan lain jika tidak ada hasil yang ditemukan. Metode `findOr` and `firstOr` akan mengembalikan satu contoh model atau, jika tidak ada hasil yang ditemukan, jalankan penutupan yang diberikan. Nilai yang dikembalikan oleh penutupan akan dianggap sebagai hasil dari metode ini:

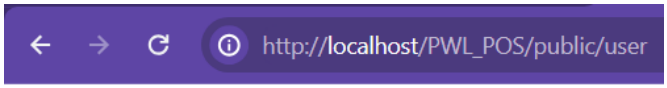
`UserController.php` *script* seperti gambar

```php
$user = UserModel::findOr(1, function () {
    // ...
});

$user = UserModel::where('level_id', '>', 3)->firstOr(function () {
    // ...
});
```

8. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```php
app > Http > Controllers > UserController.php > ...
 9   ss UserController extends Controller
12    public function index(){
41        $user = UserModel::findOr(1, ['username', 'nama'], function(){
42            abort(404);
43        });
44        return view('user', ['data'=>$user]);
45    }
```

9. Simpan kode program Langkah 8. Kemudian jalankan link http://localhost:8000/user pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



Data User

ID	Username	Nama	ID Level Pengguna
	admin	Administrator	

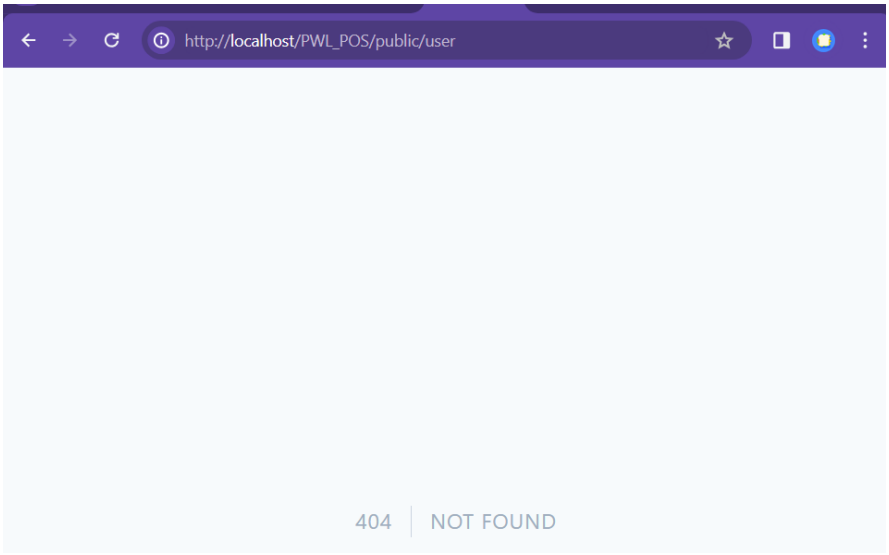
findOr digunakan untuk mencari record yang sesuai dengan parameter nya, yaitu primary key 1. Apabila record tidak ditemukan maka akan memunculkan abort(404).

10. Ubah file controller dengan nama dan ubah di bawah ini

```php
app > Http > Controllers > UserController.php > UserController > index
 9   s UserController extends Controller
12   ublic function index(){
41       $user = UserModel::findOr(20, ['username', 'nama'], function(){
42           abort(404);
43       });
44       return view('user', ['data'=>$user]);
45   }
```

`UserController.php` *script* seperti gambar

11. Simpan kode program Langkah 10. Kemudian jalankan link http://localhost:8000/user pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



findOr digunakan untuk mencari record yang sesuai dengan parameter nya, yaitu primary key 20. Karena record tidak ditemukan (tidak ada record dengan user_id = 10) maka memunculkan abort(404) NOT FOUND.

12. Laporkan hasil Praktikum-2.1 ini dan *commit* perubahan pada *git*.

Praktikum 2.2 – *Not Found Exceptions*

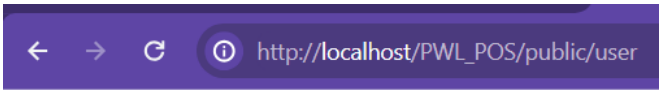
Terkadang Anda mungkin ingin memberikan pengecualian jika model tidak ditemukan. Hal ini sangat berguna dalam *rute* atau pengontrol. Metode `findOrFail` and `firstOrFail` akan mengambil hasil pertama dari kueri; namun, jika tidak ada hasil yang ditemukan, sebuah `Illuminate\Database\Eloquent\ModelNotFoundException` akan dilempar. Berikut ikuti langkah-langkah di bawah ini:

1. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

`UserController.php` *script* seperti gambar

```
app > Http > Controllers > UserController.php > UserController > index
 9   class UserController extends Controller
12       public function index(){
43           // });
44           $user = UserModel::findOrFail(1);
45           return view('user', ['data'=>$user]);
46       }
```

2. Simpan kode program Langkah 1. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



Data User

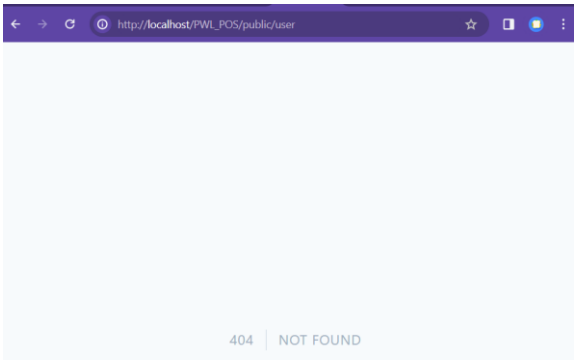
ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

findOrFile mengembalikan record sesuai parameter nya yaitu user_id = 1.

3. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
app > Http > Controllers > UserController.php > UserController > index
 9   ;s UserController extends Controller
12    public function index(){
42        //      abort(404);
43        // });
44        //$user = UserModel::findOrFail(1);
45        $user = UserModel::where('username', 'manager9')->firstOrFail();
46        return view('user', ['data'=>$user]);
```

4. Simpan kode program Langkah 3. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



`UserController.php` *script* seperti gambar

firstOrFail() digunakan untuk mencari record pertama yang cocok dengan parameter (dalam praktikum username = manager9), pabila tidak ada record yang ditemukan maka akan melempar exception 'ModelNotFoundException' (404 NOT FOUND)

5. Laporkan hasil Praktikum-2.[1] ini dan *commit* perubahan pada *git*.

Praktikum 2.[1] – *Retreiving Aggregrates*

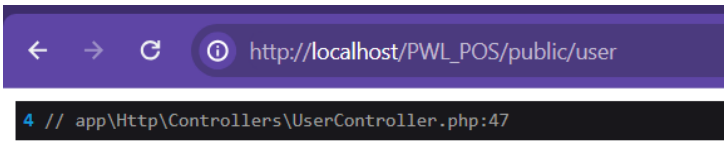
Saat berinteraksi dengan model Eloquent, Anda juga dapat menggunakan metode agregat `count`, `sum`, `max`, dan lainnya yang disediakan oleh pembuat kueri Laravel. Seperti yang Anda duga, metode ini mengembalikan nilai skalar dan contoh model Eloquent:

```
$count = UserModel::where('active', 1)->count();

$max = UserModel::where('active', 1)->max('price');
```

1. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

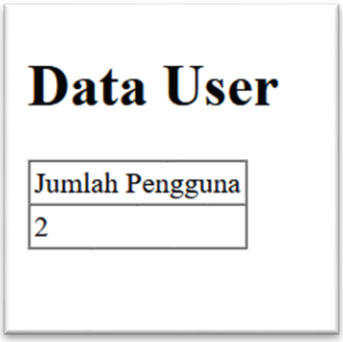
```
46          $user = UserModel::where('level_id', 2)->count();
47          dd($user);
48          return view('user', ['data'=>$user]);
49      }
```

2. Simpan kode program Langkah 1. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan



Count() digunakan untuk menghitung jumlah record di database yang sesuai dengan kondisi. Fungsi dd() → dump and die, akan menghentikan eksekusi script lebih lanjut. Oleh karena itu baris return view tidak akan dijalankan karena dihentikan dd()

3. Buat agar jumlah script pada langkah 1 bisa tampil pada halaman browser, sebagai contoh bisa lihat gambar di bawah ini dan ubah script pada file view supaya bisa muncul datanya



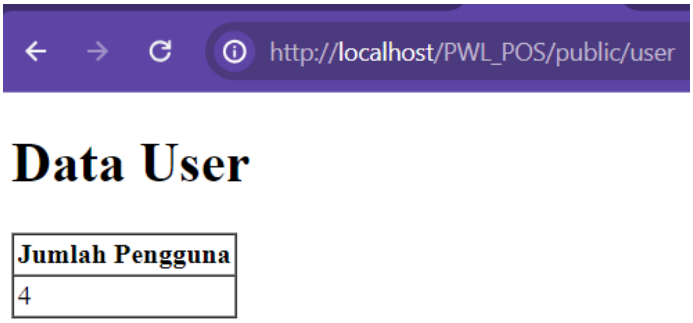
UserController

```
app > Http > Controllers > UserController.php > UserController > index
 9   class UserController extends Controller
12       public function index(){
46           $user = UserModel::where('level_id', 2)->count();
47           //dd($user);
48           return view('user', ['data'=>$user]);
49       }
50
51   }
```

user.blade

```
resources > views > user.blade.php > html > body > table > tr
 1  <!DOCTYPE html>
 2  <html>
 3      <head>
 4          <title>Data User</title>
 5      </head>
 6      <body>
 7          <h1>Data User</h1>
 8          <table border="1" cellpadding="2" cellspacing="0">
 9              <tr>
10                  <th>Jumlah Pengguna</th>
11              </tr>
12              <tr>
13                  <td>{{ $data }}
14              </tr>
15          </table>
16      </body>
17  </html>
```

Output



4. Laporkan hasil Praktikum-2.3 ini dan *commit* perubahan pada *git*.

Praktikum 2.4 – *Retreiving or Creating Models*

Metode `firstOrCreate` merupakan metode untuk melakukan *retrieving data* (mengambil data) berdasarkan nilai yang ingin dicari, jika data tidak ditemukan maka method ini akan melakukan insert ke table datadase tersebut sesuai dengan nilai yang dimasukkan.

Metode `firstOrNew`, seperti `firstOrCreate`, akan mencoba menemukan/mengambil *record/data* dalam database yang cocok dengan atribut yang diberikan. Namun, jika data tidak ditemukan, data akan disiapkan untuk di-*insert*-kan ke database dan model baru akan dikembalikan. Perhatikan bahwa model yang dikembalikan `firstOrNew` belum disimpan ke database. Anda perlu memanggil metode `save()` secara manual untuk menyimpannya:

```php
$user = UserModel::firstOrCreate(
    [
        'username' => 'manager',
        'nama' => 'Manager',
    ],
);

$user = UserModel::firstOrNew(
    [
        'username' => 'manager',
        'nama' => 'Manager',
    ],
);
```

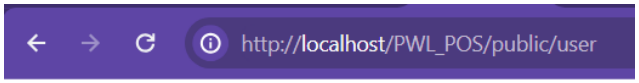
1. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
app > Http > Controllers > UserController.php > UserController > index
 9   class UserController extends Controller
12       public function index(){
48           $user = UserModel::firstOrCreate(
49               [
50                   'username' => 'manager',
51                   'nama' => 'Manager'
52               ],
53           );
54           return view('user', ['data'=>$user]);
55       }
```

2. Ubah kembali file *view* dengan nama `user.blade.php` dan ubah *script* seperti gambar di bawah ini

```
resources > views > user.blade.php > html > body > table
 1  <!DOCTYPE html>
 2  <html>
 3      <head>
 4          <title>Data User</title>
 5      </head>
 6      <body>
 7          <h1>Data User</h1>
 8          <table border="1" cellpadding="2" cellspacing="0">
 9              <tr>
10                  <th>ID</th>
11                  <th>Username</th>
12                  <th>Nama</th>
13                  <th>ID Level Pengguna</th>
14                  {{-- <th>Jumlah Pengguna</th> --}}
15              </tr>
16              <tr>
17                  <td>{{ $data->user_id }}</td>
18                  <td>{{ $data->username }}</td>
19                  <td>{{ $data->nama }}</td>
20                  <td>{{ $data->level_id }}</td>
21                  {{-- <td>{{ $data }} --}}
22              </tr>
```

3. Simpan kode program Langkah 1 dan 2. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



Data User

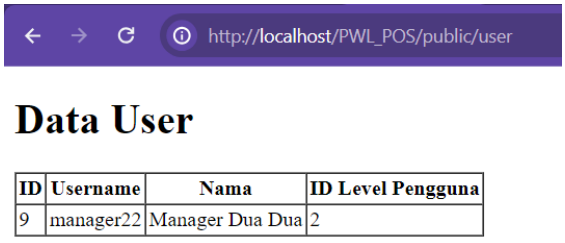
ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

firstOrCreate digunakan untuk meringkas jumlah kode ketika ingin mencari sebuah record dalam database berdasarkan kriteria. Jika tidak ditemukan maka akan membuat record baru.

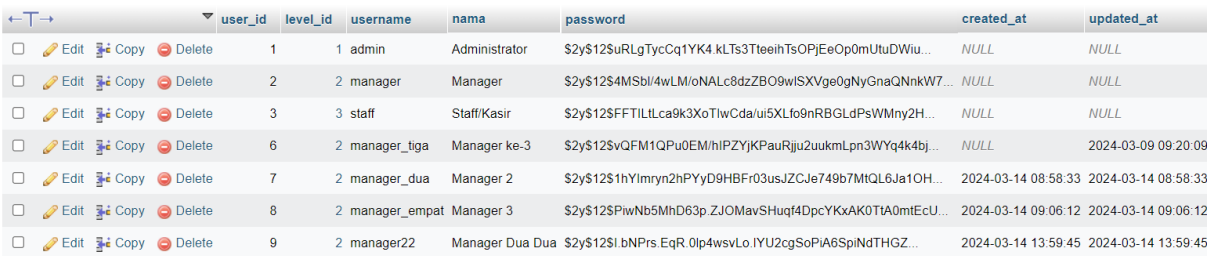
4. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
app > Http > Controllers > UserController.php > UserController > index
 9  class UserController extends Controller
12      public function index(){
48          $user = UserModel::firstOrCreate(
49              [
50                  'username' => 'manager22',
51                  'nama' => 'Manager Dua Dua',
52                  'password' => Hash::make('12345'),
53                  'level_id' =>2
54              ],
55          );
56          return view('user', ['data'=>$user]);
57      }
```

5. Simpan kode program Langkah 4. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan cek juga pada *phpMyAdmin* pada tabel `m_user` serta beri penjelasan dalam laporan



Karena record yang ingin dicari tidak ditemukan, maka fungsi firstOrCreate akan membuat record baru sesuai dengan kriteria yang ditulis. Record akan langsung ter input ke database.

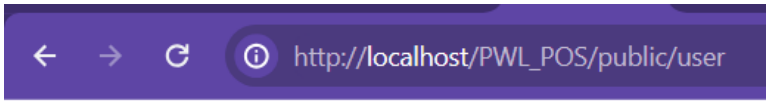


	user_id	level_id	username	nama	password	created_at	updated_at
Edit Copy Delete	1	1	admin	Administrator	$2y$12$uRLgTycCq1YK4.kLTs3TteeihTsOPjEeOp0mUtuDWiu...	NULL	NULL
Edit Copy Delete	2	2	manager	Manager	$2y$12$4MSbl/4wLM/oNALc8dzZBO9wlSXVge0gNyGnaQNnkW7...	NULL	NULL
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$FFTILtLca9k3XoTlwCda/ui5XLfo9nRBGLdPsWMny2H...	NULL	NULL
Edit Copy Delete	6	2	manager_tiga	Manager ke-3	$2y$12$vQFM1QPu0EM/hIPZYjKPauRjju2uukmLpn3WYq4k4bj...	NULL	2024-03-09 09:20:09
Edit Copy Delete	7	2	manager_dua	Manager 2	$2y$12$1hYImryn2hPYyD9HBFr03usJZCJe749b7MtQL6Ja1OH...	2024-03-14 08:58:33	2024-03-14 08:58:33
Edit Copy Delete	8	2	manager_empat	Manager 3	$2y$12$PiwNb5MhD63p.ZJOMavSHuqf4DpcYKxAK0TtA0mtEcU...	2024-03-14 09:06:12	2024-03-14 09:06:12
Edit Copy Delete	9	2	manager22	Manager Dua Dua	$2y$12$I.bNPrs.EqR.0lp4wsvLo.IYU2cgSoPiA6SpiNdTHGZ...	2024-03-14 13:59:45	2024-03-14 13:59:45

6. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
app > Http > Controllers > UserController.php > UserController > index
 9   class UserController extends Controller
12       public function index(){
56           $user = UserModel::firstOrNew(
57               [
58                   'username' => 'manager',
59                   'nama' => 'Manager'
60               ],
61           );
62           return view('user', ['data'=>$user]);
63       }
```

7. Simpan kode program Langkah 6. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



Data User

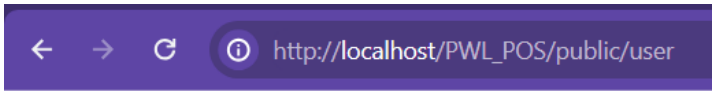
ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

firstOrNew akan mencari record yang sama di database, jika ditemukan maka akan me return sebuah instance model yang mewakili rcor. Apabila tidak ditemukan maka akan dibuat instance model baru dengan atribut yang ditentukan, namun tidak langsung tersimpan di database. Harus disimpan secara manual.

8. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
app > Http > Controllers > UserController.php > UserController > index
 9   class UserController extends Controller
12       public function index(){
56           $user = UserModel::firstOrNew(
57               [
58                   'username' => 'manager33',
59                   'nama' => 'Manager Tiga Tiga',
60                   'password' => Hash::make('12345'),
61                   'level_id' => 2
62               ],
63           );
64           return view('user', ['data'=>$user]);
65       }
```

9. Simpan kode program Langkah 8. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan cek juga pada *phpMyAdmin* pada tabel *m_user* serta beri penjelasan dalam laporan



Data User

ID	Username	Nama	ID Level Pengguna
	manager33	Manager Tiga Tiga	2

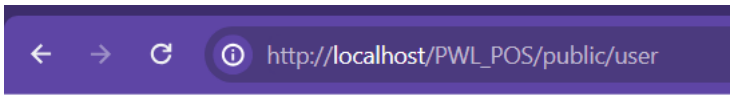
Menampilkan record yang baru di create tapi tidak langsung tersimpan di database

	user_id	level_id	username	nama	password	created_at	updated_at
Edit Copy Delete	1	1	admin	Administrator	$2y$12$uRLgTycCq1YK4.kLTs3TteeihTsOPjEeOp0mUtuDWiu...	NULL	NULL
Edit Copy Delete	2	2	manager	Manager	$2y$12$4MSbl/4wLM/oNALc8dzZBO9wlSXVge0gNyGnaQNnkW7...	NULL	NULL
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$FFTILtLca9k3XoTlwCda/ui5XLfo9nRBGLdPsWMny2H...	NULL	NULL
Edit Copy Delete	6	2	manager_tiga	Manager ke-3	$2y$12$vQFM1QPu0EM/hIPZYjKPauRjju2uukmLpn3WYq4k4bj...	NULL	2024-03-09 09:20:09
Edit Copy Delete	7	2	manager_dua	Manager 2	$2y$12$1hYImryn2hPYyD9HBFr03usJZCJe749b7MtQL6Ja1OH...	2024-03-14 08:58:33	2024-03-14 08:58:33
Edit Copy Delete	8	2	manager_empat	Manager 3	$2y$12$PiwNb5MhD63p.ZJOMavSHuqf4DpcYKxAK0TtA0mtEcU...	2024-03-14 09:06:12	2024-03-14 09:06:12
Edit Copy Delete	9	2	manager22	Manager Dua Dua	$2y$12$l.bNPrs.EqR.0lp4wsvLo.IYU2cgSoPiA6SpiNdTHGZ...	2024-03-14 13:59:45	2024-03-14 13:59:45

10. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
app > Http > Controllers > UserController.php > UserController > index
 9   class UserController extends Controller
12       public function index(){
56           $user = UserModel::firstOrNew(
57               [
58                   'username' => 'manager33',
59                   'nama' => 'Manager Tiga Tiga',
60                   'password' => Hash::make('12345'),
61                   'level_id' => 2
62               ],
63           );
64           $user->save();
65           return view('user', ['data'=>$user]);
66       }
```

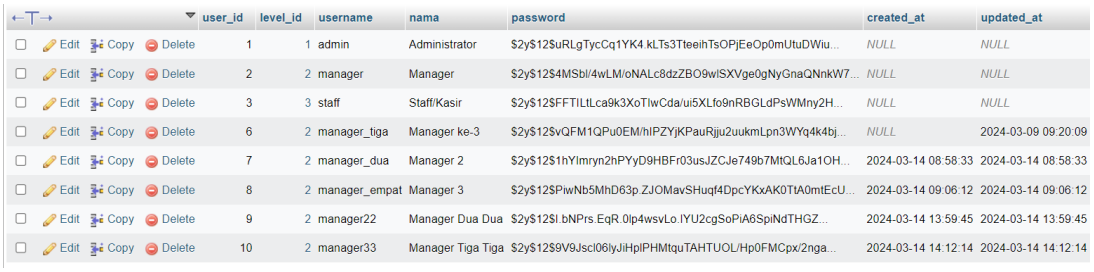
11. Simpan kode program Langkah 9. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan cek juga pada *phpMyAdmin* pada tabel `m_user` serta beri penjelasan dalam laporan



Data User

ID	Username	Nama	ID Level Pengguna
10	manager33	Manager Tiga Tiga	2

Record baru ditampilkan dan disimpan dalam database karena sudah disimpan secara manual menggunakan save()



	user_id	level_id	username	nama	password	created_at	updated_at
Edit Copy Delete	1	1	admin	Administrator	$2y$12$uRLgTycCq1YK4.kLTs3TteeihTsOPjEeOp0mUtuDWiu...	NULL	NULL
Edit Copy Delete	2	2	manager	Manager	$2y$12$4MSbl/4wLM/oNALc8dzZBO9wlSXVge0gNyGnaQNnkW7...	NULL	NULL
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$FFTILtLca9k3XoTlwCda/ui5XLfo9nRBGLdPsWMny2H...	NULL	NULL
Edit Copy Delete	6	2	manager_tiga	Manager ke-3	$2y$12$vQFM1QPu0EM/hlPZYjKPauRjju2uukmLpn3WYq4k4bj...	NULL	2024-03-09 09:20:09
Edit Copy Delete	7	2	manager_dua	Manager 2	$2y$12$1hYImryn2hPYyD9HBFr03usJZCJe749b7MtQL6Ja1OH...	2024-03-14 08:58:33	2024-03-14 08:58:33
Edit Copy Delete	8	2	manager_empat	Manager 3	$2y$12$PiwNb5MhD63p.ZJOMavSHuqf4DpcYKxAK0TtA0mtEcU...	2024-03-14 09:06:12	2024-03-14 09:06:12
Edit Copy Delete	9	2	manager22	Manager Dua Dua	$2y$12$l.bNPrs.EqR.0lp4wsvLo.IYU2cgSoPiA6SpiNdTHGZ...	2024-03-14 13:59:45	2024-03-14 13:59:45
Edit Copy Delete	10	2	manager33	Manager Tiga Tiga	$2y$12$9V9Jscl06lyJiHplPHMtquTAHTUOL/Hp0FMCpx/2nga...	2024-03-14 14:12:14	2024-03-14 14:12:14

12. Laporkan hasil Praktikum-2.4 ini dan *commit* perubahan pada *git*.

Praktikum 2.5 – *Attribute Changes*

Eloquent menyediakan metode `isDirty`, `isClean`, dan `wasChanged` untuk memeriksa keadaan internal model Anda dan menentukan bagaimana atributnya berubah sejak model pertama kali diambil.

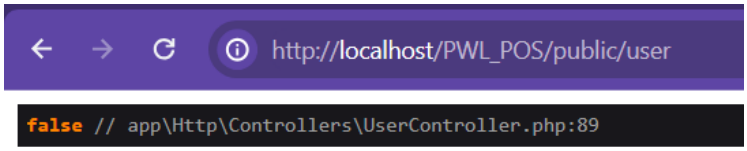
Metode `isDirty` menentukan apakah ada atribut model yang telah diubah sejak model diambil. Anda dapat meneruskan nama atribut tertentu atau serangkaian atribut ke metode `isDirty` untuk menentukan apakah ada atribut yang "kotor". Metode ini `isClean` akan menentukan apakah suatu atribut tetap tidak berubah sejak model diambil. Metode ini juga menerima argumen atribut opsional:

```php
$user = UserModel::create([
    'username' => 'manager44',
    'nama' => 'Manager44',
    'password' => Hash::make('12345'),
    'level_id' => 2,
]);

$user->username = 'manager45';

$user->isDirty(); // true
$user->isDirty('username'); // true
$user->isDirty('nama'); // false
$user->isDirty(['nama', 'username']); // true

$user->isClean(); // false
$user->isClean('username'); // false
$user->isClean('nama'); // true
$user->isClean(['nama', 'username']); // false

$user->save();

$user->isDirty(); // false
$user->isClean(); // true
```

1. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
app > Http > Controllers > UserController.php > UserController > index
 9   class UserController extends Controller
12       public function index(){
66           $user = UserModel::create([
67               'username' => 'manager55',
68               'nama' => 'Manager Lima Lima',
69               'password' => Hash::make('12345'),
70               'level_id' => 2,
71           ]);
72
73           $user->username = 'manager56';
74
75           $user->isDirty();//true
76           $user->isDirty('username');//true
77           $user->isDirty('nama');//false
78           $user->isDirty(['nama', 'username']);//true
79
80           $user->isClean();//false
81           $user->isClean('username');//false
82           $user->isClean('nama');//true
83           $user->isClean(['nama', 'username']);//false
84
85           $user->save();
86
87           $user->isDirty();//false
88           $user->isClean();//true
89           dd($user->isDirty());
90       }
91   }
```

2. Simpan kode program Langkah 1. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



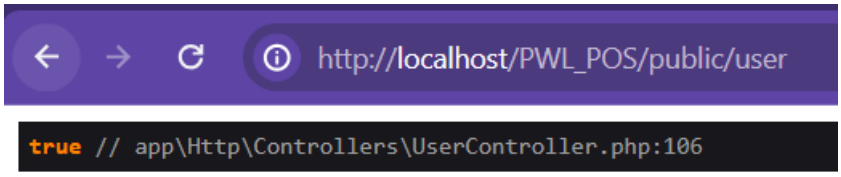
Metode ini `wasChanged` menentukan apakah ada atribut yang diubah saat model terakhir disimpan dalam siklus permintaan saat ini. Jika diperlukan, Anda dapat memberikan nama atribut untuk melihat apakah atribut tertentu telah diubah:

```php
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::create([
            'username' => 'manager11',
            'nama' => 'Manager11',
            'password' => Hash::make('12345'),
            'level_id' => 2,
        ]);

        $user->username = 'manager12';

        $user->save();

        $user->wasChanged(); // true
        $user->wasChanged('username'); // true
        $user->wasChanged(['username', 'level_id']); // true
        $user->wasChanged('nama'); // false
        $user->wasChanged(['nama', 'username']); // true
    }
}
```

3. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```php
app > Http > Controllers > UserController.php > UserController > index
 9  class UserController extends Controller
12      public function index(){
91          $user = UserModel::create([
92              'username' => 'manager11',
93              'nama' => 'Manager11',
94              'password' => Hash::make('12345'),
95              'level_id' => 2,
96          ]);
97
98          $user->username = 'manager12';
99
100         $user->save();
101
102         $user->wasChanged();//true
103         $user->wasChanged('username');//true
104         $user->wasChanged(['username', 'level_id']);//true
105         $user->wasChanged('nama');//false
106         dd($user->wasChanged(['nama', 'username']));//true
```

4. Simpan kode program Langkah 3. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



5. Laporkan hasil Praktikum-2.5 ini dan *commit* perubahan pada *git*.

Praktikum 2.6 – *Create, Read, Update, Delete (CRUD)*

Seperti yang telah kita ketahui, CRUD merupakan singkatan dari *Create, Read, Update* dan *Delete*. CRUD merupakan istilah untuk proses pengolahan data pada database, seperti input data ke database, menampilkan data dari database, mengedit data pada database dan menghapus data dari database. Ikuti langkah-langkah di bawah ini untuk melakukan CRUD dengan Eloquent

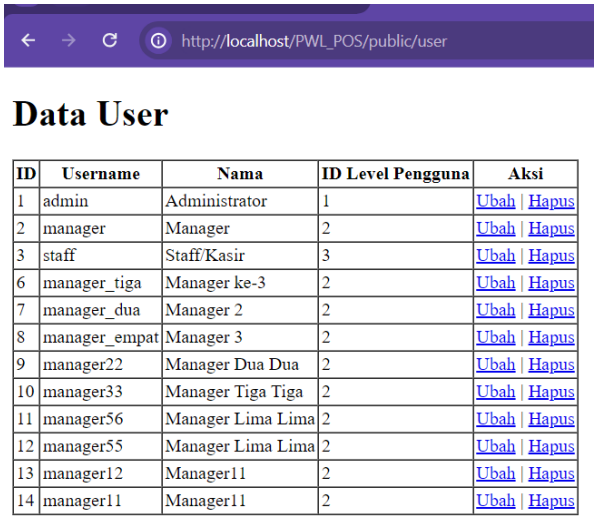
1. Buka file *view* pada `user.blade.php` dan buat scritpnya menjadi seperti di bawah ini

```
resources > views > user.blade.php > html > body > table > tr > td > a
<!DOCTYPE html>
<html>
    <head>
        <title>Data User</title>
    </head>
    <body>
        <h1>Data User</h1>
        <table border="1" cellpadding="2" cellspacing="0">
            <tr>
                <th>ID</th>
                <th>Username</th>
                <th>Nama</th>
                <th>ID Level Pengguna</th>
                <th>Aksi</th>
                {{-- <th>Jumlah Pengguna</th> --}}
            </tr>
            @foreach ($data as $d)
            <tr>
                <td>{{ $d->user_id }}</td>
                <td>{{ $d->username }}</td>
                <td>{{ $d->nama }}</td>
                <td>{{ $d->level_id }}</td>
                <td><a href="/PWL_POS/public/user/ubah/{{ $d->user_id }}">Ubah</a> |
                    <a href="/PWL_POS/public/user/hapus/{{ $d->user_id }}">Hapus</a></td>
            </tr>
            @endforeach
        </table>
    </body>
</html>
```

2. Buka file controller pada *UserController.php* dan buat scriptnya untuk *read* menjadi seperti di bawah ini

```
app > Http > Controllers > UserController.php > UserController > index
  9   class UserController extends Controller
 12       public function index(){
107           $user = UserModel::all();
108           return view('user', ['data' => $user]);
109       }
110   }
```

3. Simpan kode program Langkah 1 dan 2. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



Terdapat tabel yang menampilkan record user beserta aksi ubah | hapus

4. Langkah berikutnya membuat *create* atau tambah data user dengan cara bikin file baru pada *view* dengan nama `user_tambah.blade.php` dan buat scriptnya menjadi seperti di bawah ini

```
resources > views > user_tambah.blade.php > body > form
<body>
    <h1>Form Tambah Data User</h1>
    <form method="post" action="/PWL_POS/public/user/tambah_simpan">

        {{ csrf_field() }}

        <label>Username</label>
        <input type="text" name="username" placeholder="Masukkan Username">
        <br>
        <label>Nama</label>
        <input type="text" name="nama" placeholder="Masukkan nama">
        <br>
        <label>Password</label>
        <input type="password" name="password" placeholder="Masukkan Password">
        <br>
        <label>Level ID</label>
        <input type="number" name="level_id" placeholder="Masukkan ID Level">
        <br><br>
        <input type="submit" class="btn btn-success" value="Simpan">
    </form>
</body>
```

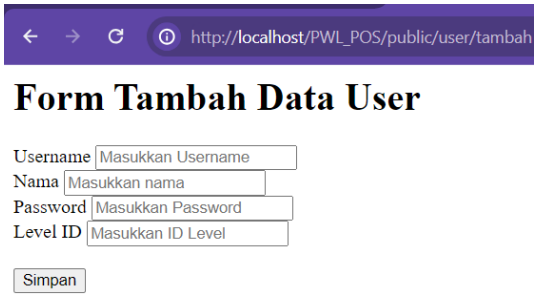
5. Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```
27 | Route::get('/user/tambah', [UserController::class, 'tambah']);
```

6. Tambahkan *script* pada controller dengan nama file *UserController.php*. Tambahkan *script* dalam class dan buat method baru dengan nama tambah dan diletakan di bawah method index seperti gambar di bawah ini

```
class UserController extends Controller

    public function tambah(){
        return view('user_tambah');
    }
}
```

7. Simpan kode program Langkah 4 s/d 6. Kemudian jalankan pada *browser* dan klik link "+ **Tambah User**" amati apa yang terjadi dan beri penjelasan dalam laporan



Akan muncul form yang digunakan untuk manambah data user.

8. Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

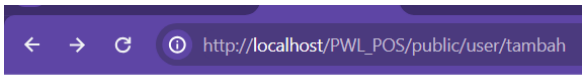
```
Route::post('/user/tambah_simpan', [UserController::class, 'tambah_simpan']);
```

9. Tambahkan *script* pada controller dengan nama file *UserController.php*. Tambahkan *script* dalam class dan buat method baru dengan nama tambah_simpan dan diletakan di bawah method tambah seperti gambar di bawah ini

```
app > Http > Controllers > UserController.php > UserController > tambah_simpan
class UserController extends Controller
    public function tambah_simpan(Request $request){
        UserModel::create([
            'username' => $request->username,
            'nama' => $request->nama,
            'password' => Hash::make('$request->password'),
            'level_id' => $request->level_id
        ]);

        return redirect('/user');
    }
}
```

10. Simpan kode program Langkah 8 dan 9. Kemudian jalankan link `localhost:8000/user/tambah` atau `localhost/PWL_POS/public/user/tambah` pada *browser*

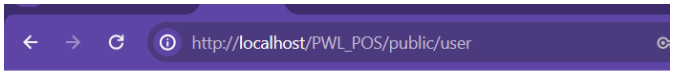
dan input formnya dan simpan, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan



Form Tambah Data User

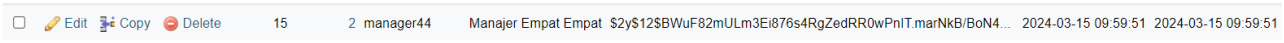
Username manager44
Nama Manajer Empat Empat
Password •••••
Level ID 2

Simpan



Data User

ID	Username	Nama	ID Level Pengguna	Aksi
1	admin	Administrator	1	Ubah | Hapus
2	manager	Manager	2	Ubah | Hapus
3	staff	Staff/Kasir	3	Ubah | Hapus
6	manager_tiga	Manager ke-3	2	Ubah | Hapus
7	manager_dua	Manager 2	2	Ubah | Hapus
8	manager_empat	Manager 3	2	Ubah | Hapus
9	manager22	Manager Dua Dua	2	Ubah | Hapus
10	manager33	Manager Tiga Tiga	2	Ubah | Hapus
11	manager56	Manager Lima Lima	2	Ubah | Hapus
12	manager55	Manager Lima Lima	2	Ubah | Hapus
13	manager12	Manager11	2	Ubah | Hapus
14	manager11	Manager11	2	Ubah | Hapus
15	manager44	Manajer Empat Empat	2	Ubah | Hapus



Akan terjadi proses penambahan data yang diinputkan lewat web. Data yang diinput akan masuk ke database dan ditampilkan.

11. Langkah berikutnya membuat *update* atau ubah data user dengan cara bikin file baru pada *view* dengan nama `user_ubah.blade.php` dan buat scriptnya menjadi seperti di bawah ini

```
resources > views > user_ubah.blade.php > body
<body>
    <h1>Form Ubah Data User</h1>
    <a href="/user">Kembali</a>
    <br><br>

    <form method="post" action="/PWL_POS/public/user/ubah_simpan/{{ $data->user_id }}">

        {{ csrf_field() }}
        {{ method_field('PUT') }}

        <label>Username</label>
        <input type="text" name="username" placeholder="Masukkan Username" value="{{ $data->username }}"_>
        <br>
        <label>Nama</label>
        <input type="text" name="nama" placeholder="Masukkan nama" value="{{ $data->nama }}">
        <br>
        <label>Password</label>
        <input type="password" name="password" placeholder="Masukkan Password" value="{{ $data->password }}">
        <br>
        <label>Level ID</label>
        <input type="number" name="level_id" placeholder="Masukkan ID Level" value="{{ $data->level_id }}">
        <br><br>
        <input type="submit" class="btn btn-success" value="Simpan">
    </form>
</body>
```

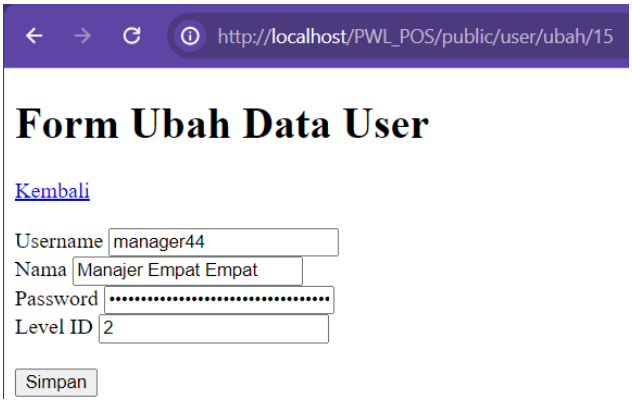
12. Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```
29  Route::get('/user/ubah/{id}', [UserController::class, 'ubah']);
```

13. Tambahkan *script* pada controller dengan nama file `UserController.php`. Tambahkan *script* dalam class dan buat method baru dengan nama ubah dan diletakan di bawah method tambah_simpan seperti gambar di bawah ini

```
126     public function ubah($id){
127         $user = UserModel::find($id);
128         return view('user_ubah', ['data' => $user]);
129     }
```

14. Simpan kode program Langkah 11 sd 13. Kemudian jalankan pada *browser* dan klik link "**Ubah**" amati apa yang terjadi dan beri penjelasan dalam laporan



Akan muncul form ubah data yang berisi data sebelumnya, namun belum bisa disimpan

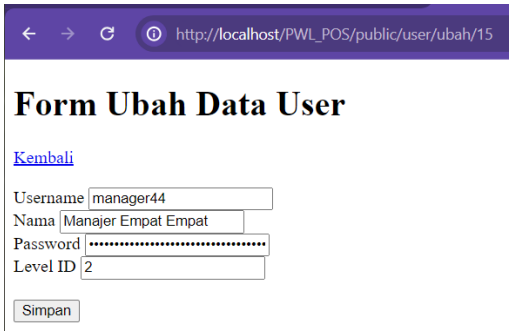
15. Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```
30  Route::put('/user/ubah_simpan/{id}', [UserController::class, 'ubah_simpan']);
```

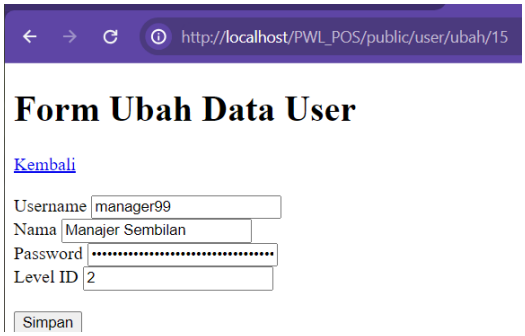
16. Tambahkan *script* pada controller dengan nama file `UserController.php`. Tambahkan *script* dalam class dan buat method baru dengan nama ubah_simpan dan diletakan di bawah method ubah seperti gambar di bawah ini

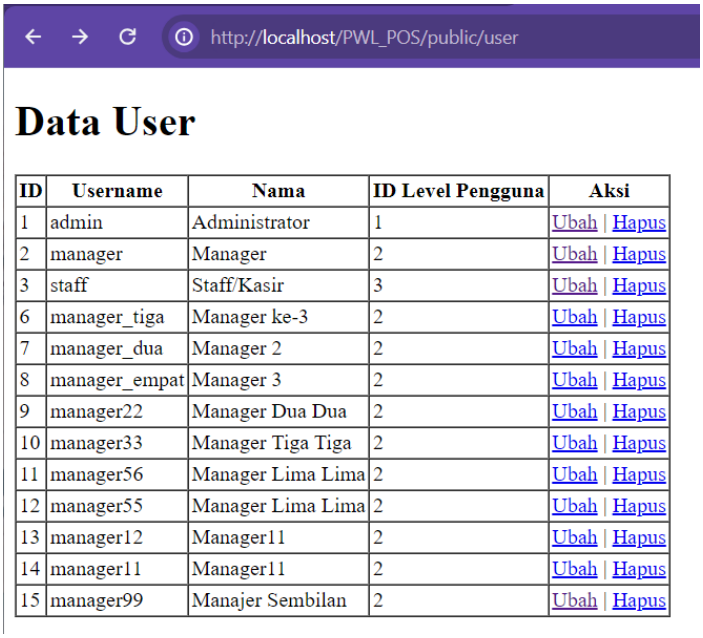
```
public function ubah_simpan($id, Request $request){
    $user = UserModel::find($id);

    $user->username = $request->username;
    $user->nama = $request->nama;
    $user->password = Hash::make('$request->password');
    $user->level_id = $request->level_id;

    $user->save();

    return redirect('/user');
}
```

17. Simpan kode program Langkah 15 dan 16. Kemudian jalankan link `localhost:8000/user/ubah/1` atau `localhost/PWL_POS/public/user/ubah/1` pada *browser* dan ubah input formnya dan klik tombol ubah, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan



Diubah menjadi





Manager44 berubah menjadi manajer99.

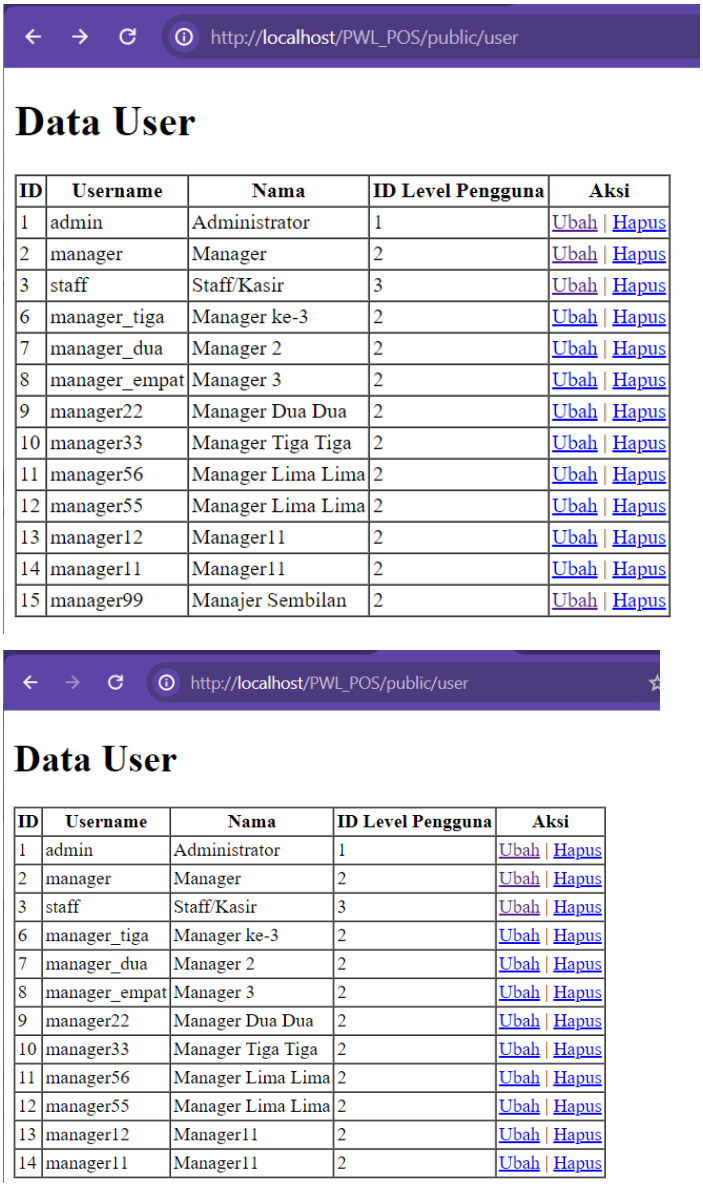
18. Berikut untuk langkah *delete* . Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```
31  Route::get('/user/hapus/{id}', [UserController::class, 'hapus']);
```

19. Tambahkan *script* pada controller dengan nama file `UserController.php`. Tambahkan *script* dalam class dan buat method baru dengan nama hapus dan diletakan di bawah method ubah_simpan seperti gambar di bawah ini

```
144     public function hapus($id){
145         $user = UserModel::find($id);
146         $user->delete();
147
148         return redirect('/user');
149     }
150 }
```

20. Simpan kode program Langkah 18 dan 19. Kemudian jalankan pada *browser* dan klik tombol hapus, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan



Data manager99 terhapus

21. Laporkan hasil Praktikum-2.6 ini dan *commit* perubahan pada *git*.

Praktikum 2.7 – *Relationships*

One to One

Hubungan satu-ke-satu adalah tipe hubungan database yang sangat mendasar. Misalnya, suatu `Usermodel` mungkin dikaitkan dengan satu `Phone` model. Untuk mendefinisikan hubungan ini,

kita akan menempatkan `phone` metode pada `User` model. Metode tersebut `phone` harus memanggil `hasOne` metode tersebut dan mengembalikan hasilnya. Metode ini `hasOne` tersedia untuk model Anda melalui kelas dasar model `Illuminate\Database\Eloquent\Model`:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasOne;

class User extends Model
{
    /**
     * Get the phone associated with the user.
     */
    public function phone(): HasOne
    {
        return $this->hasOne(Phone::class);
    }
}
```

Mendefinisikan Kebalikan dari Hubungan *One-to-one*

Jadi, kita dapat mengakses `Phone` model dari `User` model kita. Selanjutnya, mari kita tentukan hubungan pada `Phone` model yang memungkinkan kita mengakses pengguna pemilik telepon.
Kita dapat mendefinisikan kebalikan dari suatu `hasOne` hubungan menggunakan `belongsTo` metode:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Phone extends Model
{
    /**
     * Get the user that owns the phone.
     */
    public function user(): BelongsTo
    {
        return $this->belongsTo(User::class);
    }
}
```

One to Many

Hubungan satu-ke-banyak digunakan untuk mendefinisikan hubungan di mana satu model adalah induk dari satu atau lebih model turunan. Misalnya, postingan blog mungkin memiliki jumlah komentar yang tidak terbatas. Seperti semua hubungan Eloquent lainnya, hubungan satu-ke-banyak ditentukan dengan mendefinisikan metode pada model Eloquent Anda:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Post extends Model
{
    /**
     * Get the comments for the blog post.
     */
    public function comments(): HasMany
    {
        return $this->hasMany(Comment::class);
    }
}
```

One to Many (Inverse) / Belongs To

Sekarang kita dapat mengakses semua komentar postingan, mari kita tentukan hubungan agar komentar dapat mengakses postingan induknya. Untuk menentukan invers suatu `hasMany` hubungan, tentukan metode hubungan pada model anak yang memanggil `belongsTo` tersebut:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class Comment extends Model
{
    /**
     * Get the post that owns the comment.
     */
    public function post(): BelongsTo
    {
        return $this->belongsTo(Post::class);
    }
}
```

1. Buka file model pada `UserModel.php` dan tambahkan scritpnya menjadi seperti di bawah ini

app > Models > UserModel.php > UserModel > level

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user'; //Mendefinisikan nama tabel yang digunakan oleh model ini
    protected $primaryKey = 'user_id'; //mendefinisikan primary key dari tabel yang dignakan
    //protected $fillable = ['level_id', 'username', 'nama', 'password'];
    protected $fillable = ['level_id', 'username', 'nama', 'password'];

    public function level():BelongsTo{
        return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
    }
}
```

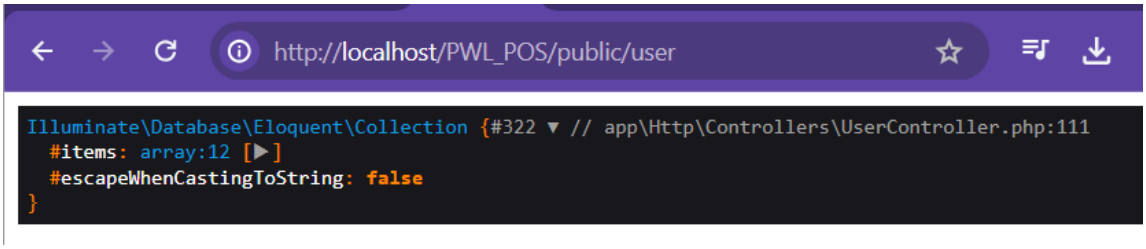
```
PS C:\laragon\www\PWL_POS> php artisan make:model LevelModel

  INFO  Model [C:\laragon\www\PWL_POS\app\Models\LevelModel.php] created successfully.
```

```php
app > Models > LevelModel.php > LevelModel > user
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class LevelModel extends Model
{
    use HasFactory;
    protected $table = 'm_level'; //Mendefinisikan nama tabel yang digunakan oleh model ini
    protected $primaryKey = 'level_id'; //mendefinisikan primary key dari tabel yang dignakan

    protected $fillable = ['level_id', 'level_kode', 'level_nama'];

    public function user():HasMany{
        return $this->hasMany(UserModel::class, 'level_id', 'level_id');
    }
}
```

2. Buka file controller pada `UserController.php` dan ubah method *script* menjadi seperti di bawah ini

```php
app > Http > Controllers > UserController.php > UserController > index
class UserController extends Controller
    public function index(){
        $user = UserModel::with('level')->get();
        dd($user);
    }
```

3. Simpan kode program Langkah 2. Kemudian jalankan link pada *browser*, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan



Mereturnkan bahwa terdapat 12 objek didalam database m_user

4. Buka file controller pada UserController.phpdan ubah method script menjadi seperti dibawah ini

```php
$user = UserModel::with('level')->get();
return view('user', ['data' => $user]);
```

5. Buka file view pada user.blade.phpdan ubah script menjadi seperti di bawah ini

resources > views > user.blade.php > html > body > table > tr > th

```
<!DOCTYPE html>
<html>
    <head>
        <title>Data User</title>
    </head>
    <body>
        <h1>Data User</h1>
        <table border="1" cellpadding="2" cellspacing="0">
            <tr>
                <th>ID</th>
                <th>Username</th>
                <th>Nama</th>
                <th>ID Level Pengguna</th>
                <th>Kode Level </th>
                <th>Nama Level </th>
                <th>Aksi</th>
                {{-- <th>Jumlah Pengguna</th> --}}
            </tr>
            @foreach ($data as $d)
            <tr>
                <td>{{ $d->user_id }}</td>
                <td>{{ $d->username }}</td>
                <td>{{ $d->nama }}</td>
                <td>{{ $d->level_id }}</td>
                <td>{{ $d->level->level_kode }}</td>
                <td>{{ $d->level->level_nama }}</td>
                <td><a href="/PWL_POS/public/user/ubah/{{ $d->user_id }}">Ubah</a> |
                    <a href="/PWL_POS/public/user/hapus/{{ $d->user_id }}">Hapus</a></td>
            </tr>
            @endforeach
        </table>
    </body>
</html>
```

6. Simpan kode program Langkah 4 dan 5. Kemudian jalankan link pada browser, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan

http://localhost/PWL_POS/public/user

Data User

ID	Username	Nama	ID Level Pengguna	Kode Level	Nama Level	Aksi
1	admin	Administrator	1	ADM	Administrator	Ubah | Hapus
2	manager	Manager	2	MNG	Manager	Ubah | Hapus
3	staff	Staff/Kasir	3	STF	Staff/Kasir	Ubah | Hapus
6	manager_tiga	Manager ke-3	2	MNG	Manager	Ubah | Hapus
7	manager_dua	Manager 2	2	MNG	Manager	Ubah | Hapus
8	manager_empat	Manager 3	2	MNG	Manager	Ubah | Hapus
9	manager22	Manager Dua Dua	2	MNG	Manager	Ubah | Hapus
10	manager33	Manager Tiga Tiga	2	MNG	Manager	Ubah | Hapus
11	manager56	Manager Lima Lima	2	MNG	Manager	Ubah | Hapus
12	manager55	Manager Lima Lima	2	MNG	Manager	Ubah | Hapus
13	manager12	Manager11	2	MNG	Manager	Ubah | Hapus
14	manager11	Manager11	2	MNG	Manager	Ubah | Hapus

7. Laporkan hasil Praktikum-2.7 ini dan *commit* perubahan pada *git*.

**** Sekian, dan selamat belajar ****