



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

**Nama : Muhammad Bagus Indrawan**

**Kelas : TI-2H**

**NIM : 2241720217**

## **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

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

#### **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'];
}
```

```
app > Models > UserModel.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  3 references | 0 implementations
9  class UserModel extends Model
10 {
11     use HasFactory;
12
13     0 references
14     protected $table = 'm_user';
15     0 references
16     protected $primaryKey = 'user_id';
17     0 references
18     protected $fillable = ['level_id', 'username', 'nama', 'password'];
19 }
```

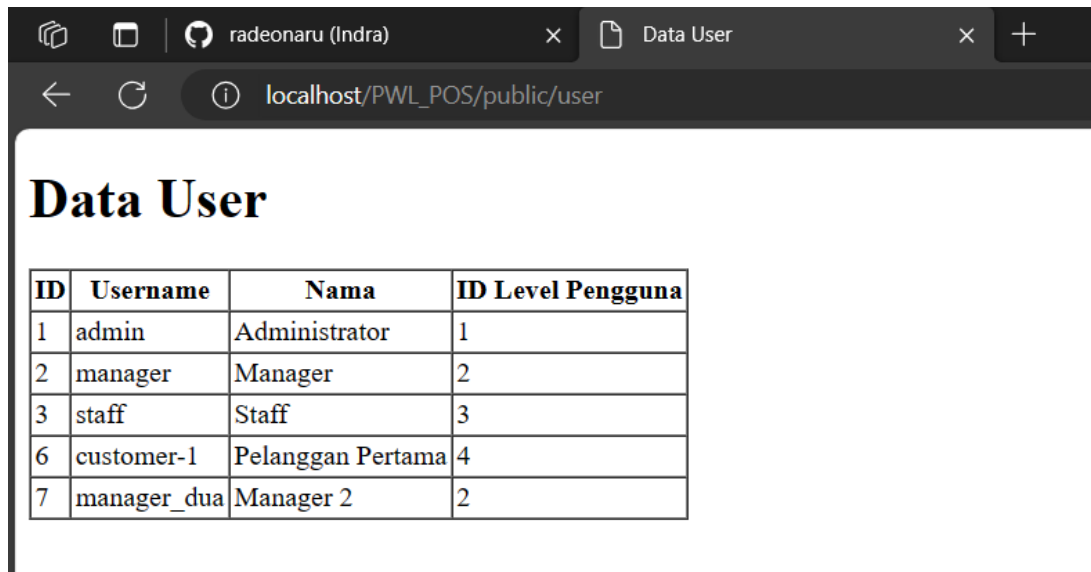
2. Buka file controller dengan nama `UserController.php` dan ubah *script* untuk menambahkan data baru seperti gambar di bawah ini



```
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
```

```
app > Http > Controllers > UserController.php
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 2 references | 0 implementations
10 class UserController extends Controller
11 {
12     1 reference | 0 overrides
13     public function index()
14     {
15         $data = [
16             'level_id' => 2,
17             'username' => 'manager_dua',
18             'nama' => 'Manager 2',
19             'password' => Hash::make('12345')
20         ];
21         UserModel::create($data);
22
23         $user = UserModel::all();
24         return view('user', ['data' => $user]);
25     }
26 }
```

3. Simpan kode program Langkah 1 dan 2, dan jalankan perintah web server. Kemudian jalankan link [localhostPWL\\_POS/public/user](localhostPWL_POS/public/user) pada *browser* dan amati apa yang terjadi



ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff	3
6	customer-1	Pelanggan Pertama	4
7	manager_dua	Manager 2	2

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

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

```
app > Models > UserModel.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  3 references | 0 implementations
9  class UserModel extends Model
10 {
11     use HasFactory;
12
13     0 references
14     protected $table = 'm_user';
15
16     0 references
17     protected $primaryKey = 'user_id';
18
19     0 references
20     protected $fillable = ['level_id', 'username', 'nama'];
21 }
```

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

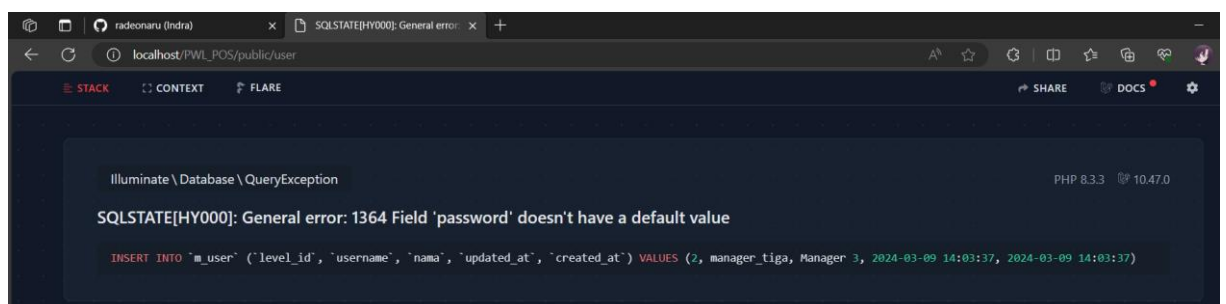


```
public function index()
{
    $data = [
        'level_id' => 2,
        'username' => 'manager_tiga',
        'nama' => 'Manager 3',
        'password' => Hash::make('12345')
    ];
    UserModel::create($data);

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

```
app > Http > Controllers > UserController.php
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  2 references | 0 implementations
10 class UserController extends Controller
11 {
12     1 reference | 0 overrides
13     public function index()
14     {
15         $data = [
16             'level_id' => 2,
17             'username' => 'manager_tiga',
18             'nama' => 'Manager 3',
19             'password' => Hash::make('12345')
20         ];
21         UserModel::create($data);
22
23         $user = UserModel::all();
24         return view('user', ['data' => $user]);
25     }
26 }
```

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



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

## 2. \$guarded



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

## B. RETRIEVING SINGLE MODELS

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

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

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

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

### Praktikum 2.1 – Retrieving Single Models

---



Buka file controller dengan nama dan ubah **UserController.php**

*script* seperti gambar

1. di bawah ini

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

```
app > Http > Controllers > UserController.php
2 references | 0 implementations
9 class UserController extends Controller
10 {
    1 reference | 0 overrides
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
20     // $data = [
21     //     'level_id' => 2,
22     //     'username' => 'manager_tiga',
23     //     'nama' => 'Manager 3',
24     //     'password' => Hash::make('12345')
25     // ];
26     // UserModel::create($data);
27
28     // $user = UserModel::all();
29     $user = UserModel::find(1);
30     return view('user', ['data' => $user]);
31 }
32
33
```

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

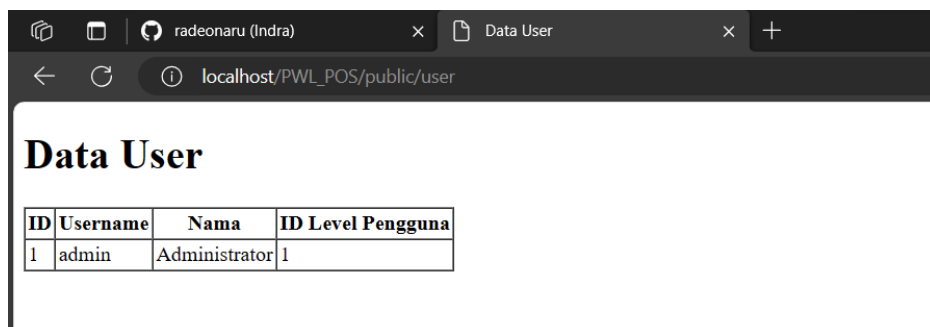
```
<body>
<h1>Data User</h1>
<table border="1" cellpadding="2" cellspacing="0">
    <tr>
        <td>ID</td>
        <td>Username</td>
        <td>Nama</td>
        <td>ID Level Pengguna</td>
    </tr>
    <tr>
        <td>{{ $data->user_id }}</td>
        <td>{{ $data->username }}</td>
        <td>{{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





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

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



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]);
    }
}
```





```
app > Http > Controllers > UserController.php
2 references | 0 implementations
class UserController extends Controller
10 {
    1 reference | 0 overrides
    public function index()
    {
        // $data = [
        //     'level_id' => 2,
        //     'username' => 'manager_dua',
        //     'nama' => 'Manager 2',
        //     'password' => Hash::make('12345')
        // ];

        // $data = [
        //     'level_id' => 2,
        //     'username' => 'manager_tiga',
        //     'nama' => 'Manager 3',
        //     'password' => Hash::make('12345')
        // ];
        // UserModel::create($data);

        // $user = UserModel::all();
        $user = UserModel::where('level_id', 1)->first();
        return view('user', ['data' => $user]);
    }
}
```

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

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

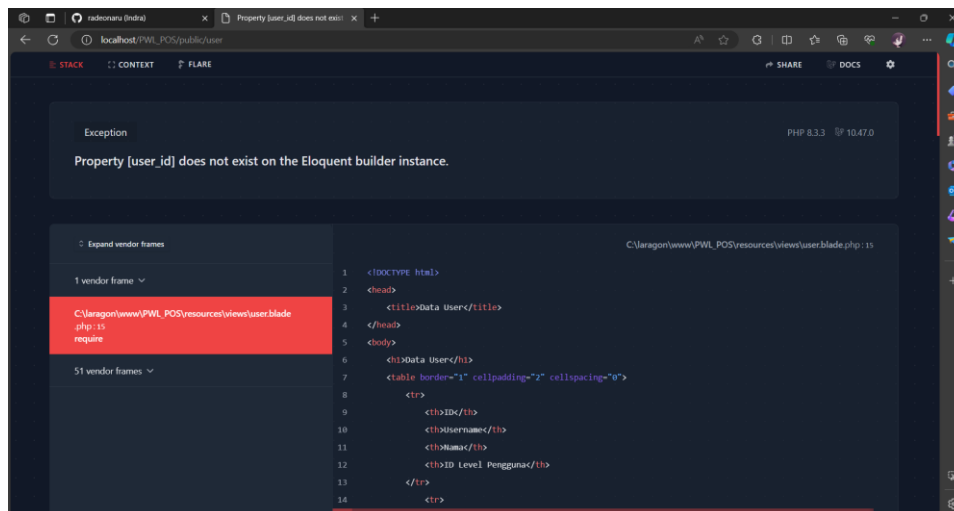
6. Ubah file controller dengan nama dan ubah di bawah ini

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



```
app > Http > Controllers > UserController.php
2 references | 0 implementations
9  class UserController extends Controller
10 {
    1 reference | 0 overrides
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
20      // $data = [
21      //     'level_id' => 2,
22      //     'username' => 'manager_tiga',
23      //     'nama' => 'Manager 3',
24      //     'password' => Hash::make('12345')
25      // ];
26      // UserModel::create($data);
27
28      // $user = UserModel::all();
29      $user = UserModel::where(['level_id', 1]);
30      return view('user', ['data' => $user]);
31  }
32 }
```

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



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



```
$user = UserModel::findOr(1, function () {  
    // ...  
});  
  
$user = UserModel::where('level_id', '>', 3)->firstOr(function () {  
    // ...  
});
```

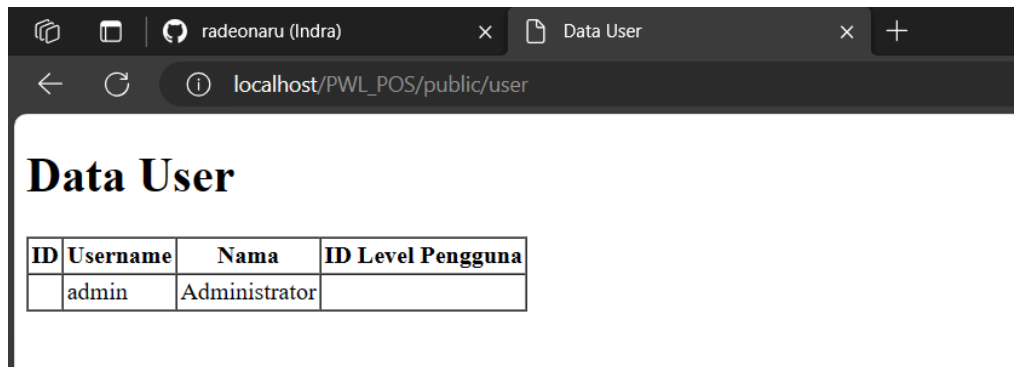
8. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

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

8

```
app > Http > Controllers > UserController.php  
1 reference | 0 overrides  
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  
20     // $data = [  
21     //     'level_id' => 2,  
22     //     'username' => 'manager_tiga',  
23     //     'nama' => 'Manager 3',  
24     //     'password' => Hash::make('12345')  
25     // ];  
26     // UserModel::create($data);  
27  
28     // $user = UserModel::all();  
29     // $user = UserModel::where('level_id', 1);  
30  
31     $user = UserModel::findOr(1, ['username', 'nama'], function() {  
32     |     abort(404);  
33     | });  
34     return view('user', ['data' => $user]);  
35 }  
36  
37
```

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



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

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

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

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

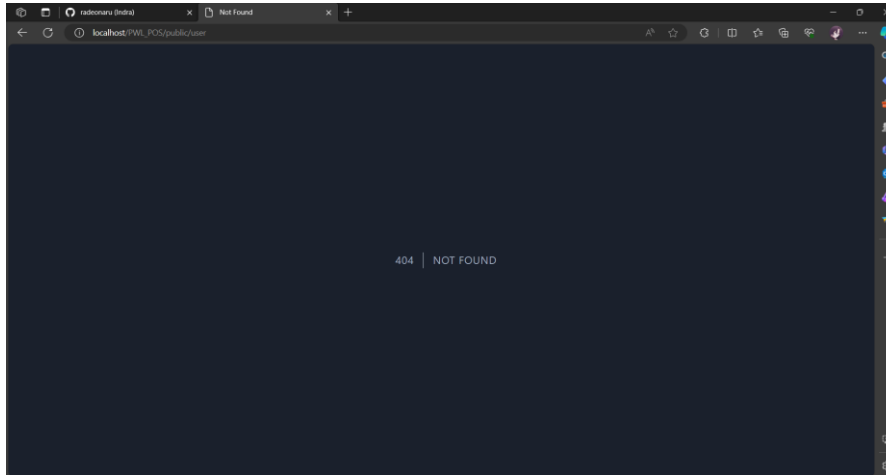
1 reference | 0 overrides
public function index()
{
    // $data = [
    //     'level_id' => 2,
    //     'username' => 'manager_dua',
    //     'nama' => 'Manager 2',
    //     'password' => Hash::make('12345')
    // ];

    // $data = [
    //     'level_id' => 2,
    //     'username' => 'manager_tiga',
    //     'nama' => 'Manager 3',
    //     'password' => Hash::make('12345')
    // ];
    // UserModel::create($data);

    // $user = UserModel::all();
    // $user = UserModel::where('level_id', 1);

    $user = UserModel::findOr(20, ['username', 'nama'], function() {
        abort(404);
    });
    return view('user', ['data' => $user]);
}
```

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



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

### Praktikum 2.2 – Not Found Exceptions

---

Terkadang Anda mungkin ingin memberikan pengecualian jika model tidak ditemukan. Hal ini sangat berguna dalam *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]);
    }
}
```



```
app > Http > Controllers > UserController.php > PHP Intelephense > UserController > index
9   class UserController extends Controller
11      public function index()
12      {
13      }
14      }
15
50      $user = UserModel::findOrFail(1);
51      return view('user', ['data' => $user]);
52
53  }
54
```

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

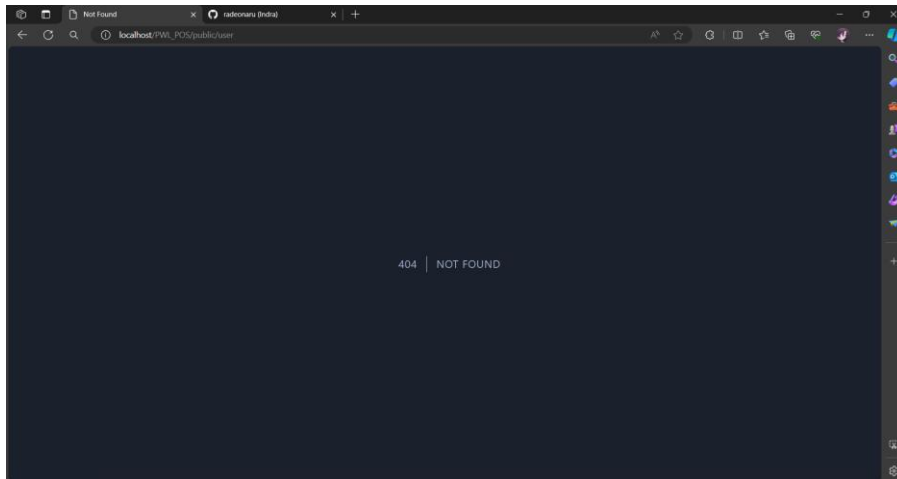
```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::where('username', 'manager9')->firstOrFail();
        return view('user', ['data' => $user]);
    }
}
```

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

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



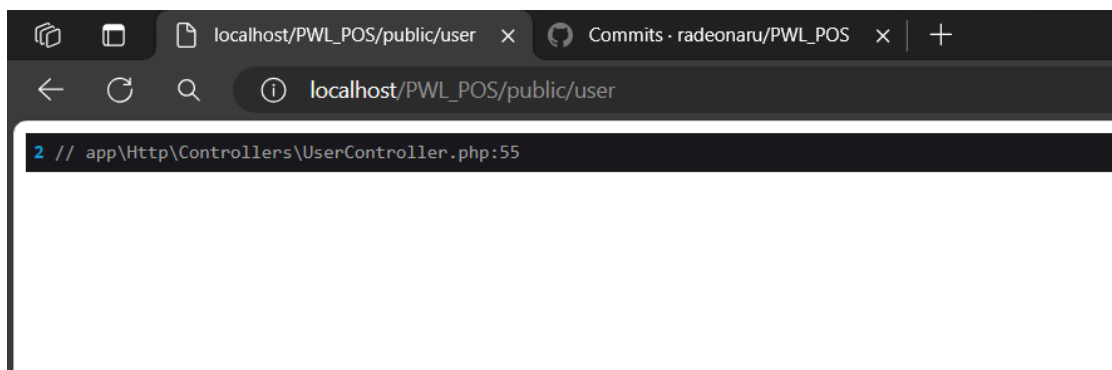
4. Simpan kode program Langkah 3. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



5. Laporkan hasil Praktikum-2.<sup>1</sup> ini dan *commit* perubahan pada *git*.

### Praktikum 2.<sup>2</sup> – Retrieving Aggregates

- 
- <sup>1</sup>. Simpan kode program Langkah 1. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



- <sup>2</sup>. 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



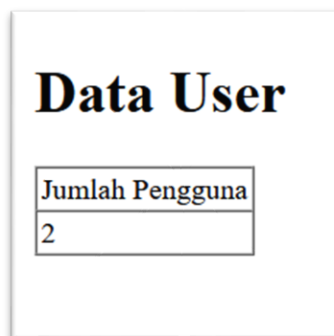


Saat berinteraksi dengan model Eloquent, Anda juga dapat menggunakan metode agregat `count`, `sum`, `max`, dan lainnya yang disediakan oleh pembuat kueri Laravel. Seperti yang Anda duga, metode ini mengembalikan nilai skalar dan contoh model Eloquent:

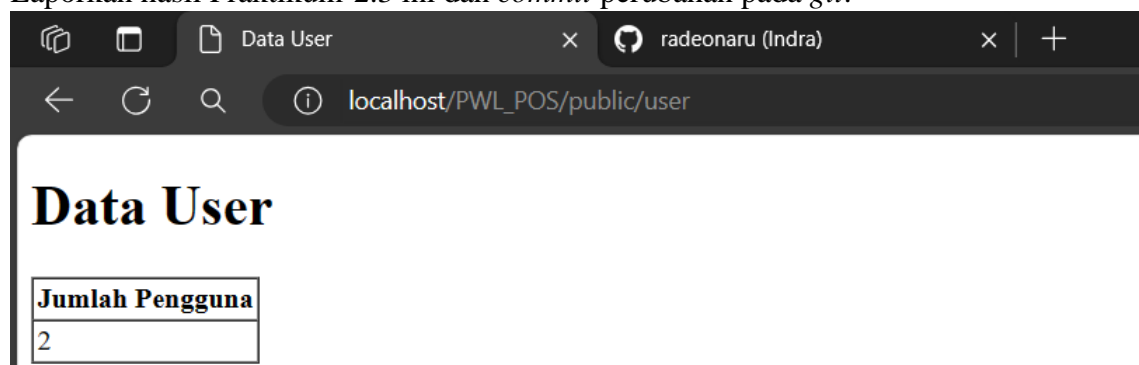
```
$count = UserModel::where('active', 1)->count();  
$max = UserModel::where('active', 1)->max('price');
```

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]);  
    }  
}
```



3. 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]);
    }
}
```



- Ubah kembali file *view* dengan nama `user.blade.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]);
    }
}
```

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

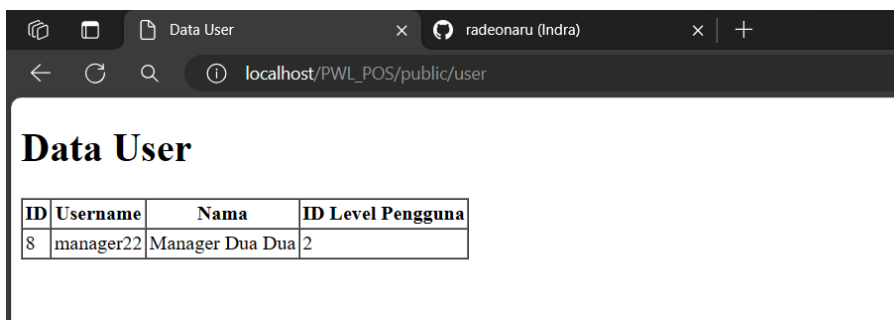


- Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
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



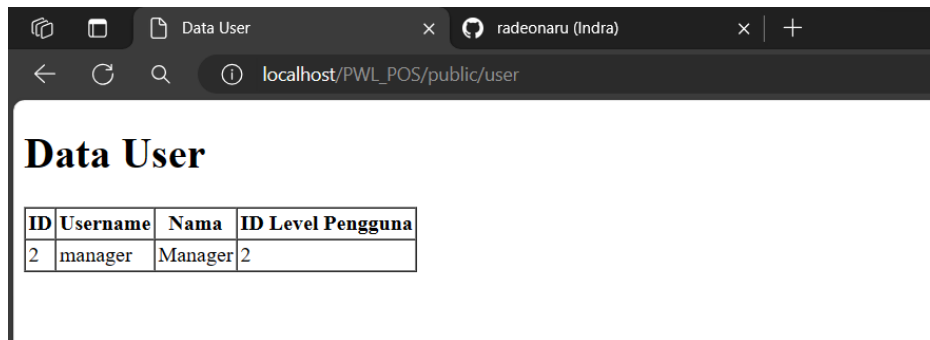
ID	Username	Nama	ID Level Pengguna
8	manager22	Manager Dua Dua	2

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::firstOrCreate(
            [
                '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



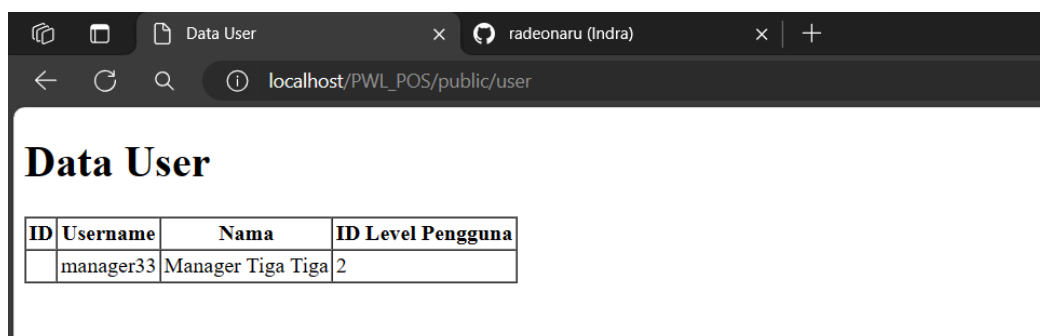
ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

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

```
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



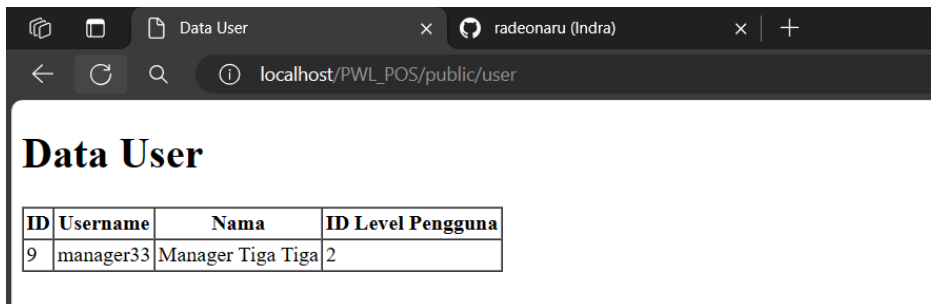
ID	Username	Nama	ID Level Pengguna
2	manager33	Manager Tiga Tiga	2

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



ID	Username	Nama	ID Level Pengguna
9	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:

```
$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';

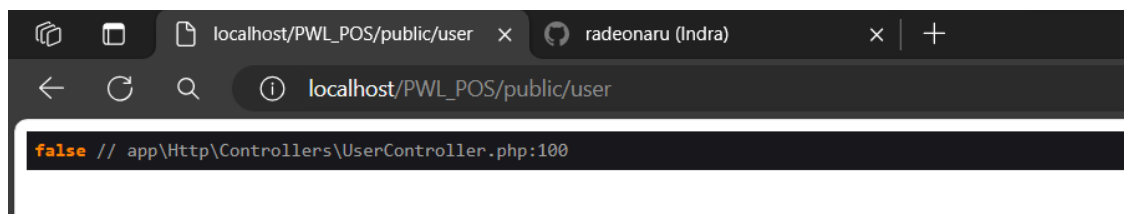
        $user->isDirty(); // true
        $user->isDirty('username'); // true
        $user->isDirty('nama'); // false
        $user->isDirty(['nama', 'username']); // true

        $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



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

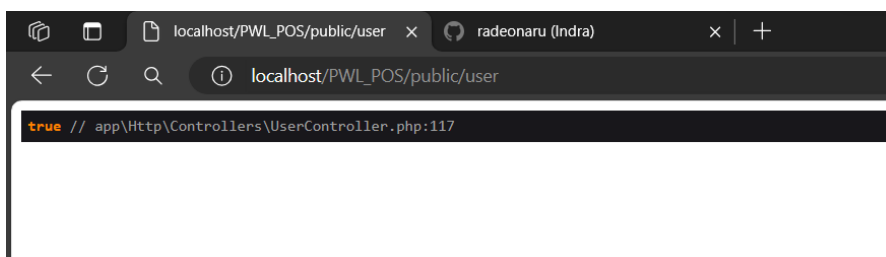
```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::create([
            'username' => 'manager11',
            'nama' => 'Manager11',
            'password' => Hash::make('12345'),
            'level_id' => 2,
        ]);

        $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





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



Browser tabs: Data User, radeonaru (Indra)

Address bar: localhost/PWL\_POS/public/user

### 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	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	customer-1	Pelanggan Pertama	4	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
10	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>

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



```
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

**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	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	customer-1	Pelanggan Pertama	4	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
10	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>

**Form Tambah Data User**

Username

Nama

Password

Label ID

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

Label ID

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 diletakkan 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

Ubah Data User

Kembali

Username Lithia

Nama Indrawan

Password .....

Label ID 3

Ubah

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 diletakkan 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

**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	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	customer-1	Pelanggan Pertama	4	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
10	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
12	Lithia	Indrawan	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
13	dsadas	dsadas	1	<a href="#">Ubah</a>   <a href="#">Hapus</a>
15	daffa	daffa	1	<a href="#">Ubah</a>   <a href="#">Hapus</a>

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	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	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	customer-1	Pelanggan Pertama	4	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
10	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
12	Lithia	Indrawan	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>



**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	3	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	customer-1	Pelanggan Pertama	4	<a href="#">Ubah</a>   <a href="#">Hapus</a>
7	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
8	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
9	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
10	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
11	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>

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

## Praktikum 2.7 – Relationships

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);
    }
}
```

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

```
Illuminate\Database\Eloquent\Collection {#319} // app\Http\Controllers\UserController.php:123
#items: array:9 [▼
  0 => App\Model\UserModel {#326}
  1 => App\Model\UserModel {#327}
  2 => App\Model\UserModel {#328}
  3 => App\Model\UserModel {#329}
  4 => App\Model\UserModel {#330}
  5 => App\Model\UserModel {#331}
  6 => App\Model\UserModel {#332}
  7 => App\Model\UserModel {#333}
  8 => App\Model\UserModel {#334}
]
#escapeWhenCastingToString: false
}
```

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

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