

PEMROGRAMAN WEB LANJUT

“Model dan Eloquent ORM”



Kelas : TI-2H

Disusun Oleh :

Fanesabhirawaning Sulistyio

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur
65141

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

Jawab:

Eloquent ORM dalam Laravel adalah teknik yang memungkinkan Anda berinteraksi dengan database menggunakan objek-objek PHP. Ini berarti Anda dapat mengakses, menambahkan, mengubah, dan menghapus data dalam database menggunakan kode PHP tanpa perlu menulis kueri SQL secara langsung.

File Model di Laravel adalah kelas PHP yang merepresentasikan tabel dalam database. Model ini berhubungan langsung dengan tabel tersebut dan memungkinkan Anda untuk melakukan operasi CRUD (Create, Read, Update, Delete) dengan mudah.

Dengan Eloquent ORM, Anda dapat mengelola data dalam database dengan cara yang lebih terstruktur dan mudah dipahami dalam aplikasi Laravel Anda.

A. PROPERTI `$fillable` DAN `$guarded`

Praktikum 1 - `$fillable`:

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



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

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

```
// Data yang akan dimasukkan ke dalam tabel m_user
$data = [
    'level_id' => 2,
    'username' => 'manager_dua',
    'nama' => 'Manager 2',
    'password' => Hash::make('12345') // Mengenkripsi password sebelum disimpan
];

// Menyimpan data ke dalam tabel m_user menggunakan model UserModel
UserModel::create($data);

// Mengambil semua data pengguna dari tabel m_user
$users = UserModel::all();
```

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

Data User

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

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

```
class UserModel extends Model
{
    use HasFactory;

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

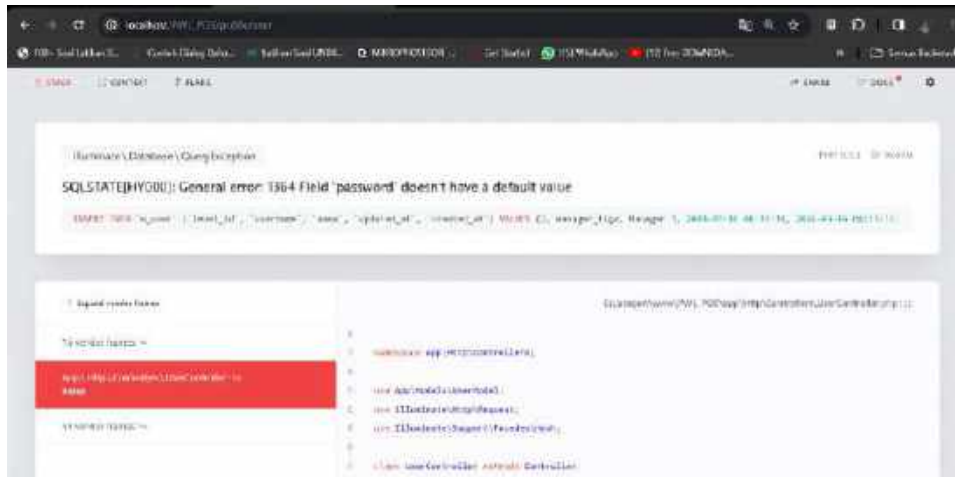
    protected $fillable = ['level_id', 'username', 'nama'];
}
```

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

```
// Data yang akan dimasukkan ke dalam tabel m_user
$data = [
    'level_id' => 2,
    'username' => 'manager_tiga',
    'nama' => 'Manager 3',
    'password' => Hash::make('12345') // Mengenkripsi password sebelum disimpan
];

// Menyimpan data ke dalam tabel m_user menggunakan model UserModel
UserModel::create($data);
```

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



Jawab:

Penggunaan `$fillable` dalam model Laravel sangat penting karena menentukan kolom mana saja yang dapat diisi oleh Eloquent ketika menyimpan data. Error yang Anda alami terjadi karena kolom `'password'` tidak termasuk dalam `$fillable`, sehingga Eloquent tidak mengizinkan pengisian kolom `'password'` secara langsung.

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

\$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*.

Jawab:

\$guarded: Kebalikannya dari \$fillable. Semua kolom yang ditambahkan ke \$guarded akan diabaikan oleh Eloquent saat melakukan operasi insert atau update. Secara default, \$guarded berisi array("*"), yang artinya semua atribut tidak bisa diatur melalui mass assignment.

Praktikum 2.1 – Retrieving Single Models

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

```
$users = UserModel::find(1);  
return view('user', ['data' => $users]);
```

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>  
      <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 link <http://localhost:8000/user> pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan

Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

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

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

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

Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Jawab:

- `UserModel::where('level_id', 1)->first();` mencari data user yang memiliki `level_id` dengan nilai 1 dari tabel yang sesuai dengan model `UserModel`. Fungsi `where('level_id', 1)` digunakan untuk menentukan kondisi pencarian berdasarkan `level_id` yang sama dengan 1, dan kemudian `first()` digunakan untuk mendapatkan data pertama yang sesuai dengan kondisi tersebut.
 - Jadi, potongan kode tersebut mencari data user yang memiliki `level_id` dengan nilai 1, kemudian menampilkan view 'user' dengan data user yang ditemukan.
6. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
$users = UserModel::firstwhere('level_id', 1);  
return view('user', ['data' => $users]);
```

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

Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Jawab:

- `UserModel::firstwhere('level_id', 1);` mencari data user yang memiliki `level_id` dengan nilai 1 dari tabel yang sesuai dengan model `UserModel`. Fungsi `firstwhere('level_id', 1)` digunakan untuk langsung mencari data pertama yang memenuhi kondisi tersebut, tanpa perlu menggunakan `where` dan `first()` secara terpisah.
 - Jadi, potongan kode tersebut mencari data user yang memiliki `level_id` dengan nilai 1, kemudian menampilkan view 'user' dengan data user yang ditemukan.
8. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
$users = UserModel::findOr(1, ['username', 'nama'], function(){
    abort(404);
});
return view('user', ['data' => $users]);
```

8

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

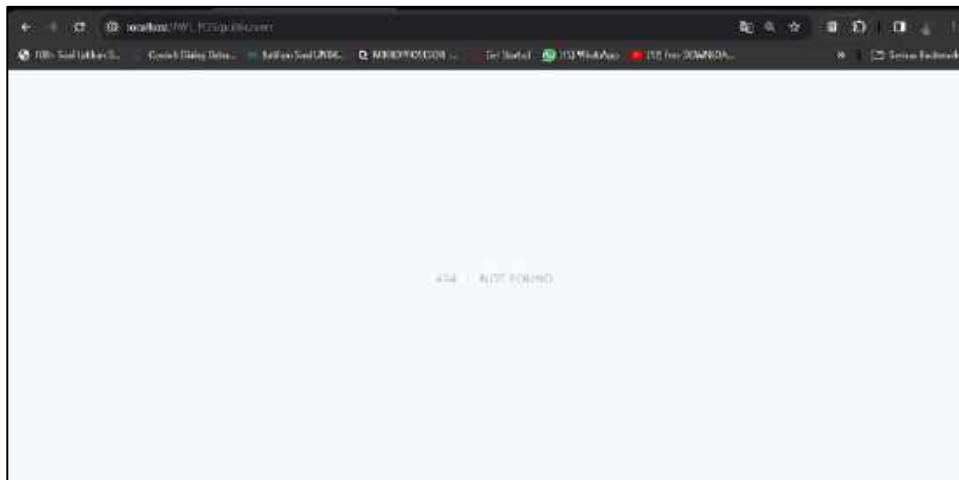
Data User

ID	Username	Nama	ID Level Pengguna
	admin	Administrator	

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

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

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

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

```
$users = UserModel::findOrFail(1);
return view('user', ['data' => $users]);
```

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

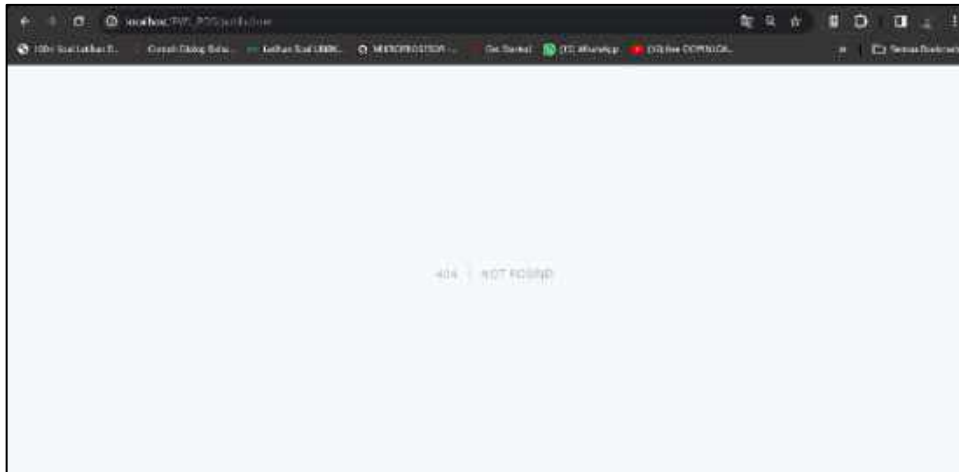
Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

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

```
$users = UserModel::where('username','manager9')->firstOrFail();  
return view('user', ['data' => $users]);
```

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



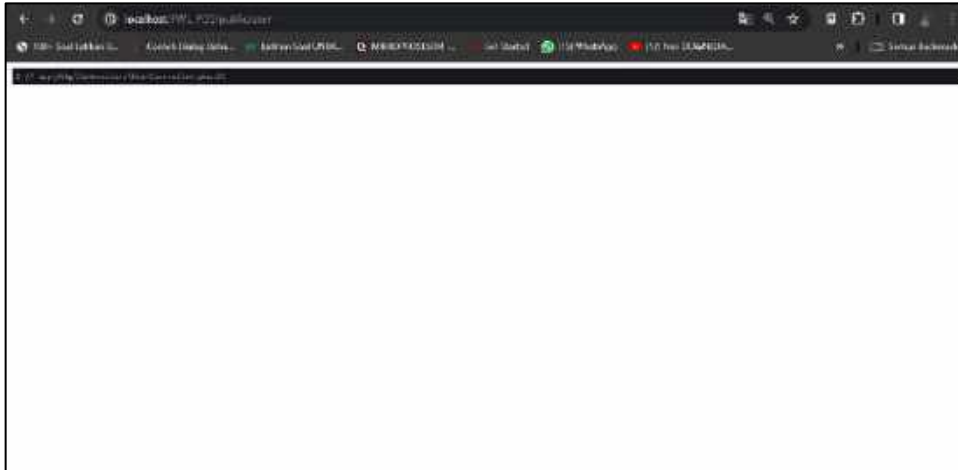
5. Laporkan hasil Praktikum-2.2 ini dan *commit* perubahan pada *git*.

Praktikum 2.3 – Retrieving Aggregates

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

```
$users = UserModel::where('level_id',2)->count();  
dd($users);  
return view('user', ['data' => $users]);
```

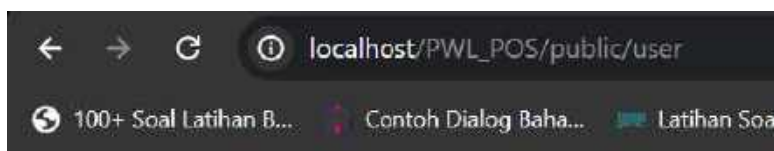
2. Simpan kode program Langkah 1. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan



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

```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Hash;
8
9  class UserController extends Controller
10 {
11     public function index()
12     {
13         $userCount = UserModel::where('level_id', 2)->count();
14         return view('user', ['data' => $userCount]);
15     }
16 }
```

```
resources > views > user.blade.php > ...
1  <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>Jumlah Pengguna</th>
10             </tr>
11
12             <tr>
13                 <td>{{ $data }}</td>
14             </tr>
15         </table>
16     </body>
17 </html>
```



Data User

Jumlah Pengguna
2

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

2.4 – Retrieving or Creating Models

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

```
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>
            <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

Jawab:

Secara sederhana, kode tersebut bertujuan untuk mencari atau membuat pengguna dengan username 'manager' dan nama 'Manager' jika belum ada dalam database.

Data User

ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

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

```
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

Jawab:

`UserModel::firstOrCreate` digunakan untuk mencari atau membuat data pengguna pertama yang memenuhi kriteria yang diberikan. Jika data pengguna dengan username 'manager22' sudah ada, maka data tersebut akan dikembalikan. Jika tidak ada, maka data baru akan dibuat dengan username 'manager22', nama 'Manager Dua Dua', password yang telah dienkripsi menggunakan `Hash::make('12345')`, dan `level_id = 2`.

Data User

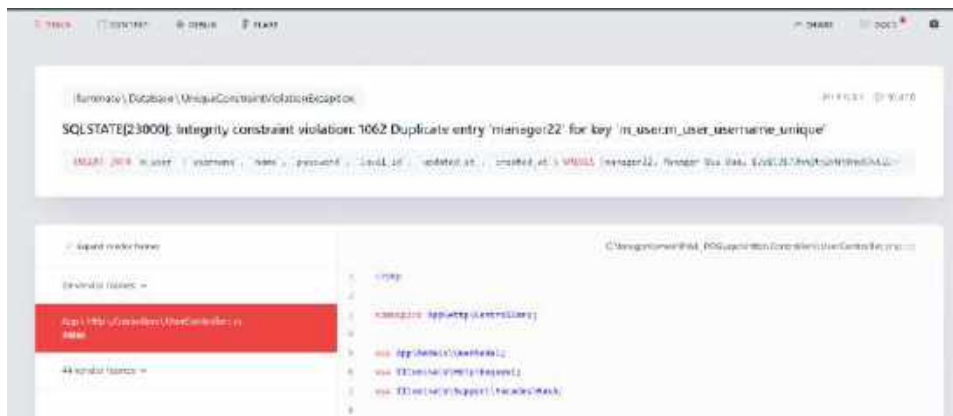
ID	Username	Nama	ID Level Pengguna
15	manager22	Manager Dua Dua	2

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

```
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

Jawab:



Ada masalah dengan username yang sama ('manager22') yang sudah ada dalam database dan mencoba dimasukkan kembali. Ini menyebabkan error karena kolom username harus unik.

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

```
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

Jawab:

Pada kode di atas, `UserModel::firstOrCreate` digunakan untuk mencari data pengguna pertama yang sesuai dengan kriteria yang diberikan. Jika data pengguna dengan username 'manager33' sudah ada, maka data tersebut akan dikembalikan. Jika tidak ada, maka data baru akan dibuat dengan username 'manager33', nama 'Manager Tiga Tiga', password yang telah dienkripsi menggunakan `Hash::make('12345')`, dan `level_id = 2`.

Data User

ID	Username	Nama	ID Level Pengguna
	manager33	Manager Tiga Tiga	2

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


```

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

Jawab:

UserModel::firstOrCreate digunakan untuk mencari data pengguna pertama yang sesuai dengan kriteria yang diberikan. Jika data pengguna dengan username 'manager33' sudah ada, maka data tersebut akan dikembalikan. Jika tidak ada, maka data baru akan dibuat dengan username 'manager33', nama 'Manager Tiga Tiga', password yang telah dienkripsi menggunakan Hash::make('12345'), dan level_id = 2. Kemudian, \$user->save() digunakan untuk menyimpan data pengguna ke dalam database.

Data User

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

ID	Username	Nama	ID Level Pengguna	password	created_at	updated_at
1	admin	Administrator	1	\$2y\$12\$5B40UyG4ne1M3HfMcDh1k5ThapngEED4LrBu...	NULL	NULL
2	manager	Manager	2	\$2y\$12\$3u88y8R8H465uM14w61U1H8a1awwCN5AE...	NULL	NULL
3	staff	Staff	3	\$2y\$12\$3u88y8R8H465uM14w61U1H8a1awwCN5AE...	NULL	NULL
12	manager123	Manager 12	2	\$2y\$12\$3u88y8R8H465uM14w61U1H8a1awwCN5AE...	2024-05-10 08:50:09	2024-05-10 08:50:09
15	manager123	Manager Tiga Tiga	2	\$2y\$12\$3u88y8R8H465uM14w61U1H8a1awwCN5AE...	2024-05-17 08:20:08	2024-05-17 08:20:08
17	manager33	Manager Tiga Tiga	2	\$2y\$12\$3u88y8R8H465uM14w61U1H8a1awwCN5AE...	2024-05-17 08:12:57	2024-05-17 08:12:57

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

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

    $user->username = 'manager55';

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

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

    $user->save();

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

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

Jawab:

Kode tersebut mencoba membuat data pengguna baru dengan username 'manager55', nama 'Manager55', dan password yang dienkripsi. Setelah itu, kode mencoba mengubah nilai username kembali menjadi 'manager55' dan memeriksa apakah ada perubahan yang terjadi pada data menggunakan metode `isDirty` dan `isClean`. Namun, karena penggunaan `isDirty` setelah menyimpan data, yang seharusnya dilakukan sebelum penyimpanan, hasilnya akan selalu `false`.

```
false // app\Http\Controllers\UserController.php:36
```

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

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

    $user->username = 'manager12';

    $user->save();

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

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

```
true // app\Http\Controllers\UserController.php:28
```

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

Praktikum 2.6 – Create, Read, Update, Delete (CRUD)

1. Buka file *view* pada `user.blade.php` dan buat scriptnya 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>
      <th>ID</th>
      <th>Username</th>
      <th>Nama</th>
      <th>ID Level Pengguna</th>
      <th>Aksi</th>
    </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

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

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
12	manager_dua	Manager 2	2	Ubah Hapus
15	manager22	Manager Dua Dua	2	Ubah Hapus
17	manager33	Manager Tiga Tiga	2	Ubah Hapus
18	manager55	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus

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

```
<?doctype html>
<html>
<head>
  <title>Form Tambah Data User</title>
</head>
<body>
  <h1>Form Tambah Data User</h1>
  <form method="post" action="/user/tambah_simpan">
    {{ csrf_field() }}
    <label for="username">Username</label>
    <input type="text" id="username" name="username" placeholder="Masukkan Username" required>
    <br>
    <label for="nama">Nama</label>
    <input type="text" id="nama" name="nama" placeholder="Masukkan Nama" required>
    <br>
    <label for="password">Password</label>
    <input type="password" id="password" name="password" placeholder="Masukkan Password" required>
    <br>
    <label for="level_id">Level ID</label>
    <input type="number" id="level_id" name="level_id" placeholder="Masukkan ID Level" required>
    <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

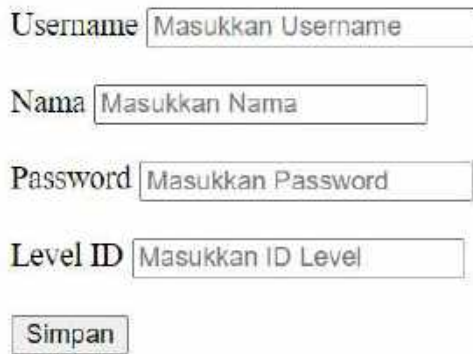
```
Route::get('/user/tambah', [UserController::class, 'tambah']);
```

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

```
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

Form Tambah Data User



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

```
Route::get('/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

Form Tambah Data User

Username

Nama

Password

Level ID

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
12	manager dua	Manager 2	2	Ubah Hapus
15	manager22	Manager Dua Dua	2	Ubah Hapus
17	manager33	Manager Tiga Tiga	2	Ubah Hapus
18	manager55	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus
27	manager16	manager16	2	Ubah Hapus

11. Langkah berikutnya membuat *update* atau ubah data user dengan cara bikin file baru pada *view* dengan nama `user_ubah.blade.php` dan buat scriptnya menjadi seperti di bawah ini

[illegible]

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

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

13. Tambahkan *script* pada controller dengan nama file `UserController.php`. Tambahkan *script* dalam class dan buat method baru dengan nama `ubah` dan diletakan di bawah method `tambah_simpan` seperti gambar di bawah ini

```
public function ubah($id)
{
    $user = UserModel::find($id);
    return view('user_ubah', ['data'=> $user]);
}
```

14. Simpan kode program Langkah 11 sd 13. Kemudian jalankan pada *browser* dan klik link “Ubah” amati apa yang terjadi dan beri penjelasan dalam laporan



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

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

16. Tambahkan *script* pada controller dengan nama file `UserController.php`. Tambahkan *script* dalam class dan buat method baru dengan nama `ubah_simpan` dan diletakan di bawah method `ubah` seperti gambar di bawah ini

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

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

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

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

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
12	manager dua	Manager 2	2	Ubah Hapus
15	manager22	Manager Dua Dua	2	Ubah Hapus
17	manager33	Manager Tiga Tiga	2	Ubah Hapus
18	manager55	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus
27	manager16	manager16	2	Ubah Hapus

Form Ubah Data User

[Kembali](#)

Username

Nama

Password

Level ID

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
12	manager_dua	Manager 2	2	Ubah Hapus
15	manager22	Manager Dua Dua	2	Ubah Hapus
17	manager33	Manager Tiga Tiga	2	Ubah Hapus
18	manager55	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus
27	manager16	Fanesa	2	Ubah Hapus

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

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

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

```
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
12	manager_dua	Manager 2	2	Ubah Hapus
15	manager22	Manager Dua Dua	2	Ubah Hapus
17	manager33	Manager Tiga Tiga	2	Ubah Hapus
18	manager55	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus
27	manager16	Fanesa	2	Ubah Hapus

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
12	manager_dua	Manager 2	2	Ubah Hapus
15	manager22	Manager Dua Dua	2	Ubah Hapus
17	manager33	Manager Tiga Tiga	2	Ubah Hapus
18	manager55	Manager55	2	Ubah Hapus
24	manager12	Manager11	2	Ubah Hapus

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

```
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  use Illuminate\Database\Eloquent\Relations\BelongsTo;
8
9  use App\Models\LevelModel; // Add this import statement
10
11 class UserModel extends Model
12 {
13     use HasFactory;
14
15     protected $table = 'm_user';
16     protected $primaryKey = 'user_id';
17
18     protected $fillable = ['level_id', 'username', 'nama', 'password'];
19
20     public function level(): BelongsTo
21     {
22         return $this->belongsTo(LevelModel::class);
23     }
24 }
```

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

```
<?php

namespace App\Models;

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

class LevelModel extends Model
{
    use HasFactory;

    protected $table = 'm_level';
    protected $primaryKey = 'level_id';

    protected $fillable = ['level_id', 'level_kode', 'level_nama'];

    public function users(): HasMany
    {
        return $this->hasMany(UserModel::class, 'level_id', 'level_id');
    }
}
```

```
Illuminate\Database\Eloquent\Collection {#318 ▼
  #items: array:8 [▶]
  #escapeWhenCastingToString: false
}
```

Mereturnkan bahwa terdapat 8 objek didalam database m_user

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



```

<body>
  <h1>Data User</h1>
  <a href="user/tambah">+ Tambah User</a>
  <table border="1" cellpadding="2" cellspacing="0">
    <tr>
      <th>ID</th>
      <th>Username</th>
      <th>Nama</th>
      <th>ID Level Pengguna</th>
      <th>Kode Level</th>
      <th>Nama Level</th>
      <th>Aksi</th>
    </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 ? $d->level->level_kode : '-' }}</td>
        <td>{{ $d->level ? $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>

```

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	Sta/Kasir	Ubah Hapus
12	manager_dua	Manager 2	2	MNG	Manager	Ubah Hapus
15	manager22	Manager Dua Dua	2	MNG	Manager	Ubah Hapus
17	manager33	Manager Tiga Tiga	2	MNG	Manager	Ubah Hapus
18	manager55	Manager55	2	MNG	Manager	Ubah Hapus
24	manager12	Manager11	2	MNG	Manager	Ubah Hapus

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

*** Sekian, dan selamat belajar ***