LAPORAN PRAKTIKUM PEMROGRAMAN WEB LANJUT (PWL)
JOBSHEET 4: MODEL dan ELOQUENT ORM

OLEH:

LOVELYTA SEKARAYU KRISDIYANTI

KELAS 2B SIB / 11 (2341760081)

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*

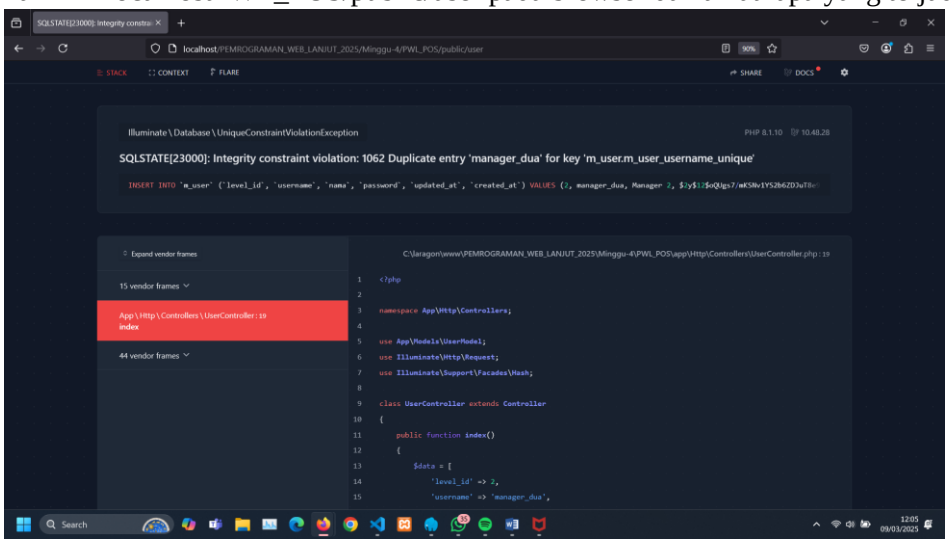
1. Buka file model dengan nama UserModel.php dan tambahkan $fillable seperti gambar di bawah ini

```php
class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';

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

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

```php
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



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

```php
class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';

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

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

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

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

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

Data User

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

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

❖ ***PRAKTIKUM 2.1***

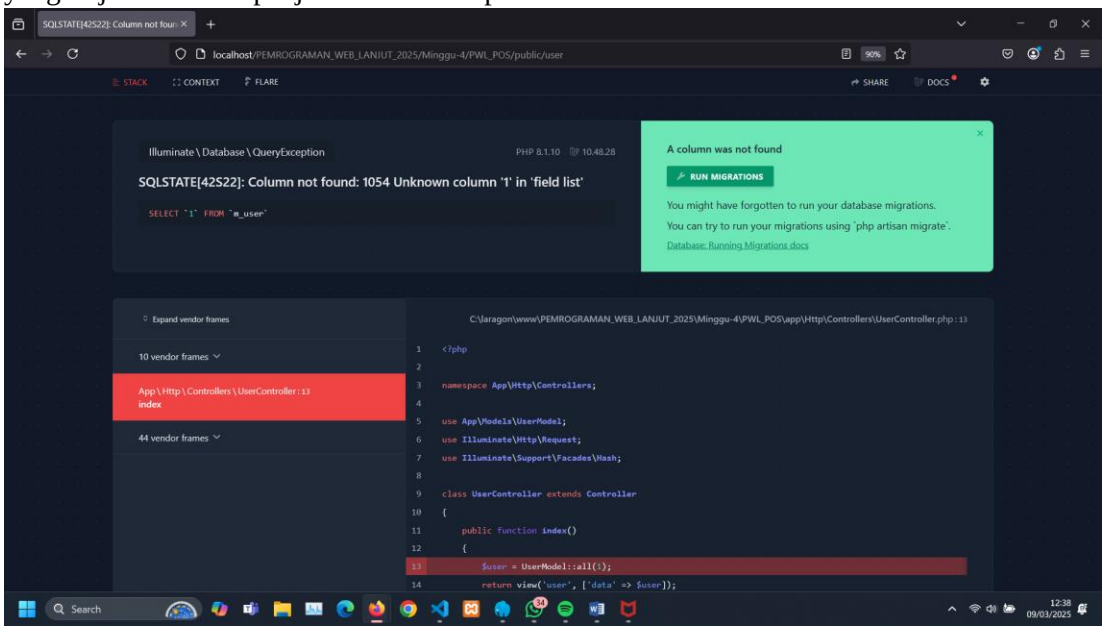
1. Buka file controller dengan nama UserController.php dan ubah script seperti gambar di bawah ini

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

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

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

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



- metode all() tidak menerima parameter sedangkan terdapat parameter pada $user = UserModel::all(1);
- all() mengembalikan semua data dalam tabel, jadi parameter 1 yang diberikan menyebabkan error.

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

```
 9  class UserController extends Controller
10  {
11      public function index()
12      {
13          $user = UserModel::where('level_id', 1)->first();
14          return view('user', ['data' => $user]);
15      }
16  }
```

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

- $user = UserModel::where('level_id', 1)->first(); Mengambil satu data pertama yang memiliki level_id = 1

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

```
 9  class UserController extends Controller
10  {
11      public function index()
12      {
13          $user = UserModel::firstWhere('level_id', 1);
14          return view('user', ['data' => $user]);
15      }
16  }
```

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

- firstWhere('level_id', 1) Secara otomatis melakukan filter where('level_id', 1)->first()

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

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

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

Data User

ID	Username	Nama	ID Level Pengguna
	admin	Administrator	

- findOr(1, ['username', 'nama'], function () { abort(404); }) Mencari data berdasarkan id = 1 dan hanya mengambil kolom yang disebutkan dalam array kedua (['username', 'nama']). Sehingga data ID dan ID Level Pengguna tidak ditampilkan.

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

```php
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

- 404 Not Found, karena jika data tidak ditemukan, fungsi callback (abort(404)) akan dieksekusi.

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

❖ ***PRAKTIKUM 2.2***

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

```
9   class UserController extends Controller
10  {
11      public function index()
12      {
13          $user = UserModel::findOrFail(1);
14          return view('user', ['data' => $user]);
15      }
16  }
```

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

Data User

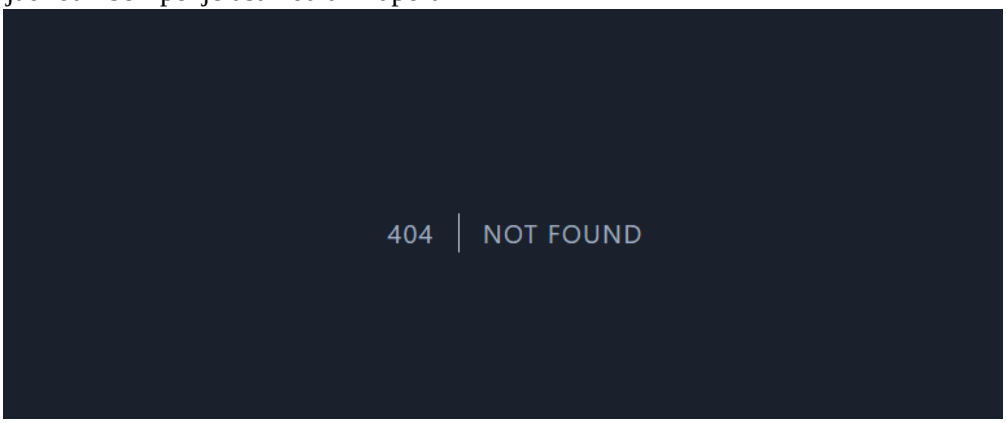
ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

- Pada findOrFail(1); jika data yang dicari ada seperti 1, maka akan ditampilkan data 1

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

```
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, karena tidak ada data manager9 yang tersimpan

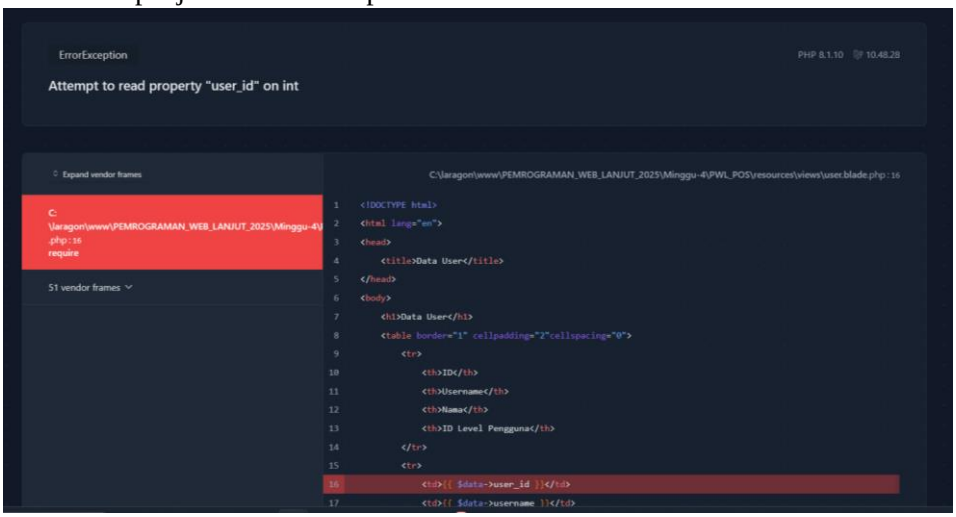
5. Laporkan hasil Praktikum-2.2 ini dan commit perubahan pada git.

❖ ***PRAKTIKUM 2.3***

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

```php
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::where('level_id', 2)->count();
        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



- Error terjadi karena $user = UserModel::where('level_id', 2)->count(); Mengembalikan integer (jumlah baris yang ditemukan), bukan objek model UserModel.

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

```php
<body>
    <h1>Data User</h1>
    <table border="1" cellpadding="2"cellspacing="0">
        <tr>
            <th>Jumlah Pengguna</th>
        </tr>
        <tr>
            <td>{{ $data }}</td>
        <tr/>
    </table>
</body>
```

Data User

Jumlah Pengguna
3

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

❖ ***PRAKTIKUM 2.4***

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

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

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

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

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

Data User

ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

- firstOrCreate() digunakan untuk menghindari duplikasi data. Jika data sudah ada, data lama digunakan. Jika tidak ada, data baru akan dibuat. Jika user manager baru pertama kali dibuat, ID-nya bisa menjadi 2 atau lebih tergantung urutan di database

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

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

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

Data User

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

user_id	level_id	username	nama	password	created_at	updated_at
1	1	admin	Administrator	$2y$12$zo1PqclpR4pJv3co9xiiV.0SdLu3sgd1fA.kmoW8pM4...	NULL	NULL
2	2	manager	Manager	$2y$12$Brt2LNnB86QOi1r4ZTTh.OKUMhTV4MsesMS85HXipOT...	NULL	NULL
3	3	staff	Staff/Kasir	$2y$12$0Vrw4gODXZ47m/910eO1PO8DH8.fsd.XSjbu4MtCqRd...	NULL	NULL
4	2	manager_dua	Manager 2	$2y$12$Cc4jprPHdmBEuCsxp.Xiie6PlY9no35zCwf/kUACQEj...	2025-03-09 04:57:47	2025-03-09 04:57:47
7	2	manager_tiga	Manager 3	$2y$12$MEVNbqB/xg08Grt7Tbt3V.fWDCSzbQvqI6FpASQvIXP...	2025-03-09 05:25:38	2025-03-09 05:25:38
8	2	manager22	Manager Dua Dua	$2y$12$ERpdOUfEsEBJXUfNWFKFKu9a8IyJq6XTjrutH.YNj1C...	2025-03-09 07:00:20	2025-03-09 07:00:20

- Data baru telah ditambahkan karena belum terdapat data dengan username manager22

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

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

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

Data User

ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

- firstOrNew() digunakan untuk mengembalikan data yang sudah ada sehingga, tidak akan mengalami duplikasi data. Jika data tidak ada, maka akan membuat objek baru tetapi tidak disimpan ke database
- Output di atas menunjukkan bahwa sudah terdapat data tersebut sehingga tidak perlu membuat data baru

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

```
9   class UserController extends Controller
0   {
1       public function index()
2       {
3           $user = UserModel::firstOrNew(
4               [
5                   'username' => 'manager33',
6                   'nama' => 'Manager Tiga Tiga',
7                   'password' => Hash::make('12345'),
8                   'level_id' => 2,
9               ],
0           );
1           return view('user', ['data' => $user]);
2       }
3   }
```

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
1	1	admin	Administrator	$2y$12$zo1PqclpR4pJv3co9xiiV.0SdLu3sgd1fA.kmoW8pM4...	NULL	NULL
2	2	manager	Manager	$2y$12$Brt2LNnB86QOi1r4ZTTh.OKUMhTV4MsesMS85HXipOT...	NULL	NULL
3	3	staff	Staff/Kasir	$2y$12$0Vrw4gODXZ47m/910eO1PO8DH8.fsd.XSjbu4MtCqRd...	NULL	NULL
4	2	manager_dua	Manager 2	$2y$12$Cc4jprPHdmBEuCsxp.Xiie6PIY9no35zCwf/kUACQEj...	2025-03-09 04:57:47	2025-03-09 04:57:47
7	2	manager_tiga	Manager 3	$2y$12$MEVNbqB/xg08Grt7Tbt3V.fWDCSzbQvqI6FpASQvIXP...	2025-03-09 05:25:38	2025-03-09 05:25:38
8	2	manager22	Manager Dua Dua	$2y$12$ERpdOUfEsEBJXUfNWFKFKu9a8IyJq6XTjrutH.YNj1C...	2025-03-09 07:00:20	2025-03-09 07:00:20

- Hasilnya adalah tidak terdapat ID, karena data tidak disimpan dalam database

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

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

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

Data User

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

user_id	level_id	username	nama	password	created_at	updated_at
1	1	admin	Administrator	$2y$12$zo1PqclpR4pJv3co9xiiV.0SdLu3sgd1fA.kmoW8pM4...	NULL	NULL
2	2	manager	Manager	$2y$12$Brt2LNnB86QOi1r4ZTTh.OKUMhTV4MsesMS85HXipOT...	NULL	NULL
3	3	staff	Staff/Kasir	$2y$12$0Vrw4gODXZ47m/910eO1PO8DH8.fsd.XSjbu4MtCqRd...	NULL	NULL
4	2	manager_dua	Manager 2	$2y$12$Cc4jprPHdmBEuCsxp.Xiie6PIY9no35zCwf/kUACQEj...	2025-03-09 04:57:47	2025-03-09 04:57:47
7	2	manager_tiga	Manager 3	$2y$12$MEVNbqB/xg08Grt7Tbt3V.fWDCSzbQvqI6FpASQvIXP...	2025-03-09 05:25:38	2025-03-09 05:25:38
8	2	manager22	Manager Dua Dua	$2y$12$ERpdOUfEsEBJXUfNWFKFKu9a8lyJq6XTjrutH.YNj1C...	2025-03-09 07:00:20	2025-03-09 07:00:20
9	2	manager33	Manager Tiga Tiga	$2y$12$NN7fpl.6U/FOyTZYAnphKOsNqFAQ8mxoFtdB08K.uZ/...	2025-03-09 07:15:00	2025-03-09 07:15:00

- Hasilnya terdapat ID dan data telah tersimpan pada database, karena $user->save(); berfungsi untuk menyimpan data user pada database

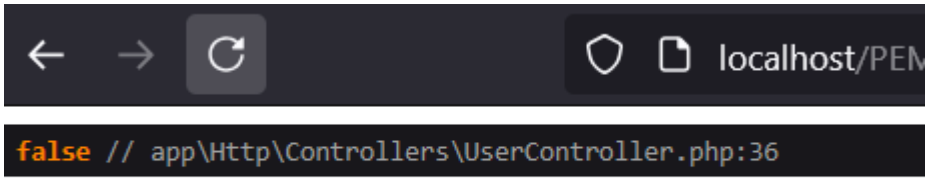
12. Laporkan hasil Praktikum-2.4 ini dan commit perubahan pada git.

❖ ***PRAKTIKUM 2.5***

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

```php
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::create(
            [
                'username' => 'manager55',
                'nama' => 'Manager55',
                'password' => Hash::make('12345'),
                'level_id' => 2,
            ]);
            $user->username = 'manager56';

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

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

            $user->save();

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

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



- isDirty() digunakan untuk mengecek apakah ada perubahan data yang belum disimpan ke database.
- isClean() digunakan untuk mengecek apakah tidak ada perubahan pada model.
- Setelah memanggil save(), model dianggap bersih (isClean() = true) karena sudah sinkron dengan database.
- Hasil akhir adalah false karena tidak ada perubahan yang belum disimpan (isDirty() = false).

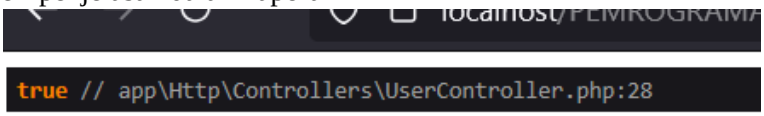
Metode ini wasChanged menentukan apakah ada atribut yang diubah saat model terakhir disimpan dalam siklus permintaan saat ini. Jika diperlukan, Anda dapat memberikan nama atribut untuk melihat apakah atribut tertentu telah diubah:

```php
public function index()
{
    $user = UserModel::create(
        [
            'username' => 'manager11',
            'nama' => 'Manager11',
            'password' => Hash::make('12345'),
            'level_id' => 2,
        ]);
        $user->username = 'manager12';

        $user->save();

        $user->wasChanged(); // true
        $user->wasChanged('username'); // true
        $user->wasChanged( ['username', 'level_id']) ; // true
        $user->wasChanged('nama'); // false
        $user->wasChanged ( ['nama', 'username' ]) ; // true
}
```

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

```php
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::create(
            [
                'username' => 'manager11',
                'nama' => 'Manager11',
                'password' => Hash::make('12345'),
                'level_id' => 2,
            ]);
            $user->username = 'manager12';

            $user->save();

            $user->wasChanged(); // true
            $user->wasChanged('username'); // true
            $user->wasChanged( ['username', 'level_id']); // true
            $user->wasChanged('nama'); // false
            dd($user->wasChanged(['nama', 'username'])); // true
    }
}
```

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



- $user->wasChanged(); mengembalikan true karena setidaknya satu atribut (username) telah diubah sebelum pemanggilan save()

- wasChanged('username') mengembalikan true karena username telah diubah dari 'manager11' menjadi 'manager12'
- wasChanged(['username', 'level_id']) mengembalikan true karena salah satu atribut dalam array tersebut (username) telah diubah
- wasChanged('nama') mengembalikan false karena atribut nama tidak diubah setelah model dibuat.
- wasChanged(['nama', 'username']) Mengembalikan true karena meskipun nama tidak berubah, username berubah. Jika salah satu dari atribut dalam array berubah, hasilnya akan tetap true

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

❖ ***PRAKTIKUM 2.6***

1. Buka file view pada user.blade.php dan buat scritpnya menjadi seperti di bawah ini

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

Data User

+ Tambah User

ID	Username	Nama	ID Level Pengguna	Aksi
1	admin	Administrator	1	Ubah | Hapus
2	manager	Manager	2	Ubah | Hapus
3	staff	Staff/Kasir	3	Ubah | Hapus
4	manager_dua	Manager 2	2	Ubah | Hapus
7	manager_tiga	Manager 3	2	Ubah | Hapus
8	manager22	Manager Dua Dua	2	Ubah | Hapus
9	manager33	Manager Tiga Tiga	2	Ubah | Hapus
10	manager56	Manager55	2	Ubah | Hapus
11	manager12	Manager11	2	Ubah | Hapus

- UserController.php mengambil semua data pengguna dengan UserModel::all() lalu mengirim data ke view user.blade.php
- user.blade.php menampilkan data dalam tabel menggunakan @foreach dan menyediakan link Ubah dan Hapus untuk setiap pengguna
- Output yang dihasilkan seperti gambar di atas. Tabel daftar pengguna muncul dengan data dari database dan setiap baris menampilkan ID, Username, Nama, Level, dan opsi aksi.

4. Langkah berikutnya membuat create atau tambah data user dengan cara bikin file baru pada view dengan nama user_tambah.blade.php dan buat scriptnya menjadi seperti di bawah ini

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Tambah User</title>
</head>
<body>
    <h1>Form Tambah Data User</h1>
    <form method="post" action="/user/tambah_simpan">
        {{ csrf_field() }}

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

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

```
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

- Not Found, karena belum terdapat Route post untuk menambah data

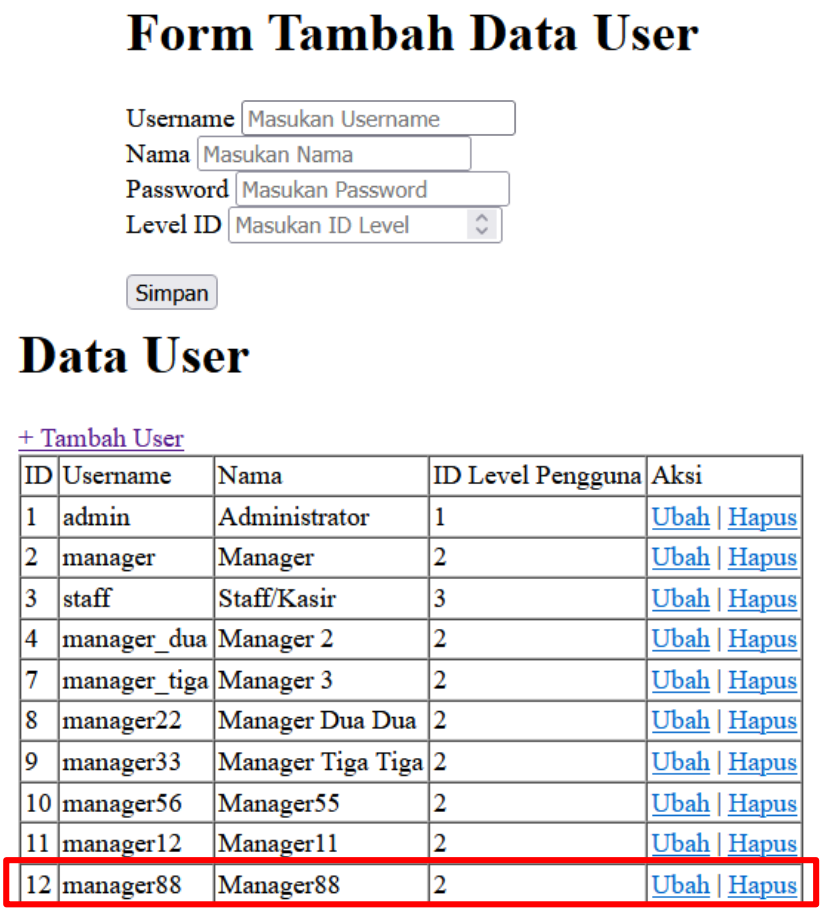
8. Tambahkan script pada routes dengan nama file web.php. Tambahkan seperti gambar di bawah ini

```
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)
{
    $request->validate([
        '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 linklocalhost: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



- Outputnya adalah Form Tambah Data User
- Route::post('/user/tambah_simpan', [UserController::class, 'tambah_simpan']); akan menghubungkan request POST ke method tambah_simpan di UserController
- Method tambah_simpan(Request $request) menerima input dari form lalu data divalidasi dan disimpan ke database. Setelah penyimpanan, pengguna diarahkan kembali ke halaman daftar user (redirect('/user')

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>
```

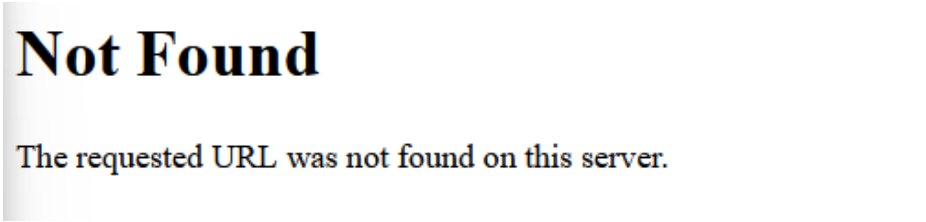
12. Tambahkan script pada routes dengan nama file web.php. Tambahkan seperti gambar di bawah ini

```
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



- Not Found, karena Route nya menggunakan get bukan put

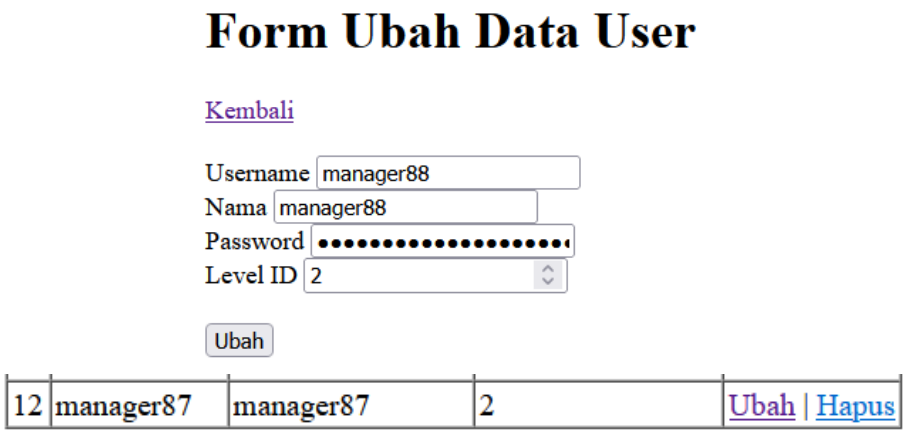
15. Tambahkan script pada routes dengan nama file web.php. Tambahkan seperti gambar di bawah ini

```
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

```php
public function ubah_simpan($id, Request $request)
{
    $user = UserModel::find($id);

    $user->username = $request->username;
    $user->nama = $request->nama;
    $user->password = Hash :: make('$request->password');
    $user->level_id = $request->level_id;

    $user->save();

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

17. Simpan kode program Langkah 15 dan 16. Kemudian jalankan link localhost:8000/user/ubah/1 atau localhost/PWL_POS/public/user/ubah/1 pada browser dan ubah input formnya dan klik tombol ubah, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan



- Route::put('/user/ubah_simpan/{id}', [UserController::class, 'ubah_simpan']); menghubungkan request PUT ke method ubah_simpan di UserController
- ubah_simpan($id, Request $request) mencari user berdasarkan $id lalu Data user diperbarui berdasarkan input form. Setelah perubahan disimpan, pengguna diarahkan kembali ke daftar user

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

```php
Route::get('/user/hapus/{id}', [UserController::class, 'hapus']);
```

19. Tambahkan script pada controller dengan nama file UserController.php. Tambahkan script dalam class dan buat method baru dengan nama hapus dan diletakan di bawah method ubah_simpan seperti gambar di bawah ini

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

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

20. Simpan kode program Langkah 18 dan 19. Kemudian jalankan pada browser dan klik tombol hapus, kemudian amati apa yang terjadi dan beri penjelasan dalam laporan

Data User

+ Tambah User

ID	Username	Nama	ID Level Pengguna	Aksi
1	admin	Administrator	1	Ubah | Hapus
2	manager	Manager	2	Ubah | Hapus
3	staff	Staff/Kasir	3	Ubah | Hapus
4	manager_dua	Manager 2	2	Ubah | Hapus
7	manager_tiga	Manager 3	2	Ubah | Hapus
8	manager22	Manager Dua Dua	2	Ubah | Hapus
9	manager33	Manager Tiga Tiga	2	Ubah | Hapus
10	manager56	Manager55	2	Ubah | Hapus
11	manager12	Manager11	2	Ubah | Hapus

- Route::get('/user/hapus/{id}', [UserController::class, 'hapus']); menghubungkan request GET ke method hapus di UserController
- hapus($id) mencari user berdasarkan $id, dan delete() digunakan untuk menghapus data user dari database. Setelah dihapus, data yang dihapus sudah tidak terdapat di dalam tabel data user lagi

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

❖ ***PRAKTIKUM 2.7***

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

```
class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';

    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

```
Illuminate\Database\Eloquent\Collection {#319 ▼ // app\Http\Controllers\UserController.php:14
  #items: array:9 [▶]
  #escapeWhenCastingToString: false
}
```

   - Terdapat relasi belongsTo dengan LevelModel berdasarkan level_id artinya, setiap user memiliki satu level tertentu yang diambil dari tabel level.
   - index() mengambil semua data dari UserModel dengan relasi level. dd($user); digunakan untuk debugging dan menampilkan hasil query di browser
   - Output di atas menampilkan array dengan 9 elemen, tetapi tidak ada error

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

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

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

```
<body>
    <h1>Data User</h1>
    <a href="http://localhost/PEMROGRAMAN_WEB_LANJU
    <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="http://localhost/PEMROGRAM
        <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	admin	Administrator	1	ADM	Administrator	Ubah | Hapus
2	manager	Manager	2	MNG	Manager	Ubah | Hapus
3	staff	Staff/Kasir	3	STF	Staff/Kasir	Ubah | Hapus
4	manager_dua	Manager 2	2	MNG	Manager	Ubah | Hapus
7	manager_tiga	Manager 3	2	MNG	Manager	Ubah | Hapus
8	manager22	Manager Dua Dua	2	MNG	Manager	Ubah | Hapus
9	manager33	Manager Tiga Tiga	2	MNG	Manager	Ubah | Hapus
10	manager56	Manager55	2	MNG	Manager	Ubah | Hapus
11	manager12	Manager11	2	MNG	Manager	Ubah | Hapus

- Ditambahkan kolom Kode Level dan Nama Level yang diambil dari relasi dengan LevelModel.

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