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

NAMA : KARTIKA TRI JULIANA
PROGRAM STUDI : D4 – Sistem Informasi Bisnis
KELAS : 2B
NIM : 2341760116

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

ORM (Object Relation Mapping) merupakan teknik yang merubah suatu table menjadi sebuah object yang nantinya mudah untuk digunakan. Object yang dibuat memiliki property yang sama dengan field — field yang ada pada table tersebut. ORM tersebut bertugas sebagai penghubung dan sekaligus mempermudah kita dalam membuat aplikasi yang menggunakan database relasional agar menjadikan tugas kita lebih efisien.

Kelebihan - Kelebihan Menggunakan ORM

1. Terdapat banyak fitur seperti transactions, connection pooling, migrations, seeds, streams, dan lain sebagainya.

2. perintah query memiliki kinerja yang lebih baik, daripada kita menulisnya secara manual.
3. Kita menulis model data hanya di satu tempat, sehingga lebih mudah untuk update, maintain, dan reuse the code.
4. Memungkinkan kita memanfaatkan OOP (object oriented programming) dengan baik

Di Laravel sendiri telah disediakan Eloquent ORM untuk mempermudah kita dalam melakukan berbagai macam query ke database, dan membuat pekerjaan kita menjadi lebih mudah karena tidak perlu menuliskan query sql yang panjang untuk memproses data.

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:

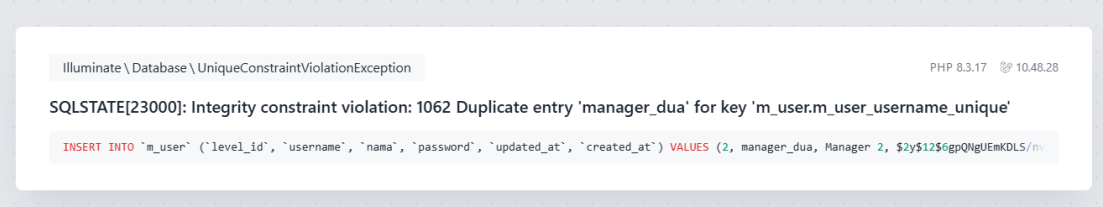
1. Buka file model dengan nama `UserModel.php` dan tambahkan `$fillable` seperti gambar di bawah ini

```
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 Illuminate\Http\Request;
use App\Models\UserModel;
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 `localhostPWL_POS/public/user` pada *browser* dan amati apa yang terjadi = terjadi error, karena terjadi duplikat untuk key m_user



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

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

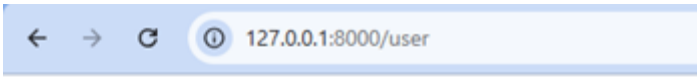
5. Ubah kembali file controller `UserController.php` seperti pada gambar di bawah hanya bagian array pada *$data*

```php
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

   = ketika password dihapus maka yang terjadi adalah menjadi normal karena tidak ada field yang ditolak oleh Laravel

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



Data User

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

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***. ***Mass Assignment*** adalah fitur canggih yang menyederhanakan proses pengaturan beberapa atribut model sekaligus, menghemat waktu dan tenaga. Pada praktikum ini, kita akan mengeksplorasi konsep penugasan massal di Laravel dan bagaimana hal itu dapat dimanfaatkan secara efektif untuk meningkatkan alur kerja pengembangan Anda.

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('level_id', 1)->first();

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

Praktikum 2.1 – Retrieving Single Models

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

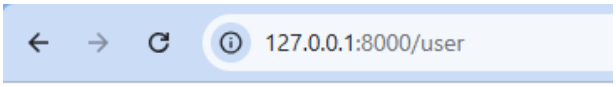
```php
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::find(1);
        return view('user', ['data' => $user]);
    }
}
```

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

```
<body>
    <h1>Data User</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <td>ID</td>
            <td>Username</td>
            <td>Nama</td>
            <td>ID Level Pengguna</td>
        </tr>
        <tr>
            <td>{{ $data->user_id }}</td>
            <td>{{ $data->username }}</td>
            <td>{{ $data->nama }}</td>
            <td>{{ $data->level_id }}</td>
        </tr>
    </table>
</body>
```

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

   = pada userController perintah yang digunakan adalah find(1) dimana array tersebut merujuk pada array 1 yang berisikan data dengan id 1 yakni dengan username admin



Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1

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

```php
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 pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan

   = hasilnya akan tetap sama seperti sebelumnya karena $user meminta unuk mencari dimana level_id = 1 pada query m_user serta penggunaan first() itu digunakan untuk memunculkan hasil 1 data pertama

Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1

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

```php
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 pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan
   = mencari data user di tabel _user dengan level_id = 1 dimana mengambil satu data pertama yang ditemukan, penggunaan firstWhere('level_id',1) adalah satu method langsung untuk mencari satu data pertama yang sesuai

Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
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 maka akan menjalankan didalam fungsi. Nilai yang dikembalikan oleh fungsi 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
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 pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan

   = findOr(1, [...], function () { ... }) digunakan mencari data user berdasarkan primary key (id = 1), ['username', 'nama'] menentukan kolom yang dimunculkan dan akan dilakukan pengembalian user, jika tidak ditemukan maka fungsi akan memunculkan abort (404) yang artinya error

Data User

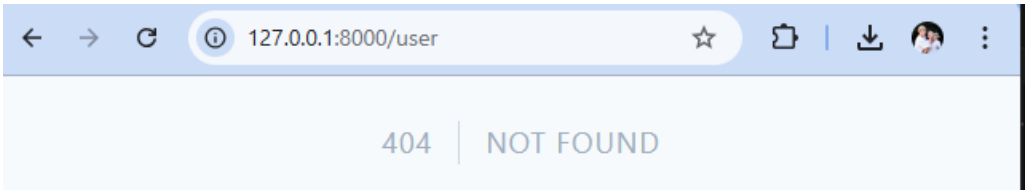
ID	Username	Nama	ID Level Pengguna
	admin	Administrator	
	admin	Administrator	
	admin	Administrator	
	admin	Administrator	
	admin	Administrator	
	admin	Administrator	

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

```php
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 pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan

    = terjadi error karena findOr(20,... digunakan mencari di primary key dengan nomor sekian, dan nomor 20 tidak ada pada primary key sehingga terjad error (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

```php
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

   = sama seperti kode sebelumnya, UserModel::findOrFail(1) mencari data yang dimana mencari id 1, findOrFail akan mengembalikan data jika ditemukan, jika tidak ditemukan maka akan memunculkan error 404

Data User

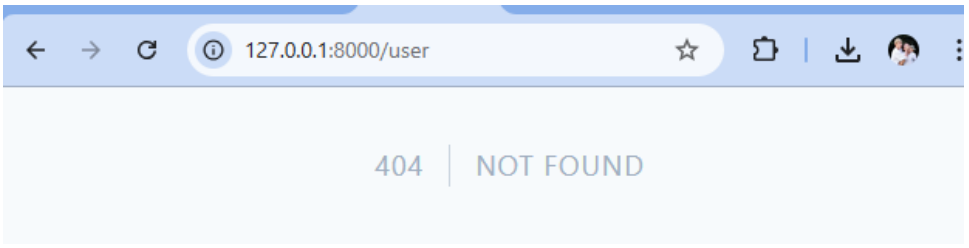
ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1
1	admin	Administrator	1

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

```php
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

   = terjadi error karena pemanggilan database diluar isi database, contohnya di database ada field username tetapi kode meminta untuk memunculkan username dengan isi manager9 yang tentunya tidak ada di database sehingga firstOrFail() akan mengembalikan nilai error karena tidak ditemukan data tersebut, semisal di ubah menjadi manager saja maka hasil akan muncul di browser



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

```
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

   = UserModel::where('level_id', 2) digunakan untuk mencari level_id = 2, count() untuk menghitung jumlah user yang diinginkan kode yakni id = 2. dd($user) nama lain nya adalah dump and die yang digunaan untuk menampilkan isi variabel dan menghentikan eksekusi. Hasil dibawah memunculkan dump dari variabel $user

```
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

Data User

Jumlah Pengguna
2

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

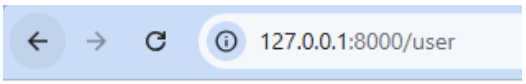
   Penjelasan : pada kode userController dd($user) dihapus untuk menghindari pemberhentian eksekusi, kemudian ubah di vier -> user.blade untuk diubah tampilannya menjadi sesuai yang diinginkan, kode diubah seperti berikut dan akan memunculkan hasil dimana count ata jumlah dari level_id = 2 ada 3 jumlahnya

   Kode :

```php
<?php

namespace App\Http\Controllers;

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

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

}
```

```blade
<!DOCTYPE html>
<html>
<head>
    <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>
```

   Hasil :



Data User

Jumlah Pengguna
3

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

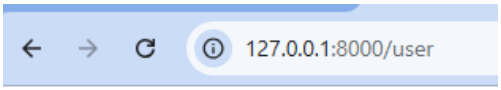
```php
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrCreate(
            [
                'username' => 'manager',
                'nama' => 'Manager',
            ],
        );

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

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

```blade
<body>
    <h1>Data User</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <td>ID</td>
            <td>Username</td>
            <td>Nama</td>
            <td>ID Level Pengguna</td>
        </tr>
        <tr>
            <td>{{ $data->user_id }}</td>
            <td>{{ $data->username }}</td>
            <td>{{ $data->nama }}</td>
            <td>{{ $data->level_id }}</td>
        </tr>
    </table>
</body>
```

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

   = memunculkan username dengan isi manager dan nama Manager, kemudian view yang diinginkan yakni memunculkan data yakni user_id, username, nama dan level_id sehingga diperoleh hasil seperti berikut



Data User

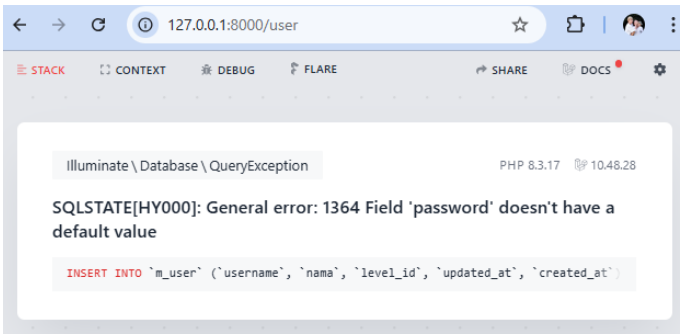
ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

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

```php
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]);
    }
}
```

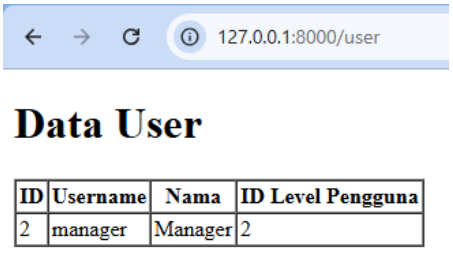
5. Simpan kode program Langkah 4. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan cek juga pada *phpMyAdmin* pada tabel `m_user` serta beri penjelasan dalam laporan

   = akan terjadi error karena username manager22 dan nama Manager dua dua tidak ada dalam database dan password tidak terdeteksi oleh field dalam laravel sehingga terjadi error



Solusinya adalah ubah isi dari username dan nama yang sesuai dengan isi database dan hapus line password maka akan muncul hasilnya

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



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

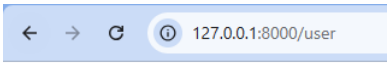
```php
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrNew(
            [
                'username' => 'manager',
                'nama' => 'Manager',
            ],
        );

        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

   = pada kode tersebut hasilnya tetap sama tetapi yang membedakan adalah firstOrCreate dan firstOrNew

   firstOrCreate : jika data tidak ada maka akan membuat dan menyimpan pada database

   firstOrNew : jika tidak ada maka hanya membuat instance baru tapi tidak menyimpannya dalam database



Data User

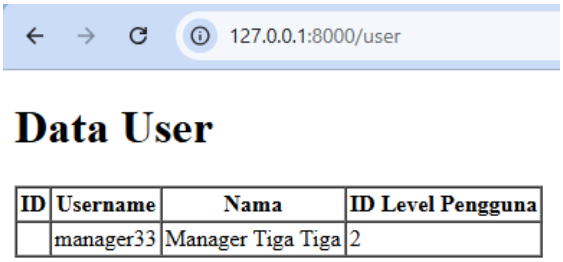
ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

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

```php
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

   = sesuai penjelasan diatas, firstOrNew hanya akan memunculkan data yang valid saja, contohnya username manager33 dan nama Manager Tiga Tiga tidak ada di database tetapi bisa meuncul karena field yang valid ada di level_id dengan nilai 2 maka yang dimunculkan adalah manager tetapi tampilan nya saja yang manager33 tetapi di databse hanya manager saja.

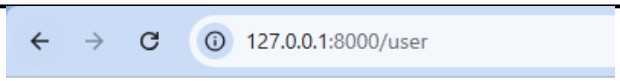


10. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```php
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]);
    }
}
```

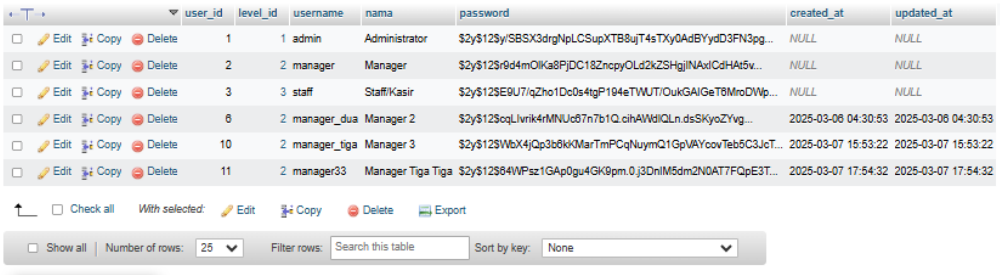
11. Simpan kode program Langkah 9. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan cek juga pada *phpMyAdmin* pada tabel `m_user` serta beri penjelasan dalam laporan

    = setelah dilakukan save manual maka secara otomatis akan membuat database baru



Data User

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



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:

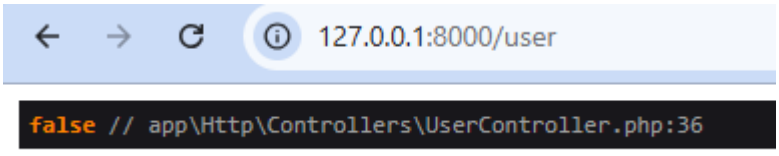
```php
$user = UserModel::create([
    'username' => 'manager44',
    'nama' => 'Manager44',
    'password' => Hash::make('12345'),
    'level_id' => 2,
]);

$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
```

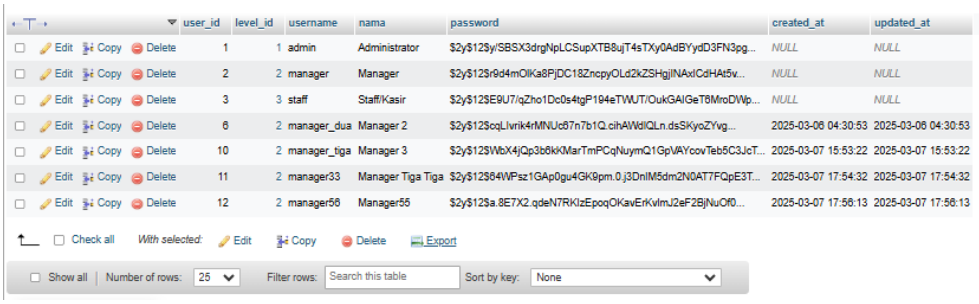
1. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```php
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::create([
            'username' => 'manager55',
            '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

   = setelah save akan menghasilkan false pada browser dikarenakan save tersebut merujuk ke kode isDirty(); yang artinya semua perubahan telah disimpan





	user_id	level_id	username	nama	password	created_at	updated_at
Edit Copy Delete	1	1	admin	Administrator	$2y$12$y/SBSX3drgNpLCSupXTB8ujT4sTXy0AdBYydD3FN3pg...	NULL	NULL
Edit Copy Delete	2	2	manager	Manager	$2y$12$r9d4mOlKa8PjDC18ZncpyOLd2kZSHgjlNAxlCdHAt5v...	NULL	NULL
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$E9U7/qZho1Dc0s4tgP194eTWJT/OukGAlGeT6MroDWp...	NULL	NULL
Edit Copy Delete	6	2	manager_dua	Manager 2	$2y$12$cqLIvrik4rMNUc87n7b1Q.cihAWdlQLn.dsSKyoZYvg...	2025-03-06 04:30:53	2025-03-06 04:30:53
Edit Copy Delete	10	2	manager_tiga	Manager 3	$2y$12$WbX4jQp3b6kKMarTmPCqNuymQ1GpVAYcovTeb5C3JcT...	2025-03-07 15:53:22	2025-03-07 15:53:22
Edit Copy Delete	11	2	manager33	Manager Tiga Tiga	$2y$12$64WPsz1GAp0gu4GK9pm.0.j3DnIM5dm2N0AT7FQpE3T...	2025-03-07 17:54:32	2025-03-07 17:54:32
Edit Copy Delete	12	2	manager56	Manager55	$2y$12$a.8E7X2.qdeN7RKIzEpoqOKavErKvlmJ2eF2BjNuOf0...	2025-03-07 17:56:13	2025-03-07 17:56:13

Check all With selected: Edit Copy Delete Export

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

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
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
        $user->wasChanged('nama'); // false
        dd($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

   = hasil true karena dalam wasChanged(['nama', 'username']) berubah, meskipun nama tidak berubah tetapi username berubah dan mengembalikan nilai true

```
true // app\Http\Controllers\UserController.php:29
```

Edit Copy Delete	12	2	manager56	Manager55	$2y$12$a.8E7X2.qdeN7RKIzEpoqOKavErKvlmJ2eF2BjNuOf0...	2025-03-07 17:56:13	2025-03-07 17:56:13
Edit Copy Delete	13	2	manager12	Manager11	$2y$12$CTMq4rt2fdNFSuzrkFqu8OaBUZR8rgblPxamB9Wmo3P...	2025-03-07 18:01:02	2025-03-07 18:01:03

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

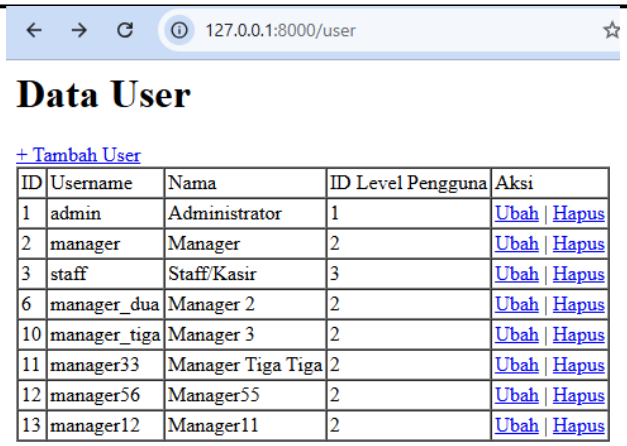
```
<body>
    <h1>Data User</h1>
    <a href="/user/tambah">+ Tambah User</a>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <td>ID</td>
            <td>Username</td>
            <td>Nama</td>
            <td>ID Level Pengguna</td>
            <td>Aksi</td>
        </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>
```

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

```
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

   = pada user.blade memunculkan semua tampilan dari user_id, username, nama, level_id dengan tiap barisnya akan diberikan aksi berubah ubah atau hapus, kemudian controller akan dipanggil dan muncul di browser seperti berikut



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_dua	Manager 2	2	Ubah | Hapus
10	manager_tiga	Manager 3	2	Ubah | Hapus
11	manager33	Manager Tiga Tiga	2	Ubah | Hapus
12	manager56	Manager55	2	Ubah | Hapus
13	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

```
<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>
```

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

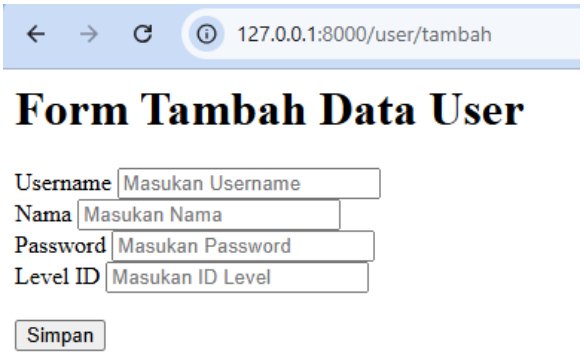
```
Route::get('/user/tambah', [UserController::class, 'tambah']);
```

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
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

   = pada langkah 4 digunakan untuk memperlihatkan view dari form dan menambahkan token csrf_field() sebagai perlindungan dari serangan, kemudian langkah 5 untuk menyambungkan ke route agar bisa m=dimunculkan pada browser, langkah 7 digunakan untuk aksi yang lebih dalam sehingga ketika menekan +tambah user maka akan merujuk ke halaman lain yakni user/tambah



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

```php
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
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

    = $request menunjukkan jika isi menyesuaikan pengguna tetapi untuk level_id karena dia foreign key maka harus sesuai dengan yang ada di database, yang artinya tidak oelh melebihi, misal level_id hanya sampai 3 maka harus memilih antara 1-3 sesuai username yang dipakai, manager dapat level_id = 2 maka sesuaikan

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
6	manager_dua	Manager 2	2	Ubah | Hapus
10	manager_tiga	Manager 3	2	Ubah | Hapus
11	manager33	Manager Tiga Tiga	2	Ubah | Hapus
12	manager56	Manager55	2	Ubah | Hapus
13	manager12	Manager11	2	Ubah | Hapus
16	Tika	Manager	2	Ubah | Hapus

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

```
<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>
```

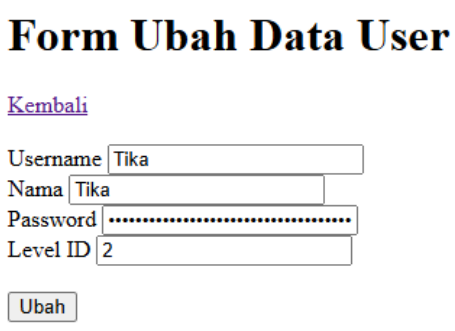
12. Tambahkan *script* pada *routes* dengan nama file `web.php`. Tambahkan seperti gambar di bawah ini

```
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

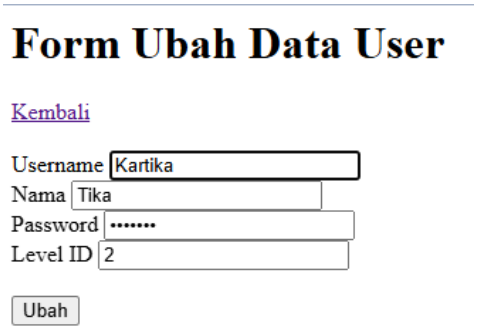
```
Route::put('/user/ubah_simpan/{id}', [UserController::class, 'ubah_simpan']);
```

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

    = disini saya mencoba mengubah dari data saya dengan mengubah username dan password dan berhasil dilakukan



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
6	manager_dua	Manager 2	2	Ubah | Hapus
10	manager_tiga	Manager 3	2	Ubah | Hapus
11	manager33	Manager Tiga Tiga	2	Ubah | Hapus
12	manager56	Manager55	2	Ubah | Hapus
13	manager12	Manager11	2	Ubah | Hapus
16	Kartika	Tika	2	Ubah | Hapus

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

```
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

    = setelah menekan hapus pada username Kartika maka yang terjadi adalah data tersebut secara otomatis akan terhapus dari tabel

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
6	manager_dua	Manager 2	2	Ubah | Hapus
10	manager_tiga	Manager 3	2	Ubah | Hapus
11	manager33	Manager Tiga Tiga	2	Ubah | Hapus
12	manager56	Manager55	2	Ubah | Hapus
13	manager12	Manager11	2	Ubah | Hapus

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 model `Levelmodel`. Untuk mendefinisikan hubungan ini, kita akan menempatkan `Levelmodel` metode pada model `Usermodel`. Metode tersebut `Levelmodel` 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;

You, 1 second ago | 1 author (You)
class UserModel extends Model
{
    public function level(): HasOne
    {
        return $this->hasOne(LevelModel::class);
    }
}
```

Mendefinisikan Kebalikan dari Hubungan *One-to-one*

Jadi, kita dapat mengakses model `Levelmodel` dari model `Usermodel` kita. Selanjutnya, mari kita tentukan hubungan pada model `Levelmodel` yang memungkinkan kita mengakses user. 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 LevelModel extends Model
{
    public function user(): BelongsTo
    {
        return $this->belongsTo(UserModel::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, 1 kategori mungkin memiliki jumlah barang 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 KategoriModel extends Model
{
    public function barang(): HasMany
    {
        return $this->hasMany(BarangModel::class, 'barang_id', 'barang_id');
    }
}
```

One to Many (Inverse) / Belongs To

Sekarang kita dapat mengakses semua barang, mari kita tentukan hubungan agar barang dapat mengakses kategori 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 BarangModel extends Model
{
    public function kategori(): BelongsTo
    {
        return $this->belongsTo(KategoriModel::class, 'kategori_id', 'kategori_id');
    }
}
```

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

```php
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, 'level_id', 'level_id');
    }
}
```

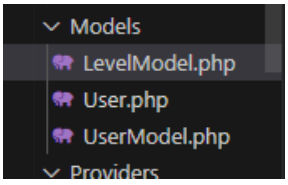
2. Buka file controller pada `UserController.php` dan ubah method *script* menjadi seperti di bawah ini

```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

   = sebelumnya saya mengalami error

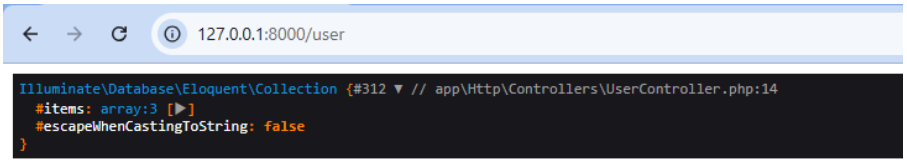
Sehingga solusinya adalah membuat file LevelModel pada Model yang berisikan kode berikut




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

namespace App\Models;

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

class LevelModel extends Model
{
    use HasFactory;

    protected $table = 'm_level';
    protected $guard = [];
}
```

Dan pada UserModel tambahkan kode berikut kemudian liat hasilnya di browser

```php
protected $primaryKey = 'user_id';
protected $table = 'm_level';
```



Penjelasan lainnya, pada arahannya memakai $fillable dimana digantikan oleh $guard dimana array tersebut masih kosong sehingga menghindari pengisian massal yang dilakukan oleh sistem maka di definisikan dengan $guard

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

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

5. Buka file view pada `user.blade.php` dan ubah *script* menjadi seperti di bawah ini

```php
<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>Kode Level</td>
            <td>Nama Level</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>{{ $d->level->level_kode }}</td>
                <td>{{ $d->level->level_nama }}</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>
```

6. Simpan kode program Langkah 4 dan 5. Kemudian jalankan link pada *browser*, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan
   = hanya memunculkan sebagian dari isi tabel karena pemanggilan yang dilakukan ada pada kode berikut.

```php
<td>{{ $d->level->level_kode }}</td>
<td>{{ $d->level->level_nama }}</td>
```

   Pemaggilan dilakukan karena diambil relai level yang berasal dari $user = UserModel::with('level')->get();

Data User

+ Tambah User

ID	Username	Nama	Level Kode	Level Nama	Aksi
			ADM	Administrator	Ubah | Hapus
			MNG	Manager	Ubah | Hapus
			STF	Staff/Kasir	Ubah | Hapus

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