

**Laporan Praktikum Pemrograman Web Lanjut**

**Jobsheet 4 : MODEL dan ELOUENT ORM**



Oleh :

Danica Nasywa Putriniair (2341760122)

Kelas SIB 2B / 05

**PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS JURUSAN**

**TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur

65141

## A. Properti \$fillable dan \$guarded

### 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
```

Kode :

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

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

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

Hasil :

## Data Level Pengguna

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

Penjelasan :

Terjadi penambahan data dengan ID 5

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

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

Hasil :

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

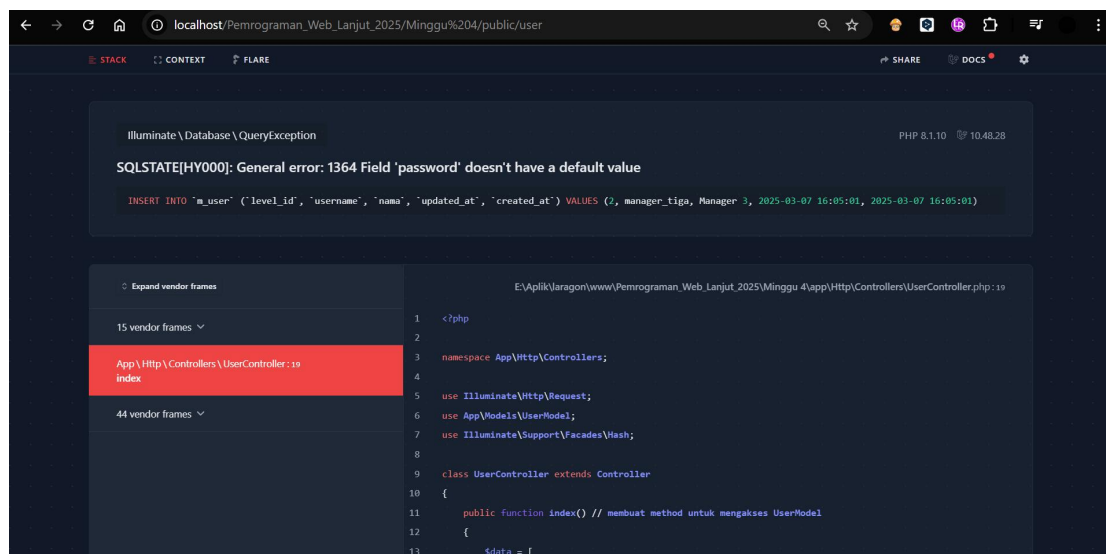
5. Ubah kembali file controller UserController.php seperti pada gambar di bawah hanya bagian array pada \$data

```
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

Hasil :



Penjelasan :

Muncul *error* disebabkan model di UserModel, atribut password belum ditambahkan ke dalam \$fillable.

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

## B. Retrieving Single Models

### 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]);
    }
}
```

Hasil :

```
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  class UserController extends Controller
10 {
11     public function index() // membuat method untuk mengakses UserModel
12     {
13         $user = UserModel::find(1); // mengambil semua data user
14         return view('user', ['data' => $user]); // mengirim data ke tampilan
15     }
16 }
17 |
```

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>
```

Hasil :

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

Hasil:

---

## Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Penjelasan :

Data yang ditampilkan adalah pengguna dengan ID Level Pengguna 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

Hasil:

---

## Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Penjelasan :

Tidak ada perubahan karena permintaan yang dilakukan masih sama.

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::firstWhere('level_id', 1);
        return view('user', ['data' => $user]);
    }
}

```

Hasil :

```

app > Http > Controllers > UserController.php > UserController > index
7   use Illuminate\Support\Facades\Hash;
8
9   class UserController extends Controller
10  {
11      public function index() // membuat method untuk mengakses UserModel
12      {
13          $user = UserModel::where(['level_id', 1]); // mengambil semua data user
14          return view('user', ['data' => $user]); // mengirim data ke tampilan
15      }
16  }
17

```

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

Hasil :

## Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Penjelasan :

UserModel::firstWhere('level\_id', 1); digunakan untuk mencari data dalam tabel dengan mengintrepesentasikan dari UserModel di mana kolom level\_id bernilai 1, dan langsung mengambil satu data pertama yang ditemukan sesuai dengan kondisi tersebut. Jik ada pengguna dengan level\_id = 1, variabel \$user akan berisi objek UserModel dari hasil pencarian, jika tidak ditemukan akan bernilai null.

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]);
    }
}

```

Hasil :

```

public function index() // membuat method untuk mengakses UserModel
{
    $user = UserModel::findOr(1, ['username', 'nama'], function(){ // mencari data user berdasarkan primary key (user_id = 1), hanya mengambil kolom username dan nama.
        abort(404); // jika tidak ditemukan, jalankan callback yang memanggil. abort 404 = menghentikan eksekusi dan mengembalikan HTTP response 4040 (not found)
    }); // mengambil semua data user
    return view('user', ['data' => $user]); // mengirim data ke tampilan
}

```

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

Hasil:

## Data User

ID	Username	Nama	ID Level Pengguna
	admin	Administrator	

Penjelasan :

Menampilkan data user yang berdasarkan primary key (hanya username dan nama saja yang ditampilkan)

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



```

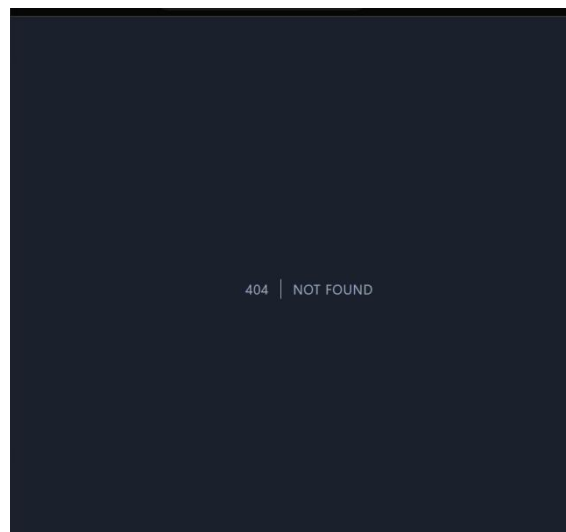
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::findOr(20, ['username', 'nama'], function () {
            abort(404);
        });

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

```

11. Simpan kode program Langkah 10. Kemudian jalankan pada browser dan amati apa yang terjadi dan beri penjelasan dalam laporan

Hasil:



Penjelasan:

Data yang dicari tidak ditemukan maka dikembalikan nilai not found pada data.

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

## Praktikum 2.2 - Not Found Exceptions

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]);
    }
}
```

Hasil :

```
$user = UserModel::findOrFail(1); // mengambil semua data user
return view('user', ['data' => $user]); // mengirim data ke tampilan
```

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

Hasil:

## Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Penjelasan:

Menampilkan data berdasarkan primary key dengan nilai 1.

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

Hasil :



404 | NOT FOUND

Penjelasan:

Karena data primary key yang dicari tidak ada di dalam database, sehingga muncul error 404 (not found).

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

### Praktikum 2.3 - *Rereiving Aggregrates*

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

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

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

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

Hasil:

```
public function index() // membuat method untuk mengakses UserModel
{
    $user = UserModel::where('level_id', 2)->count(); // mengambil semua data user
    dd($user);
    return view('user', ['data' => $user]); // mengirim data ke tampilan
}
```

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

Hasil:

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

Penjelasan:

- Count() hanya digunakan untuk menghitung jumlah user dengan level\_id = 2, bukan mengambil datanya
  - dd(\$user) digunakan untuk debuggging dan melihat hasil query
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

Kode:

- UserController

```

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         $jumlahPengguna = UserModel::where('level_id', 2)->count();
14         return view('user', ['jumlahPengguna' => $jumlahPengguna]);
15     }
16 }

```

- user.blade.php

```

1 <!DOCTYPE html>
2 <html>
3
4 <head>
5     <title>Data Level Pengguna</title>
6 </head>
7
8 <body>
9     <h1>Data User</h1>
10    <table border="1" cellpadding="2" cellspacing="0">
11        <tr>
12            <th>Jumlah Pengguna</th>
13        </tr>
14        <tr>
15            <td>{{ $jumlahPengguna }}</td>
16        </tr>
17    </table>
18 </body>
19 </html>

```

Hasil:

## Data User

Jumlah Pengguna
2

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

## Praktikum 2.4 - Retrieving or Creating Models

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

Hasil:

## Data Level Pengguna

ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

Penjelasan:

firstOrCreate() akan mencari data pertama yang memiliki username = 'manager' dan



nama 'Manager' pada database. Jika data ditemukan, maka data yang sudah ada akan dikembalikan. Jika data tidak ditemukan, maka data baru akan dibuat dengan nilai yang diberikan. Dan ini digunakan untuk mencegah redundansi data dan memastikan hanya satu entri yang ada di database berdasarkan kondisi yang telah diberikan.

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

Hasil :

- UserController

```
class UserController extends Controller
{
    public function index() // membuat method untuk mengakses UserModel
    {
        $user = UserModel::firstOrCreate(
            [
                'username' => 'manager22', // laravel mencari data dengan username ma
                'nama' => 'Manager Dua Dua',
                'level_id' => 2, // Ditambahkan agar pencarian lebih spesifik
            ],
            [
                'password' => Hash::make('12345'),
            ]
        );

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

- UserModel

```
class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan
    protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel

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

- Pada website

## Data User

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

- Pada database

	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\$W5hDu1C7ARJb8/9o2FgFYeh21ulBfHQHPcttRuyWIX...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$WJQo2IAfk.NAZwJXNTRAKeObSf1HIYbRvsFoXA.7KTU...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Zo1fWNRHPoUjIA9IdaKi.X.Aorgt/AY73vzvpU8fUli...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	37	2	manager_dua	Manager 2	\$2y\$12\$Qq7s7P6wCVYem0moxNxoOy6pxHmR4uNUARSivVj80...	2025-03-10 02:31:59	2025-03-10 02:31:59
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	38	2	manager22	Manager Dua Dua	\$2y\$12\$W37qyR4owOa6krt1IoAqeW90rF91tGzKgooVLlBkP...	2025-03-10 04:44:26	2025-03-10 04:44:26

Penjelasan:

Cara kerja fisrtOrCreate() adalah

- Laravel mencari data di dalam database berdasarkan array pertama
- Jika ditemukan, maka data tersebut akan dikembalikan tanpa perubahan
- Jika data tidak ditemukam, Laravel akan membuat data baru menggunakan kombinasi dari array pertama dan array kedua

Mengapa harus ada 2 array dalam kode tersebut?

Karena memiliki peran yang berbeda, yaitu:

- Array pertama sebagai kriteria pencarian (where)
- Array kedua digunakan sebagai data yang akan ditambahkan apabila data yang di array pertama tidak ada di database.

Saya tidak menggunakan arahan bapak karena yang muncul error, dan pada database tidak ada perubahan data. Maka dari itu, saya mencari cara bagaimana agar muncul pada database.

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

Hasil:

## Data User

ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

Penjelasan:

firstOrCreate() adalah metode yang mirip dengan firstOrCreate(), tetapi tidak menyimpan data ke dalam database secara otomatis.

- Jika data ditemukan maka Laravel akan mengembalikan model yang sudah ada.
- Jika data tidak ditemukan maka Laravel membuat insrance model baru, tetapi tidak menyimpannya ke dalam database.

Cara Kerja

- Laravel akan mencari data berdasarkan array pertama.
- Jika data ditemukan, maka model tersebut dikembalikan.
- Jika data tidak ditemukan, maka Laravel membuat objek baru, tetapi tidak menyimpannya ke dalam database.
- Jika ingin menyimpan data yang baru dibuat, kita harus memanggil save() secara manual.

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

Hasil

- Kode

```

$user = UserModel::firstOrCreate( //mencari data pengguna
    [
        'username' => 'manager33',
        'nama' => 'Manager Tiga Tiga',
        'password' => Hash::make('12345'),
        'level_id' => 2
    ],
); // jika ditemukan, mengembalikan data tersebut
// jika tidak ditemukan , membuat record baru de
return view('user', ['data' => $user]);

```

- Web

## Data User

ID	Username	Nama	ID Level Pengguna
	manager33	Manager Tiga Tiga	2

- phpMyAdmin

		user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	admin	Administrator	\$2y\$12\$W5hDu1C7ARJb8/9o2FgFYeh21ulbIHQHPcttRuyWIX...	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	2	manager	Manager	\$2y\$12\$WJQo2IAIk.NAzWJXNTRakeObsf1HIYbRvsFoXA.7kTU...	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Zo1WNRHPofU.tA9ldaKl.X.Aorgt/Ay73vzvpU8fUi...	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	37	2	manager_dua	Manager 2	\$2y\$12\$Qq7s7Pi6wCVYem0mxxNxoOy6pxHmR4uNJUARSivVj80...	2025-03-10 02:31:59	2025-03-10 02:31:59
<input type="checkbox"/>	Edit Copy Delete	38	2	manager22	Manager Dua Dua	\$2y\$12\$Xw37qyR4owOa6krt1loAqe/w90rF91tGzKgooVLIBKp...	2025-03-10 04:44:26	2025-03-10 04:44:26

Penjelasan:

Tidak bisa tersimpan pada database karena kurangnya fungsi pada kode kurang lengkap.

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]);
    }
}
```

Hasil :

```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::firstOrCreate(
            [
                'username' => 'manager33',
                'nama' => 'Manager Tiga Tiga',
                'level_id' => 2, // Kriteria pencarian
            ],
            [
                'password' => Hash::make('12345') // Data tambahan jika tidak ditemukan
            ]
        );

        $user->save(); // Simpan data baru ke database

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

phpMyAdmin

			user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	1 admin	Administrator	\$2y\$12\$W5hDu1C7ARJb8/9o2FgFYeh21ulbflHQHPcttRuyWDX...	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	2	2 manager	Manager	\$2y\$12\$WJQo2IAIk NAzwJXNTRAKEObSf1HiYbRvsFoXA.7kTU...	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	3	3 staff	Staff/Kasir	\$2y\$12\$Zo1fWNRHPotUjIA9IdaKI.X.Aorgt/AY73vzvpU8tUi...	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	37	2 manager_dua	Manager 2	\$2y\$12\$Qq7s7P6wCVYem0moxNxoOy6pxHmR4uNJUARSivVj80...	2025-03-10 02:31:59	2025-03-10 02:31:59
<input type="checkbox"/>	Edit	Copy	Delete	38	2 manager22	Manager Dua Dua	\$2y\$12\$W37qyR4owOa6krt1IoAqe/w90rF91IGzKgoovLJBKp...	2025-03-10 04:44:26	2025-03-10 04:44:26
<input type="checkbox"/>	Edit	Copy	Delete	39	2 manager33	Manager Tiga Tiga	\$2y\$12\$dwX4EeQIPy2PbRgXKvGvlunV6ZMLPTQy.ZjjJ7InWdr...	2025-03-10 04:50:57	2025-03-10 04:50:57

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

Fungsi save() digunakan untuk menyimpan data yang mau diinputkan, dan data akan tersimpan secara otomatis.

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



## Praktikum 2.5 - Attribute Changes

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());
    }
}
```

Hasil :

```
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(); // Simpan data baru ke database

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

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

Hasil :

```
false // app\Http\Controllers\UserController.php:57
```

Penjelasan:

- `create()` = digunakan untuk membuat record baru dalam database.
- Mengubah username manager55 menjadi manager56
- `isDirty()` digunakan untuk mengecek apakah suatu atribut mengalami perubahan dibandingkan dengan data yang terdapat pada database

Kode	Penjelasan
<code>\$user-&gt;isDirty();</code>	true, karena ada perubahan pada atribut username
<code>\$user-&gt;isDirty('username');</code>	true, karena username berubah.
<code>\$user-&gt;isDirty('nama');</code>	false, karena nama masih sama seperti yang ada di database.
<code>\$user-&gt;isDirty(['nama', 'username']);</code>	true, karena setidaknya satu dari atribut (username) berubah.

- `isClean()` digunakan untuk mengecek apakah atribut belum mengalami perubahan.

Kode	Penjelasan
<code>\$user-&gt;isClean();</code>	false, karena ada perubahan pada username.
<code>\$user-&gt;isClean('username');</code>	false, karena username sudah diubah
<code>\$user-&gt;isClean('nama');</code>	true, karena nama belum berubah.
<code>\$user-&gt;isClean(['nama', 'username']);</code>	false, karena setidaknya satu atribut berubah (username).

Setelah `save()`, model tidak lagi “dirty”, kecuali ada perubahan yang dilakukan setelah pemanggilan.

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

Hasil :

```

true // app\Http\Controllers\UserController.php:49

```

Penjelasan :

Metode \$user->wasChanged() digunakan untuk mengecek apakah ada perubahan

pada model setelah disimpan ke database dengan \$user->save()

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

## Praktikum 2.6 - Create, read, Update, dan Delete (CRUD)

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 scriptnya untuk read menjadi seperti di bawah ini

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

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

Hasil:

## 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	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
6	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
13	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
21	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
22	manager11	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
28	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
31	manager55	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>

Penjelasan:

Terdapat aksi yang menunjukkan 2 tautan Ubah (edit data) dan Hapus (menghapus data yang tidak diperlukan) dengan parameter `user_id`.

- 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>
```

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

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

- 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]);
    }

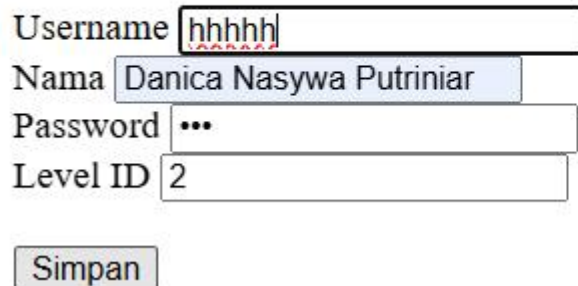
    public function tambah()
    {
        return view('user_tambah');
    }
}
```

- 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



Hasil :

## Form Tambah Data User



Username

Nama

Password

Level ID

Penjelasan:

Pada kode yang di ketik terdapat perubahan pada href agar dapat terhubung pada web yang diinginkan.

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

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

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

```
public function tambah_simpan(Request $request)
{
    UserModel::create([
        'username' => $request->username,
        'nama' => $request->nama,
        'password' => Hash::make('$request->password'),
        'level_id' => $request->level_id
    ]);

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

10. Simpan kode program Langkah 8 dan 9. Kemudian jalankan link localhost:8000/user/tambah atau localhost/PWL\_POS/public/user/tambah pada browser dan input formnya dan simpan, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan

Penjelasan:

Setelah menginputkan data maka halaman setelah menyimpan data adalah halaman awal

# Data User

## + Tambah User

ID	Username	Nama	IDPengguna	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>
37	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
38	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
39	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
40	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
41	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
44	hhhhh	Danica Nasywa Putrinier	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>

Dan terdapat tambahan data dengan id 44

- 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>
```

- 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

Penjelasan :

Ubah digunakan untuk mengubah data yang akan diganti.

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

Penjelasan :

Setelah klik tombol ubah data yang ingin diubah telah berubah

+ Tambah User

ID	Username	Nama	IDPengguna	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>
37	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
38	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
39	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
40	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
41	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
44	hhhh	da	2	<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

Penjelasan:

Data dapat terhapus

+ Tambah User

ID	Username	Nama	IDPengguna	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>
37	manager_dua	Manager 2	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
38	manager22	Manager Dua Dua	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
39	manager33	Manager Tiga Tiga	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
40	manager56	Manager55	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>
41	manager12	Manager11	2	<a href="#">Ubah</a>   <a href="#">Hapus</a>

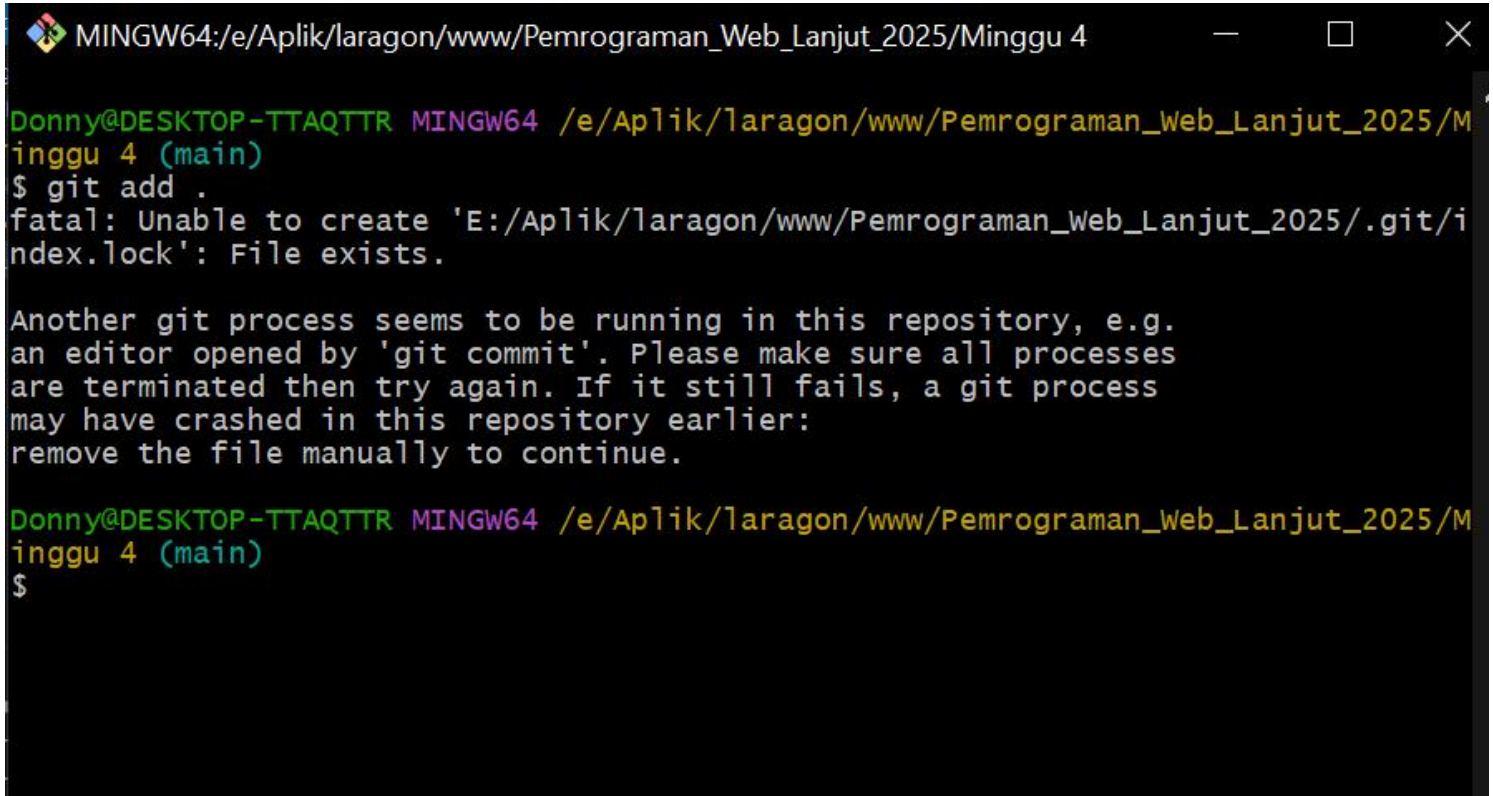


21. Laporkan hasil Praktikum-2.6 ini dan commit perubahan pada git.

```
fatal: Unable to create 'E:/Aplik/laragon/www/Pemrograman_Web_Lanjut_2025/.git/index.lock': File exists.

Another git process seems to be running in this repository, e.g.
an editor opened by 'git commit'. Please make sure all processes
are terminated then try again. If it still fails, a git process
may have crashed in this repository earlier:
remove the file manually to continue.
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 4> git add .
fatal: Unable to create 'E:/Aplik/laragon/www/Pemrograman_Web_Lanjut_2025/.git/index.lock': File exists.

Another git process seems to be running in this repository, e.g.
an editor opened by 'git commit'. Please make sure all processes
are terminated then try again. If it still fails, a git process
may have crashed in this repository earlier:
remove the file manually to continue.
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 4> █
```

A screenshot of a Windows terminal window titled "MINGW64:/e/Aplik/laragon/www/Pemrograman\_Web\_Lanjut\_2025/Minggu 4". The terminal shows a user named "Donny@DESKTOP-TTAQTTR" in the "MINGW64" environment. The user is in the directory "/e/Aplik/laragon/www/Pemrograman\_Web\_Lanjut\_2025/Minggu 4 (main)". They run the command "\$ git add .". The terminal outputs a "fatal: Unable to create 'E:/Aplik/laragon/www/Pemrograman\_Web\_Lanjut\_2025/.git/index.lock': File exists." error. Below this, a message states: "Another git process seems to be running in this repository, e.g. an editor opened by 'git commit'. Please make sure all processes are terminated then try again. If it still fails, a git process may have crashed in this repository earlier: remove the file manually to continue." The prompt "\$" is shown again at the bottom, indicating the command execution has finished.

```
MINGW64:/e/Aplik/laragon/www/Pemrograman_Web_Lanjut_2025/Minggu 4
Donny@DESKTOP-TTAQTTR MINGW64 /e/Aplik/laragon/www/Pemrograman_Web_Lanjut_2025/M
inggu 4 (main)
$ git add .
fatal: Unable to create 'E:/Aplik/laragon/www/Pemrograman_Web_Lanjut_2025/.git/i
ndex.lock': File exists.

Another git process seems to be running in this repository, e.g.
an editor opened by 'git commit'. Please make sure all processes
are terminated then try again. If it still fails, a git process
may have crashed in this repository earlier:
remove the file manually to continue.

Donny@DESKTOP-TTAQTTR MINGW64 /e/Aplik/laragon/www/Pemrograman_Web_Lanjut_2025/M
inggu 4 (main)
$
```

Gabisa push di git hub

## Praktikum 2.7 - Relationship

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

Hasil:

```
Illuminate\Database\Eloquent\Collection {#319 ▼ // app\Http\Controllers\UserController.php:16
  #items: array:8 [▶]
  #escapeWhenCastingToString: false
}
```

Penjelasan:

- Model LevelMode = untuk merepresentasikan tabel m\_level.
  - Model UserModel = untuk merepresentasikan tabel m\_user dan memiliki relasi many-to-one dengan LevelMode.
  - Method index() = mengambil semua data dari tabel m\_user dan sekaligus memuat relasi dengan tabel m\_level.
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

Hasil:

## Data User

[+ Tambah User](#)

ID	Username	Nama	IDPengguna	Kode Level	Nama Level	
1	admin	Administrator	1	ADM		<a href="#">Ubah</a>   <a href="#">Hapus</a>
2	manager	Manager	2	MNG		<a href="#">Ubah</a>   <a href="#">Hapus</a>
3	staff	Staff/Kasir	3	STF		<a href="#">Ubah</a>   <a href="#">Hapus</a>
37	manager_dua	Manager 2	2	MNG		<a href="#">Ubah</a>   <a href="#">Hapus</a>
38	manager22	Manager Dua Dua	2	MNG		<a href="#">Ubah</a>   <a href="#">Hapus</a>
39	manager33	Manager Tiga Tiga	2	MNG		<a href="#">Ubah</a>   <a href="#">Hapus</a>
40	manager56	Manager55	2	MNG		<a href="#">Ubah</a>   <a href="#">Hapus</a>
41	manager12	Manager11	2	MNG		<a href="#">Ubah</a>   <a href="#">Hapus</a>

Penjelasan:

Menampilkan kode level dan nama level pada tabel

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

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 4> git add .  
fatal: Unable to create 'E:/Aplik/laragon/www/Pemrograman_Web_Lanjut_2025/.git/index.lock': File exists.
```

Another git process seems to be running in this repository, e.g.  
an editor opened by 'git commit'. Please make sure all processes  
are terminated then try again. If it still fails, a git process  
may have crashed in this repository earlier:  
remove the file manually to continue.

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 4> git commit -m "Praktikum 2.6 dan 2.7"  
fatal: Unable to create 'E:/Aplik/laragon/www/Pemrograman Web Lanjut 2025/.git/index.lock': File exists.
```

Another git process seems to be running in this repository, e.g.  
an editor opened by 'git commit'. Please make sure all processes  
are terminated then try again. If it still fails, a git process  
may have crashed in this repository earlier:  
remove the file manually to continue.

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 4> █
```