



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

**Fallujah Ramadi C – 09**  
**SIB 2B**

## **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 pertemuan kali ini kita akan memahami tentang bagaimana 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 pelajari

**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



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\UserModel;
7 use Illuminate\Support\Facades\Hash;
8
9 class UserController extends Controller
10 {
11     public function index()
12     {
13         $data = [
14             'level_id' => 2,
15             'username' => 'manager_dua',
16             'nama' => 'Manager 2',
17             'password' => Hash::make('12345')
18         ];
19         UserModel::create($data);
20
21         $user = UserModel::all();
22         return view('user', ['data' => $user]);
23     }
24 }
25
```

3. Simpan kode program Langkah 1 dan 2, dan jalankan perintah web server. Kemudian jalankan link [localhostPWL\\_POS/public/user](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
5	customer-1	Pelanggan Pertama	5
6	manager_dua	Manager 2	2

Username `manager_dua` berhasil ditambahkan

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

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

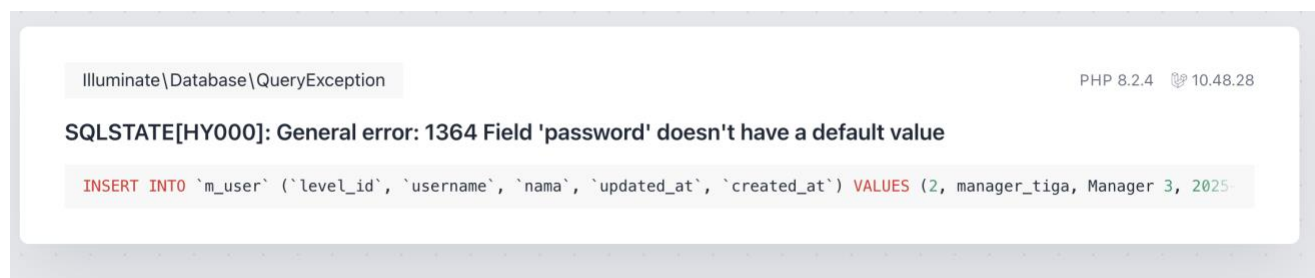
5. Ubah kembali file controller `UserController.php` seperti pada gambar di bawah hanya bagian array pada `$data`



```
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



Terjadi error karena variable 'password' tidak ada di fillable



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*. *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:

```
// 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

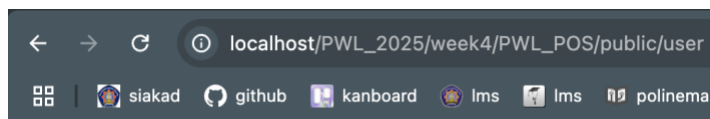
```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::find(1);
        return view('user', ['data' => $user]);
    }
}
```



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



## Data User

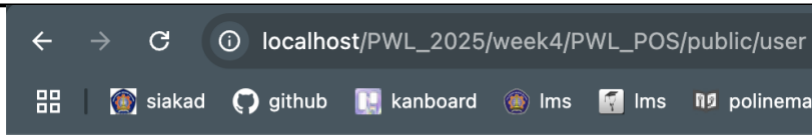
ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Yang terjadi adalah hanya muncul data yang memiliki id 1

4. 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', 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



## Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Sama seperti langkah sebelumnya karena data yang dimunculkan adalah data yang dimana level id nya adalah 1

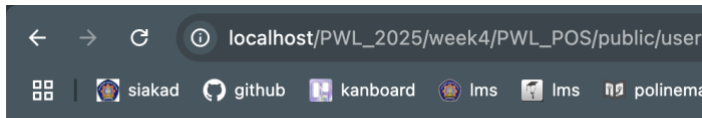
- Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::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



## Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Sama seperti langkah kedua, namun lebih ringkas karena first di tuliskan diawal

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:

```
$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

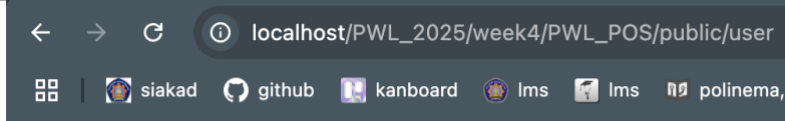
```
class UserController extends Controller  
{  
    public function index()  
    {  
        $user = UserModel::findOr(1, ['username', 'nama'], function () {  
            abort(404);  
        });  
  
        return view('user', ['data' => $user]);  
    }  
}
```

9. Simpan kode program Langkah 8. Kemudian pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan





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>



## Data User

ID	Username	Nama	ID Level Pengguna
	admin	Administrator	

Mencari data dengan id 1, lalu hanya menampilkan username dan nama nya saja yang sesuai dengan id yang dicari.

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



```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::findOrFail(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



404 | NOT FOUND

Terjadi error karena data yang diinginkan tidak ada, di kode meminta untuk memunculkan data dengan id 20, sedangkan di database tidak sampai 20 data, maka yang terjadi adalah error 404 dari 'abort(404)'

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 *route* 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



```
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

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Usaha untuk mengambil data dari tabel dengan id 1, jika ada maka data akan muncul, sebaliknya jika tidak ada maka akan muncul 404 error

3. 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('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

404 | NOT FOUND

404 not found karena data yang diminta tidak ada didalam tabel

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

### Praktikum 2.3 – Retrieving Aggregates

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');
```

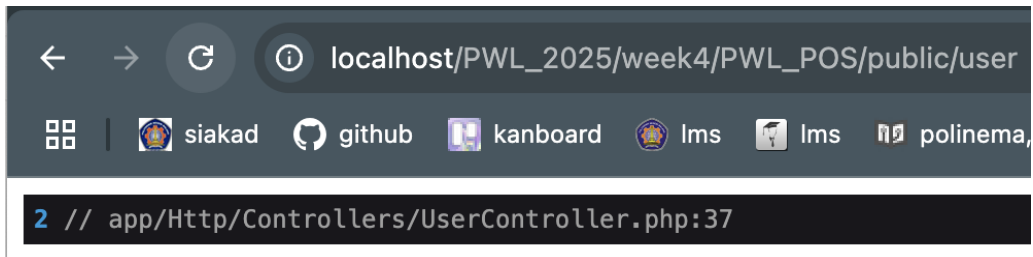
1. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
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

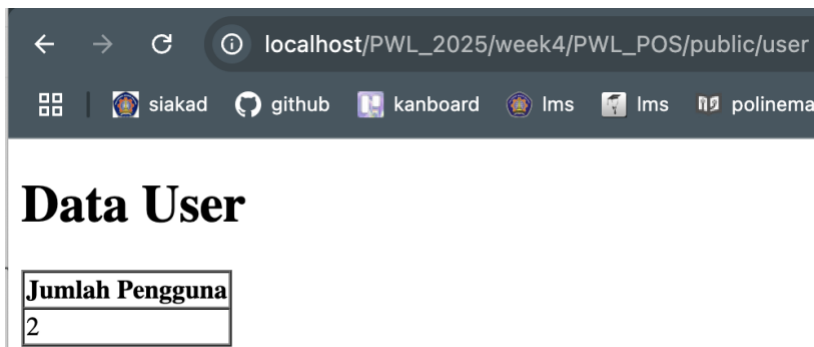


terjadi dan beri penjelasan dalam laporan



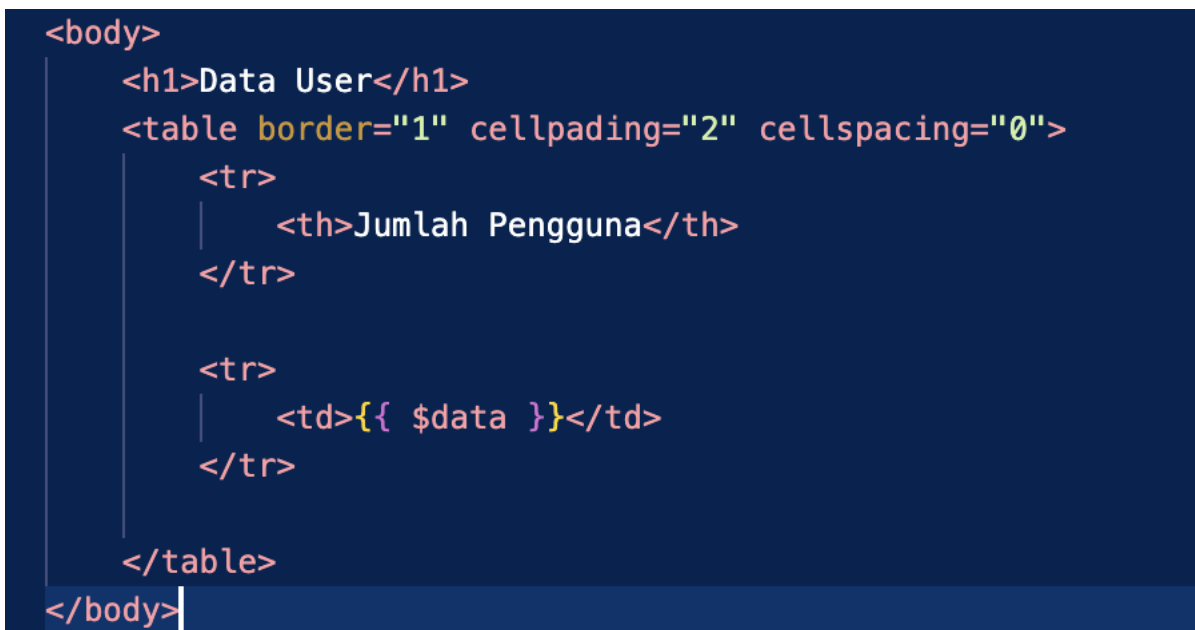
Muncul angka 2 karena terbaca ada 2 username dengan id yang sama yaitu 2

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





## Data User

Jumlah Pengguna
2

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

### Praktikum 2.4 – Retrieving 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 database tersebut sesuai dengan nilai yang dimasukkan.

Metode `firstOrCreate`, 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 `firstOrCreate` belum disimpan ke database. Anda perlu memanggil metode `save()` secara manual untuk menyimpannya:

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

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

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::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

```
<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

ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2





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>

---

Data yang dituliskan muncul karena ada dalam table database

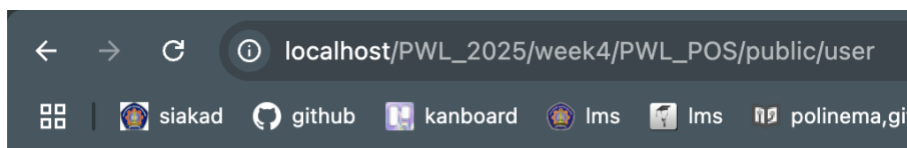
4. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini



```
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
7	manager22	Manager Dua Dua	2

	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$op3iK/W.emd55cRhgZemnes0kVxR4bXbbtgxLKmh5.3...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$SaRnXoE13Q2Qoc4jNbtYrtO4znspBFY1goGsjK.Hlyt/...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Original length 60	\$2y\$12\$A.x7FqQHv5ftERJfoJtJe7baA0V2aUVAX0OqJUDHJa...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	5	customer-1	Pelanggan Pertama	\$2y\$12\$bFIE0iY1.BDEA2iD1sK5EeJk5mUfIL.1apmpffhwFhG...	NULL	2025-03-08 05:52:33
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	2	manager_dua	Manager 2	\$2y\$12\$ITc0c1anK3ltoX/GbOI/4OQzeRe7Ncl1mYybWt7va....	2025-03-08 09:00:42	2025-03-08 09:00:42
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	2	manager22	Manager Dua Dua	\$2y\$12\$EDMit0QbBWhl/fZmdiZGOeINgVUVA.2ldnugAu5LoL...	2025-03-08 15:06:26	2025-03-08 15:06:26

Pada halaman website data muncul seperti biasa, lalu di dalam tabel *m\_user* menjadi ada data baru karena data yang dituliskan itu belum ada di dalam table sebelumnya.

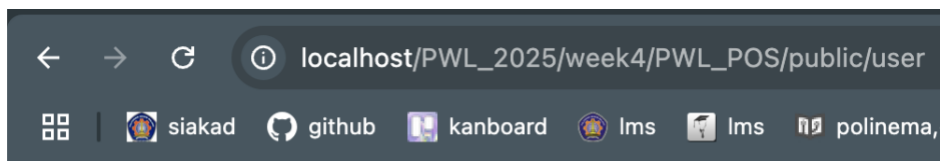
6. Ubah file controller dengan nama *UserController.php* dan ubah *script* seperti gambar di bawah ini



```
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

Data yang tampil adalah data pada tabel dengan id 2

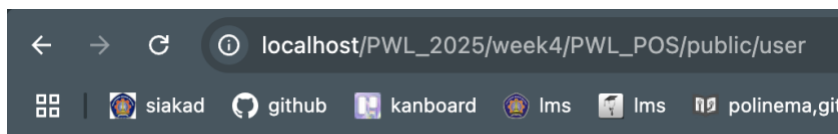
8. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini



```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrCreate(
            [
                '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
<input type="checkbox"/>	1	1	admin	Administrator	\$2y\$12\$op3ik/W.emd55cRhgzemnes0kVxR4bXbgtgLKmh5.3...	NULL	NULL
<input type="checkbox"/>	2	2	manager	Manager	\$2y\$12\$aRnXoE13Q2Qoc4JNbtYrtO4znspBFY1goGsjKHlyt/...	NULL	NULL
<input type="checkbox"/>	3	3	staff	Staff/Kasir	\$2y\$12\$A.x7FqOHVo5fiERJfoJJe7baA0V2aUVAX0OqJUDHJa...	NULL	NULL
<input type="checkbox"/>	5	5	customer-1	Pelanggan Pertama	\$2y\$12\$bFIE0iY1.BDEA2ID1sK5EeJk5mUILL.1apmpffhwFhG...	NULL	2025-03-08 05:52:33
<input type="checkbox"/>	6	2	manager_dua	Manager 2	\$2y\$12\$ITc0c1anK3ltoX/GbOI/4OQzeRe7Ncl1mYybWl7va....	2025-03-08 09:00:42	2025-03-08 09:00:42
<input type="checkbox"/>	7	2	manager22	Manager Dua Dua	\$2y\$12\$EDMi0QbBWhf/IfZmdiZGOOelNGvUVA.2fdnugAu5LoL...	2025-03-08 15:06:26	2025-03-08 15:06:26

Pada halaman website muncul data yang baru di tuliskan, namun di database tidak terbuat datanya karena menggunakan method 'firstOrCreate'

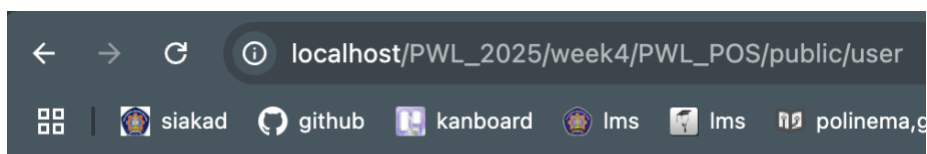
10. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini



```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrCreate(
            [
                '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
8	manager33	Manager Tiga Tiga	2

	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$op3iKW.emd55cRhgZemnes0kVxR4bXbtlxLkmh5.3...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$SaRnXoE13Q2Qoc4JNbtYrtO4znspBFY1goGsjK.Hlyt/...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$A.x7FqOHVo5tIERJfoJtJe7baA0V2aUVAX0QqJUDHJa...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	5	customer-1	Pelanggan Pertama	\$2y\$12\$bFIE0Yl.BDEA2ID1sK5EaJk5mUfLl.1apmpfthwFhG...	NULL	2025-03-08 05:52:33
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	2	manager_dua	Manager 2	\$2y\$12\$IToC1anK3ltoX/GbO/4OQzeRe7Nola1mYybWt7va....	2025-03-08 09:00:42	2025-03-08 09:00:42
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	2	manager22	Manager Dua Dua	\$2y\$12\$EDMt0QbWWhf/IZmdZGOeINgVUVA.2fdnugAu5LoL...	2025-03-08 15:06:26	2025-03-08 15:06:26
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	2	manager33	Manager Tiga Tiga	\$2y\$12\$Yr6YBblo6tFqZkgnV8P8.3oS7zXnE6HtsBqZj0D0l3...	2025-03-08 15:24:37	2025-03-08 15:24:37

Kode '\$user->save()' membuat data baru yang belum ada didalam table menjadi dimasukan kedalam table.

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:

```
$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



```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::create([
            'username' => 'manager55',
            'nama' => 'Manager55',
            'password' => Hash::make('12345'),
            'level_id' => 2,
        ]);

        $user->username = 'manager56';

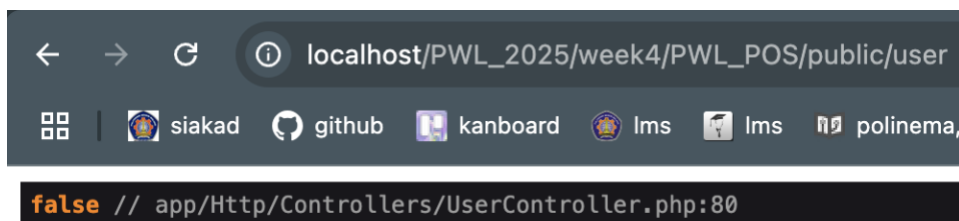
        $user->isDirty(); // true
        $user->isDirty('username'); // true
        $user->isDirty('nama'); // false
        $user->isDirty(['nama', 'username']); // true

        $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



Membuat pengguna baru, mengubah username lalu melakukan perubahan dengan `isDirty` dan `isClean`,

`$user->isDirty()` akan jadi `false`, karena semua perubahan sudah disimpan.

`$user->isClean()` akan jadi `true`, karena tidak ada lagi perubahan setelah disimpan.

Lalu `dd($user->isDirty())` akan menampilkan `false`, karena model sudah `clean` setelah di `save`

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:





```
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

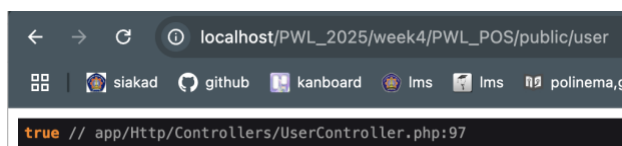
```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::create([
            'username' => 'manager11',
            'nama' => 'Manager11',
            'password' => Hash::make('12345'),
            'level_id' => 2,
        ]);

        $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



Metode `wasChanged()` digunakan untuk mengecek apakah suatu atribut memang benar-benar berubah saat `save()` terakhir dilakukan.

`dd` Akan menampilkan `true` jika salah satu dari nama atau username berubah saat `save()`. Karena nama tidak berubah, hasil akan tergantung apakah username berubah atau tidak

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 scripnya 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 scripnya 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



localhost/PWL\_2025/week4/PWL\_POS/public/user

siakad github kanboard lms lms polinema

### Data User

[+ Tambah User](#)

ID	Username	Nama	ID Level Pengguna	Aksi
1	admin	Administrator	1	<a href="#">Ubah</a>   <a href="#">Hapus</a>
2	manager	Manager	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
3	staff	Staff/Kasir	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
5	customer-1	Pelanggan Pertama	5	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
10	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>

Di halaman web menampilkan seluruh isi tabel, dan terdapat kolom baru yaitu aksi ubah dan hapus

- 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 diletakkan di bawah method `index` seperti gambar di bawah ini

```
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



← → ↻ ⓘ 127.0.0.1:8000/user/tambah

❏ | siakad github kanboard lms

## Form Tambah Data User

Username

Nama

Password

Level ID

Muncul form untuk mengisi data user

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

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

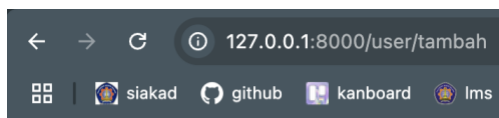


9. Tambahkan *script* pada controller dengan nama file *UserController.php*. Tambahkan *script* dalam class dan buat method baru dengan nama tambah\_simpan dan diletakkan di bawah method tambah seperti gambar di bawah ini

```
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](localhost/PWL_POS/public/user/tambah) pada *browser* dan input formnya dan simpan, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan



## Form Tambah Data User

Username

Nama

Password

Level ID

Terdapat form untuk menuliskan username, nama, password, dan level id, dan bisa di simpan

## Data User

[+ Tambah User](#)

ID	Username	Nama	ID Level Pengguna	Aksi
1	admin	Administrator	1	<a href="#">Ubah</a>   <a href="#">Hapus</a>
2	manager	Manager	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
3	staff	Staff/Kasir	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
5	customer-1	Pelanggan Pertama	5	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
10	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	test	ujah	5	<a href="#">Ubah</a>   <a href="#">Hapus</a>



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

## Form Ubah Data User

[Kembali](#)

Username	<input type="text" value="admin"/>
Nama	<input type="text" value="Administrator"/>
Password	<input type="password" value="....."/>
Level ID	<input type="text" value="1"/>
<input type="button" value="Ubah"/>	

Data yang sebelumnya bisa di ganti, namun belum bisa di simpan perubahan nya

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

## Form Ubah Data User

[Kembali](#)

Username   
Nama   
Password   
Level ID

## Data User

[+ Tambah User](#)

ID	Username	Nama	ID Level Pengguna	Aksi
1	admin1	Administrator1	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
2	manager	Manager	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
3	staff	Staff/Kasir	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
5	customer-1	Pelanggan Pertama	5	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
10	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	test	ujah	5	<a href="#">Ubah</a>   <a href="#">Hapus</a>

Data yang diubah sudah bisa disimpan didalam table

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



```
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

## Data User

[+ Tambah User](#)

ID	Username	Nama	ID Level Pengguna	Aksi
1	admin1	Administrator1	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
2	manager	Manager	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
3	staff	Staff/Kasir	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
5	customer-1	Pelanggan Pertama	5	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
10	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	test	ujah	5	<a href="#">Ubah</a>   <a href="#">Hapus</a>

Sebelum di hapus

## Data User

[+ Tambah User](#)

ID	Username	Nama	ID Level Pengguna	Aksi
1	admin1	Administrator1	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
2	manager	Manager	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
3	staff	Staff/Kasir	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
5	customer-1	Pelanggan Pertama	5	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	test	ujah	5	<a href="#">Ubah</a>   <a href="#">Hapus</a>

Setelah di hapus data dengan id 10

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

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

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

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

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 scripnya menjadi seperti 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'];

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

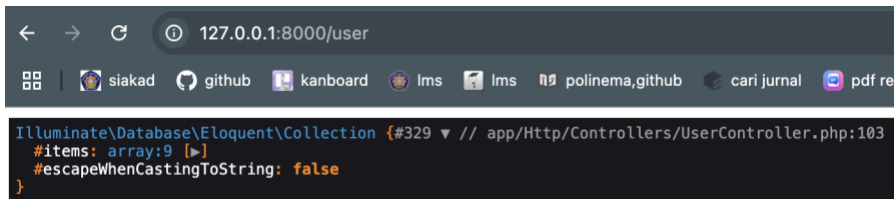
2. Buka file controller pada **UserController.php** dan ubah method *script* menjadi seperti di bawah ini





```
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



Output berhasil mengambil data dari table `m_user` dengan relasi ke `m_level`, tetapi hanya menampilkan struktur koleksi data tanpa detail isinya

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

```
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

```
<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





## Data User

[+ Tambah User](#)

ID	Username	Nama	ID Level Pengguna	Kode Level	Nama Level	Aksi
1	admin1	Administrator1	2	MNG	Manajer	<a href="#">Ubah</a>   <a href="#">Hapus</a>
2	manager	Manager	2	MNG	Manajer	<a href="#">Ubah</a>   <a href="#">Hapus</a>
3	staff	Staff/Kasir	3	STF	Staff/Kasir	<a href="#">Ubah</a>   <a href="#">Hapus</a>
5	customer-1	Pelanggan Pertama	5	CUS	Pelanggan,	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	manager_dua	Manager 2	2	MNG	Manajer	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager22	Manager Dua Dua	2	MNG	Manajer	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager33	Manager Tiga Tiga	2	MNG	Manajer	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager56	Manager55	2	MNG	Manajer	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	test	ujah	5	CUS	Pelanggan,	<a href="#">Ubah</a>   <a href="#">Hapus</a>

Jika semua sintaks sudah benar, akan menampilkan daftar user lengkap dengan informasi levelnya.

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

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



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>

---