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)

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

Eloquent ORM adalah sebuah fitur dalam framework Laravel yang memungkinkan pengembang untuk berinteraksi dengan database menggunakan objek-objek PHP. Ini mengubah interaksi dengan database menjadi lebih mudah dan lebih ekspresif, serta mengikuti pola-pola desain aktiv record. Secara khusus, Eloquent memungkinkan Anda untuk mendefinisikan model-model PHP yang merepresentasikan tabel-tabel dalam basis data Anda. Setiap model biasanya berkorespondensi dengan sebuah tabel dalam database, dan instance dari model ini dapat digunakan untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada data dalam tabel tersebut.

Hubungan antara Eloquent ORM dan file model di Laravel adalah sebagai berikut:

1. Definisi Model: File model dalam Laravel adalah kelas-kelas PHP yang digunakan untuk merepresentasikan tabel-tabel dalam database. Setiap model biasanya terkait dengan satu tabel dalam database, dan nama model biasanya sesuai dengan nama tabel dalam bentuk singular. Misalnya, jika Anda memiliki tabel "users", Anda mungkin akan memiliki model yang disebut "User".
2. Menggunakan Eloquent: Dalam file model, Anda dapat mendefinisikan hubungan antar model, validasi data, dan perilaku lainnya menggunakan fitur-fitur yang disediakan oleh Eloquent ORM. Anda dapat menentukan koneksi database, nama tabel, dan kolom-kolom yang dapat diisi secara massal (mass assignable), serta menentukan hubungan antar model seperti "one-to-one", "one-to-many", "many-to-many", dan sebagainya.
3. Operasi CRUD: Dengan menggunakan Eloquent ORM, Anda dapat dengan mudah melakukan operasi CRUD pada data dalam database. Anda dapat membuat, membaca, memperbarui, dan menghapus data menggunakan metode-metode yang disediakan oleh Eloquent, seperti create(), find(), save(), update(), delete(), dan banyak lagi.
4. Query Builder: Eloquent juga menyediakan Query Builder yang kuat, yang memungkinkan Anda untuk menulis query kompleks menggunakan sintaks yang ekspresif dan mudah dipahami. Anda dapat menambahkan kriteria-kriteria pencarian, mengurutkan hasil, melakukan join antar tabel, dan melakukan banyak lagi dengan mudah menggunakan metode-metode yang disediakan.

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

```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';
    protected $primaryKey = 'user_id';
    /**
     * The attributes that are mass assignable
     *
     * @var array
     */
    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
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $data = [
            'level_id' => 2,
            'username' => 'manager_dua',
            'nama' => 'Manager 2',
            'password' => Hash::make('12345')
        ];
        UserModel::create($data);

        $user = UserModel::all();
        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
23	customer-1	Pelanggan Pertama	4
24	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';
    protected $primaryKey = 'user_id';
    /**
     * The attributes that are mass assignable
     *
     * @var array
     */
    protected $fillable = ['level_id', 'username', 'nama', 'password'];
}
```

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $data = [
            'level_id' => 2,
            'username' => 'manager_tiga',
            'nama' => 'Manager 3',
            'password' => Hash::make('12345')
        ];
        UserModel::create($data);

        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }
}
```

6. Simpan kode program Langkah 4 dan 5. Kemudian jalankan 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
23	customer-1	Pelanggan Pertama	4
24	manager_dua	Manager 2	2
25	manager_tiga	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).

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

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::find(1);
        return view('user', ['data' => $user]);
    }
}
```

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

```php
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <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>
        </tr>
        <tr>
            <td>{{ $data->user_id }}</td>
            <td>{{ $data->username }}</td>
            <td>{{ $data->nama }}</td>
            <td>{{ $data->level_id }}</td>
        </tr>
    </table>
</body>
</html>
```

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

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::where('level_id', 1)->first();
        return view('user', ['data' => $user]);
    }
}
```

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

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstWhere('level_id', 1);
        return view('user', ['data' => $user]);
    }
}
```

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

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:

```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
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::findOr(1, ['username', 'nama'], function (){
            abort(404);
        });

        return view('user', ['data' => $user]);
    }
}
```

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	

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::findOr(20, ['username', 'nama'], function (){
            abort(404);
        });

        return view('user', ['data' => $user]);
    }
}
```

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

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::findOrFail(1);
        return view('user', ['data' => $user]);
    }
}
```

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

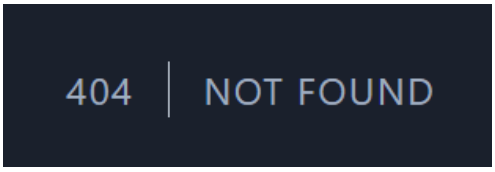
Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::where('username', 'manager9')->firstOrFail();
        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



5. Laporkan hasil Praktikum-2.2 ini dan *commit* perubahan pada *git*.

Praktikum 2.3 – *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

```
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::where('level_id', 2)->count();
        dd($user);
        return view('user', ['data' => $user]);
    }
}
```

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

```
3 // app\Http\Controllers\UserController.php:14
```

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.php M | user.blade.php M

app > Http > Controllers > UserController.php > ...

```
1   <?php
2
3   namespace App\Http\Controllers;
4
5   use App\Models\UserModel;
6   use Illuminate\Http\Request;
7   use Illuminate\Support\Facades\Hash;
8
9   class UserController extends Controller
10  {
11      public function index()
12      {
13          $user = UserModel::where('level_id', 2)->count();
14          return view('user', ['data' => $user]);
15      }
16  }
```

UserController.php M | user.blade.php M

resources > views > user.blade.php > ...

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Data User</title>
</head>
<body>
    <h1>Data User</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <th>Jumlah Pengguna</th>
        </tr>
        <tr>
            <td>{{ $data }}</td>
        </tr>
    </table>
</body>
</html>
```

Data User

Jumlah Pengguna
3

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrCreate(
            [
                'username' => 'manager',
                'nama' => 'Manager',
            ],
        );
        return view('user', ['data' => $user]);
    }
}
```

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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <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>
        </tr>
        <tr>
            <td>{{ $data->user_id }}</td>
            <td>{{ $data->username }}</td>
            <td>{{ $data->nama }}</td>
            <td>{{ $data->level_id }}</td>
        </tr>
    </table>
</body>
</html>
```

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

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

```
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrCreate(
            [
                'username' => 'manager22',
                'nama' => 'Manager Dua Dua',
                'password' => Hash::make('12345'),
                'level_id' => 2
            ],
        );
        return view('user', ['data' => $user]);
    }
}
```

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

Data User

ID	Username	Nama	ID Level Pengguna
26	manager22	Manager Dua Dua	2

	user_id	level_id	username	nama	password	created_at	updated_at
Edit Copy Delete	1	1	admin	Administrator	$2y$12$Q3sMfxqwmPOB1V2sza2bQedvyzPJ0dwi1/YdRL8k5KY...	NULL	NULL
Edit Copy Delete	2	2	manager	Manager	$2y$12$hQBsQ6Fhola/O0KLacyXler76u6xJ8lvEEFETw7SkII...	NULL	NULL
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$Pqcbw4JJwv/Q/8CwbGnnbuFIQfxLjhkgV72CH8HcY1P...	NULL	NULL
Edit Copy Delete	23	4	customer-1	Pelanggan Pertama	$2y$12$flQv7isSpxUNrQIbBneqa.2csoU2ZNseGTR5uluDBbD...	NULL	2024-03-12 10:05:44
Edit Copy Delete	24	2	manager_dua	Manager 2	$2y$12$9aNfLnSEI5wP8IzGNY0B2.Vkeoq9/GF7Fn1PIJXWNL6...	2024-03-20 05:42:51	2024-03-20 05:42:51
Edit Copy Delete	25	2	manager_tiga	Manager 3	$2y$12$wM.1x6lw.DAYVLn/A42fJ.DVFIhCG6mW/XM0LPptXDp...	2024-03-20 05:48:32	2024-03-20 05:48:32
Edit Copy Delete	26	2	manager22	Manager Dua Dua	$2y$12$FO3MrtF3fHNM4N1T1IM5Se3XNVetQQy3mHgEq8OusDY...	2024-03-20 07:54:43	2024-03-20 07:54:43

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrNew(
            [
                'username' => 'manager',
                'nama' => 'Manager',
            ],
        );
        return view('user', ['data' => $user]);
    }
}
```

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

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrNew(
            [
                'username' => 'manager33',
                'nama' => 'Manager Tiga Tiga',
                'password' => Hash::make('12345'),
                'level_id' => 2
            ],
        );
        return view('user', ['data' => $user]);
    }
}
```

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

	user_id	level_id	username	nama	password	created_at	updated_at
Edit Copy Delete	1	1	admin	Administrator	$2y$12$Q3sMfxqwmPOB1V2sza2bQedvyzPJ0dwi1/YdRL8k5KY...	NULL	NULL
Edit Copy Delete	2	2	manager	Manager	$2y$12$hQBsQ6FhoIa/O0KLacyXIer76u6xJ8lvEEFETw7SkII...	NULL	NULL
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$Pqcbw4JJwv/Q/8CwbGnnbuFlQfxLjhkgV72CH8HcY1P...	NULL	NULL
Edit Copy Delete	23	4	customer-1	Pelanggan Pertama	$2y$12$flQv7isSpxUNrQlbBneqa.2csoU2ZNseGTR5uluDBbD...	NULL	2024-03-12 10:05:44
Edit Copy Delete	24	2	manager_dua	Manager 2	$2y$12$9aNfLnSEl5wP8lzGNY0B2.Vkeoq9/GF7Fn1PlJXWNL6...	2024-03-20 05:42:51	2024-03-20 05:42:51
Edit Copy Delete	25	2	manager_tiga	Manager 3	$2y$12$wM.1x6lw.DAYVLn/A42fJ.DVFlhCG6mW/XM0LPptXDp...	2024-03-20 05:48:32	2024-03-20 05:48:32
Edit Copy Delete	26	2	manager22	Manager Dua Dua	$2y$12$FO3MrtF3fHNM4N1T1lM5Se3XNVetQQy3mHgEq8OusDY...	2024-03-20 07:54:43	2024-03-20 07:54:43

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrNew(
            [
                'username' => 'manager33',
                'nama' => 'Manager Tiga Tiga',
                'password' => Hash::make('12345'),
                'level_id' => 2
            ],
        );
        $user -> save();

        return view('user', ['data' => $user]);
    }
}
```

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
27	manager33	Manager Tiga Tiga	2

	user_id	level_id	username	nama	password	created_at	updated_at
Edit Copy Delete	1	1	admin	Administrator	$2y$12$Q3sMfxqwmPOB1V2sza2bQedvyzPJ0dwi1/YdRL8k5KY...	NULL	NULL
Edit Copy Delete	2	2	manager	Manager	$2y$12$hQBsQ6Fhola/O0KLacyXIer76u6xJ8lvEEFETw7SkII...	NULL	NULL
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$Pqcbw4JJwv/Q/8OwbGnnbuFlQfxLjhkgV72CH8HcY1P...	NULL	NULL
Edit Copy Delete	23	4	customer-1	Pelanggan Pertama	$2y$12$flQv7isSpxUNrQlbBneqa.2csoU2ZNseGTR5uluDBbD...	NULL	2024-03-12 10:05:44
Edit Copy Delete	24	2	manager_dua	Manager 2	$2y$12$9aNfLnSEl5wP8lzGNY0B2.Vkeoq9/GF7Fn1PIJXWNL6...	2024-03-20 05:42:51	2024-03-20 05:42:51
Edit Copy Delete	25	2	manager_tiga	Manager 3	$2y$12$wM.1x6lw.DAYVLn/A42fJ.DVFIhCG6mW/XM0LPptXDp...	2024-03-20 05:48:32	2024-03-20 05:48:32
Edit Copy Delete	26	2	manager22	Manager Dua Dua	$2y$12$FO3MrtF3fHNM4N1T1IM5Se3XNVetQQy3mHgEq8OusDY...	2024-03-20 07:54:43	2024-03-20 07:54:43
Edit Copy Delete	27	2	manager33	Manager Tiga Tiga	$2y$12$Z5OoR5KLfD9fUqpQKTlzSe.OwSqq8b/H51ZwUWzgmko...	2024-03-21 23:57:47	2024-03-21 23:57:47

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::create([
            'username' => 'manager5',
            'nama' => 'Manager55',
            'password' => Hash::make('12345'),
            'level_id' => 2,
        ]);

        $user->username = 'manager56';

        $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
        $user->isClean(); // True
        dd($user->isDirty());
    }
}
```

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

```
false // app\Http\Controllers\UserController.php:36
```

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

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
    }
}
```

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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Data User</title>
</head>
<body>
    <h1>Data User</h1>
    <a href="/user/tambah">+ Tambah User</a>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <td>ID</td>
            <td>Username</td>
            <td>Nama</td>
            <td>ID Level Pengguna</td>
            <td>Aksi</td>
        </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="/user/ubah/{{ $d->user_id }}">Ubah</a> | <a href="/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

```
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }
}
```

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

Data User

+ Tambah 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
23	customer-1	Pelanggan Pertama	4	Ubah | Hapus
24	manager_dua	Manager 2	2	Ubah | Hapus
25	manager_tiga	Manager 3	2	Ubah | Hapus
26	manager22	Manager Dua Dua	2	Ubah | Hapus
27	manager33	Manager Tiga Tiga	2	Ubah | Hapus
31	manager56	Manager55	2	Ubah | Hapus
32	manager12	Manager11	2	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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h1>Form Tambah Data User</h1>
    <form method="post" action="/user/tambah_simpan">
        {{ csrf_field() }}

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

    </form>
</body>
</html>
```

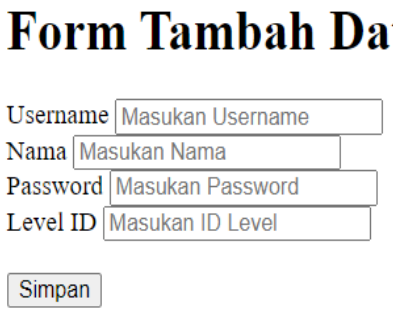
5. Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```php
<?php

use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;


Route::get('/', function () {
    return view('welcome');
});

Route::get('/level', [LevelController::class, 'index']);
Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);
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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }

    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



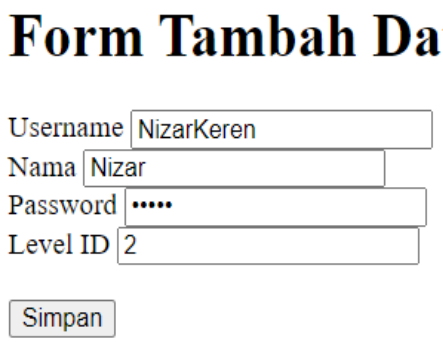
8. Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```php
<?php

use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;


Route::get('/', function () {
    return view('welcome');
});

Route::get('/level', [LevelController::class, 'index']);
Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);
Route::get('/user/tambah', [UserController::class, 'tambah']);
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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }

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

    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



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

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Document</title>
</head>
<body>
    <h1>Form Ubah Data User</h1>
    <a href="/user">Kembali</a>
    <br><br>

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

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

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

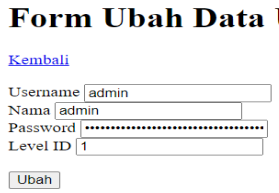
12. Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```
<?php

use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;


Route::get('/', function () {
    return view('welcome');
});

Route::get('/level', [LevelController::class, 'index']);
Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);
Route::get('/user/tambah', [UserController::class, 'tambah']);
Route::post('/user/tambah_simpan', [UserController::class, 'tambah_simpan']);
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

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

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



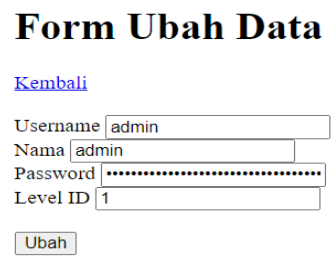
15. Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```php
<?php

use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;


Route::get('/', function () {
    return view('welcome');
});

Route::get('/level', [LevelController::class, 'index']);
Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);
Route::get('/user/tambah', [UserController::class, 'tambah']);
Route::post('/user/tambah_simpan', [UserController::class, 'tambah_simpan']);
Route::get('/user/ubah/{id}', [UserController::class, 'ubah']);
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

```php
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



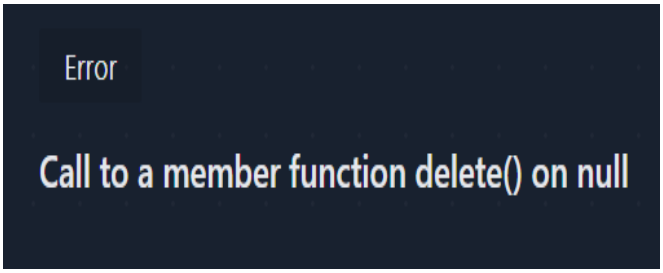
18. Berikut untuk langkah *delete* . Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```php
<?php

use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;


Route::get('/', function () {
    return view('welcome');
});

Route::get('/level', [LevelController::class, 'index']);
Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);
Route::get('/user/tambah', [UserController::class, 'tambah']);
Route::post('/user/tambah_simpan', [UserController::class, 'tambah_simpan']);
Route::get('/user/ubah/{id}', [UserController::class, 'ubah']);
Route::put('/user/ubah_simpan/{id}', [UserController::class, 'ubah_simpan']);
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

```php
public function hapus($id)
{
    $user = UserModel::find($id);
    $user->delete();

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

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



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);
    }
}
```

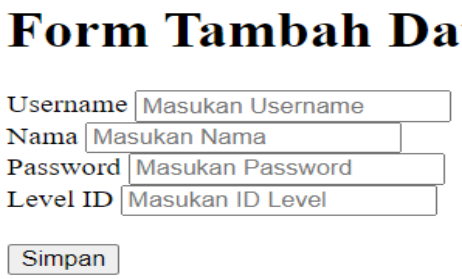
1. Buka file model pada `UserModel.php` dan tambahkan scritpnya menjadi seperti di bawah ini

```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';
    protected $primaryKey = 'user_id';
    /**
     * The attributes that are mass assignable
     *
     * @var array
     */
    protected $fillable = ['level_id', 'username', 'nama', 'password'];

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

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

```php
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



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

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