



v

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

JOBSHEET 04

MODEL dan ELOQUENT ORM

Sebelumnya kita sudah membahas mengenai *Migration*, *Seeder*, *DB Façade*, *Query Builder*, dan sedikit tentang *Eloquent ORM* yang ada di Laravel. Sebelum kita masuk pada pembuatan aplikasi berbasis website, alangkah baiknya kita perlu menyiapkan Basis data sebagai tempat menyimpan data-data pada aplikasi kita nanti. Selain itu, umumnya kita perlu menyiapkan juga data awal yang kita gunakan sebelum membuat aplikasi, seperti data user administrator, data pengaturan sistem, dll.

Dalam pertemuan kali ini kita akan memahami tentang bagaimana cara menampilkan data, mengubah data, dan menghapus data menggunakan teknik Eloquent.

Sesuai dengan **studi Kasus PWL.pdf**.

Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL_POS**.

Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

ORM (Object Relation Mapping) merupakan teknik yang merubah suatu table menjadi sebuah object yang nantinya mudah untuk digunakan. Object yang dibuat memiliki property yang sama dengan field — field yang ada pada table tersebut. ORM tersebut bertugas sebagai penghubung dan sekaligus mempermudah kita dalam membuat aplikasi yang menggunakan database relasional agar menjadikan tugas kita lebih efisien.

Kelebihan - Kelebihan Menggunakan ORM

1. Terdapat banyak fitur seperti transactions, connection pooling, migrations, seeds, streams, dan lain sebagainya.



2. perintah query memiliki kinerja yang lebih baik, daripada kita menulisnya secara manual.
 3. Kita menulis model data hanya di satu tempat, sehingga lebih mudah untuk update, maintain, dan reuse the code.
 4. Memungkinkan kita memanfaatkan OOP (object oriented programming) dengan baik
- Di Laravel sendiri telah disediakan Eloquent ORM untuk mempermudah kita dalam melakukan berbagai macam query ke database, dan membuat pekerjaan kita menjadi lebih mudah karena tidak perlu menuliskan query sql yang panjang untuk memproses data.

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

1. `$fillable`

Variable `$fillable` berguna untuk mendaftarkan atribut (nama kolom) yang bisa kita isi ketika melakukan insert atau update ke database. Sebelumnya kita sudah memahami menambahkan record baru ke database. Untuk langkah menambahkan Variable `$fillable` bisa dengan menambahkan *script* seperti di bawah ini pada file model

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

Praktikum 1 - `$fillable`:

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

```
class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = ['level_id', 'username', 'nama', 'password'];
}
```

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



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

3. Simpan kode program Langkah 1 dan 2, dan jalankan perintah web server. Kemudian jalankan link localhostPWL_POS/public/user 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	Staf/Kasir	3
4	customer-1	Pelanggan Pertama	4
5	manager_dua	Manager 2	2

Terdapat inputan data baru pada tabel m_user (ID NO 5)

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

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

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



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

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

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

SQLSTATE[HY000]: General error: 1364 Field 'password' doesn't have a default value

Atribut password belum didaftarkan melalui \$fillable sehingga tidak bisa melakukan insert data ke tabel m_user secara langsung



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

2. `$guarded`

Kebalikan dari `$fillable` adalah `$guarded`. Semua kolom yang kita tambahkan ke `$guarded` akan diabaikan oleh Eloquent ketika kita melakukan insert/update. Secara default `$guarded` isinya `array("*")`, yang berarti semua atribut tidak bisa diset melalui **mass assignment**. **Mass Assignment** adalah fitur canggih yang menyederhanakan proses pengaturan beberapa atribut model sekaligus, menghemat waktu dan tenaga. Pada praktikum ini, kita akan mengeksplorasi konsep penugasan massal di Laravel dan bagaimana hal itu dapat dimanfaatkan secara efektif untuk meningkatkan alur kerja pengembangan Anda.

B. RETRIEVING SINGLE MODELS

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

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

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

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

Praktikum 2.1 – Retrieving Single Models

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

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



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

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

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

Data User

ID	Username	Nama	ID Level Pengguna
1	admin		1

`$user = UserModel::find(1);` mencari field PK pada tabel `m_user` dimana field tersebut adalah `user_id` dengan nilai = 1. Hasilnya seperti gambar diatas

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

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

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



Data User

ID	Username	Nama	ID Level Pengguna
1	admin		1

`$user = UserModel::where('level_id', 1)->first();` mencari data berdasarkan **kolom level_id** dengan nilai 1. Ditampilkan seperti tampilan diatas.

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

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



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

Data User

ID	Username	Nama	ID Level Pengguna
1	admin		1

`$user = UserModel::firstWhere('level_id', 1);` merupakan shortcut dari `where('level_id', 1)->first();` dimana mencari field `level_id` dengan nilai 1

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

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

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

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

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



Data User

ID	Username	Nama	ID Level Pengguna
	admin		

findfor() digunakan untuk mencari data pada suatu tabel. Pada kode program diatas dicari nilai = 1 pada field PK yaitu user_id namun yang ditampilkan hanya field username dan nama.

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::findOrFail(20, ['username', 'nama'], function () {
            abort(404);
        });

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

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

404 | NOT FOUND

Pada kode program diatas, `findOrFail` mencari `user_id = 20` dan menampilkan hanya field `username` dan `nama`. Namun pada tabel `m_user` tidak ada `user_id=20` sehingga ditampilkan pesan 'Not Found'

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

Praktikum 2.2 – Not Found Exceptions

Terkadang Anda mungkin ingin memberikan pengecualian jika model tidak ditemukan. Hal ini sangat berguna dalam *route* atau pengontrol. Metode `findOrFail` and `firstOrFail` akan mengambil hasil pertama dari kueri; namun, jika tidak ada hasil yang ditemukan, sebuah `Illuminate\Database\Eloquent\ModelNotFoundException` akan dilempar. Berikut ikuti langkah-langkah di bawah ini:

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

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

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



Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Pada kode program diatas, findOrFail(1) mencari field PK yaitu field user_id dengan nilai 1. Ditemukan data seperti diatas.

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



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

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

404 | NOT FOUND

`firstOrFail()` mencari data dengan kondisi tertentu. Pada kode diatas dicari data dengan `username = manager9`. Namun pada tabel `m_user` tidak ada sehingga ditampilkan pesan `Not Found`

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

SINI

Praktikum 2.3 – Retrieving Aggregates

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

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

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

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

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



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

\$user = UserModel::where('level_id', 2)->count(); digunakan untuk menghitung jumlah level_id=2. Hasilnya adalah 2.

dd(\$user); adalah **Dump and Die**, untuk menampilkan nilai dari variabel \$user dan menghentikan eksekusi program setelahnya sehingga tidak ada view yang dirender

// app\Http\Controllers\UserController.php:14 menunjukkan bahwa output berasal dari **baris ke-14 di UserController.php**, tempat fungsi dd(\$user); dieksekusi

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

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

Praktikum 2.4 – Retrieving or Creating Models

Metode `firstOrCreate` merupakan metode untuk melakukan *retrieving data* (mengambil data) berdasarkan nilai yang ingin dicari, jika data tidak ditemukan maka method ini akan melakukan insert ke table database tersebut sesuai dengan nilai yang dimasukkan.

Metode `firstOrCreate`, seperti `firstOrCreate`, akan mencoba menemukan/mengambil *record/data* dalam database yang cocok dengan atribut yang diberikan. Namun, jika data tidak ditemukan, data akan disiapkan untuk di-*insert*-kan ke database dan model baru akan dikembalikan. Perhatikan bahwa model yang dikembalikan `firstOrCreate` belum disimpan ke database. Anda perlu memanggil metode `save()` secara manual untuk menyimpannya:

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

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

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



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

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

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

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

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

Data User

ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

`firstOrCreate` digunakan untuk mencari data di database berdasarkan kondisi yang diberikan. Jika data ditemukan, maka data yang ada akan dikembalikan. Jika tidak ditemukan, Laravel akan **membuat** (insert) data baru dengan nilai yang diberikan. Dari `$user` kita mengetahui bahwa yang dicari adalah `username=manager` dan pada database data tersebut ada

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



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
<http://www.polinema.ac.id>

Reza Angelina Febriyanti / 27 / SIB 2A

di bawah ini



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

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

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

Data User

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

`firstOrCreate` pada kode program diatas digunakan untuk mencari `username=manager22`. Didatabase tidak ada data tersebut sehingga `firstOrCreate` melakukan insert ke tabel `m_user`

user_id	level_id	username	nama	password	created_at	updated_at
1	1	admin	Administrator	\$2y\$12\$iAKc0U2nsQAVaBzTU5Kq/OxbnL10DyHe1Zh6t6bmje...	NULL	NULL
2	2	manager	Manager	\$2y\$12\$3L/c7Xx3YGYuyML98Edo0OhdGFD2t8jMEP42LTdWFT...	NULL	NULL
3	3	staff	Staff/Kasir	\$2y\$12\$uw1YvyYGGFTuoq4oJ37tFjk.SoptYimMkM1F0wJHJO...	NULL	NULL
4	4	customer-1	Pelanggan Pertama	\$2y\$12\$IOpmbp1t1ihb4PmmH4pEO.COh7Xzba0CKpqCeAAkZX...	NULL	2025-03-06 08:58:46
5	2	manager_dua	Manager 2	\$2y\$12\$u6c9w9hUNMY2iFQA9elsudo/arRUdB4r7cn5r609d9...	2025-03-08 04:25:25	2025-03-08 04:25:25
6	2	manager22	Manager Dua Dua	\$2y\$12\$eeYVwRUQVlm5xP5SgLiB.3SYbh/5CmSmGkeWSTdxgQ...	2025-03-08 15:17:55	2025-03-08 15:17:55

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

Data User

ID	Username	Nama	ID Level Pengguna
2	manager	Manager	2

Ditampilkan data dengan username=manager

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



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

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

9. Simpan kode program Langkah 8. Kemudian jalankan pada *browser* dan amati apa yang terjadi dan cek juga pada *phpMyAdmin* pada tabel *m_user* serta beri penjelasan dalam laporan

Data User

ID	Username	Nama	ID Level Pengguna
	manager23	Manager Tiga Tiga	2

`firstOrCreate` mencari data dengan `username=manager23`. Namun pada database tidak ada namun tampilan diatas menampilkannya namun kolom id tidak ada outputnya

user_id	level_id	username	nama	password	created_at	updated_at
1	1	admin	Administrator	\$2y\$12\$IAKc0U2nsQA/vaBzTU5Kq/ObxnbL10DyHe1Zhc6t8mje...	NULL	NULL
2	2	manager	Manager	\$2y\$12\$3L/c7Xx3YGXuyML98Edo0OhdGFD2t8jMEP42LTd2WTT...	NULL	NULL
3	3	staff	Staf/Kasir	\$2y\$12\$uw1YvyYGGFTuoq4oJ37Fjk.SoptYimMkM1F0wJHJO...	NULL	NULL
4	4	customer-1	Pelanggan Pertama	\$2y\$12\$IOPmbp1t1ihb4PmmH4pE0O.COh7Xzba0CKpqCeAAkZX...	NULL	2025-03-06 08:58:46
5	2	manager_dua	Manager 2	\$2y\$12\$u6c9w9hUNMY2iFQAcs9elsudo/arRUdB4r7cn5r609d9...	2025-03-08 04:25:25	2025-03-08 04:25:25
6	2	manager22	Manager Dua Dua	\$2y\$12\$eeYVwRUQVIm5xP5SlgLiB.3SYbh/5CmSmGkeWSTdxgQ...	2025-03-08 15:17:55	2025-03-08 15:17:55

Nyatanya didatabase data tersebut belum ditambahkan

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



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

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

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

Data User

ID	Username	Nama	ID Level Pengguna
7	manager23	Manager Tiga Tiga	2

Tampilan diatas terdapat kolom id yang sebelumnya tidak ada inputannya

user_id	level_id	username	nama	password	created_at	updated_at
1	1	admin	Administrator	\$2y\$12\$IAKc0U2nsQAVaBzTU5Kq/ObxnbL10DyHe1Zhcf8tmje...	NULL	NULL
2	2	manager	Manager	\$2y\$12\$3L/c7Xx3YGYuML98Edo0OhdGFD2t8JMEP42LTd2WIT...	NULL	NULL
3	3	staff	Staff/Kasir	\$2y\$12\$uw1YvyYGGFTuoq4oJ37fFjk.SoptYimMkM1F0wJHJO...	NULL	NULL
4	4	customer-1	Pelanggan Pertama	\$2y\$12\$IOpmbp1t1ihb4PmmH4pE0O.COh7Xzba0CKpqCeAAkZX...	NULL	2025-03-06 08:58:41
5	2	manager_dua	Manager 2	\$2y\$12\$u6c9w9hUNMY2iFQAc9elsudo/arRUdB4r7cn5r609d9...	2025-03-08 04:25:25	2025-03-08 04:25:25
6	2	manager22	Manager Dua Dua	\$2y\$12\$eeYVwRUQVIm5xP5SIglJB.3SYbh/5CmSmGkeWSTdxgQ...	2025-03-08 15:17:55	2025-03-08 15:17:55
7	2	manager23	Manager Tiga Tiga	\$2y\$12\$8TDKtNCTzwVPajZYpWq/w.ZuYy.GTRsPzodAOsdqmWQ...	2025-03-08 15:44:29	2025-03-08 15:44:29

Pada database username=manager23 telah ditambahkan

Jadi `$user->save()`; digunakan untuk menjalankan query insert yg fiks disimpan di database

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



Praktikum 2.5 – Attribute Changes LANJUTT

Eloquent menyediakan metode `isDirty`, `isClean`, dan `wasChanged` untuk memeriksa keadaan internal model Anda dan menentukan bagaimana atributnya berubah sejak model pertama kali diambil.

Metode `isDirty` menentukan apakah ada atribut model yang telah diubah sejak model diambil. Anda dapat meneruskan nama atribut tertentu atau serangkaian atribut ke metode `isDirty` untuk menentukan apakah ada atribut yang "kotor". Metode ini `isClean` akan menentukan apakah suatu atribut tetap tidak berubah sejak model diambil. Metode ini juga menerima argumen atribut opsional:

```
$user = UserModel::create([
    'username' => 'manager44',
    'nama' => 'Manager44',
    'password' => Hash::make('12345'),
    'level_id' => 2,
]);

$user->username = 'manager45';

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

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

$user->save();

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

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



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

        $user->username = 'manager56';

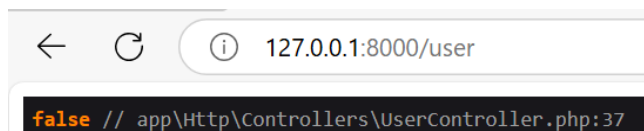
        $user->isDirty(); // true
        $user->isDirty('username'); // true
        $user->isDirty('nama'); // false
        $user->isDirty(['nama', 'username']); // true

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

        $user->save();

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

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



Hasil diatas memunculkan pesan false dari `dd($user->isDirty());`; yang artinya tidak ada perubahan pada database. Namun sebelumnya username=manager55 saya hapus dari database sehingga pada database username=manager56 bisa dibuat.

```
mysql> SELECT * FROM m_user;
```

	user_id	level_id	username	nama	password
	created_at		updated_at		
	1	1	admin	Administrator	\$2y\$12\$IAKc0U2nsQAVaBzTU5Kq/ObxnbL180DyHe1Zhc6tbmjea8Lf8335xw
	2	2	manager	Manager	\$2y\$12\$3L/c7Xx3YGxuyML98Edo0HdGFD2t8jMEP42LTd2WFTwoHgJrEu9S
	3	3	staff	Staf/Kasir	\$2y\$12\$um1YvyYGGFTuoq4oJ37tF.jk.SoptYimMkH1F0wJHJOMk6JQxgMgy
	4	4	customer-1	Pelanggan Pertama	\$2y\$12\$IOPmbp1t1ihb4PmmH4pE00.CO7Xzba8CHpqCeAAkZX5xf8QmLu3u
	5	2	manager_dua	Manager 2	\$2y\$12\$u6c9w9hUNMY2iFQAc9eLsdo/arRudB4r7cn5r69d95t5KqOqcVq
	2025-03-08 04:25:25	2025-03-08 04:25:25			
	6	2	manager22	Manager Dua Dua	\$2y\$12\$eeYVwRUQVln5xP55IglJB.3SYbh/5CnSmGkeWSTdxgQsobBDD/n12
	2025-03-08 15:17:55	2025-03-08 15:17:55			
	7	2	manager23	Manager Tiga Tiga	\$2y\$12\$8TDKhNCTzwVPajZYpMq/w.ZuYy.GTRsPzodA0sdqmQ8/DuXL6N1.
	2025-03-08 15:44:29	2025-03-08 15:44:29			
	14	2	manager56	Manager55	\$2y\$12\$zrNzndUDgkRoVxxsGtxS1.RiqnfHRULFN0rDCEYqISxY.cLhYVphq
	2025-03-09 04:04:29	2025-03-09 04:04:29			

8 rows in set (0.00 sec)

isDirty() mengembalikan true jika ada setidaknya satu perubahan dalam model

isClean() akan mengembalikan true jika tidak ada perubahan dalam model



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
<http://www.polinema.ac.id>

Reza Angelina Febriyanti / 27 / SIB 2A

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

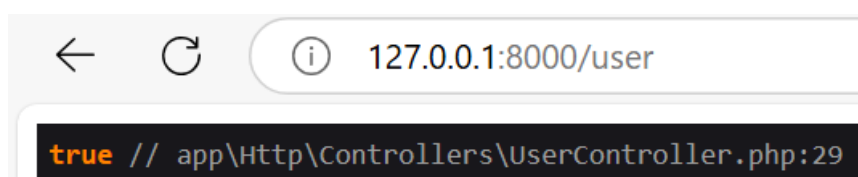
```
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::create([
            'username' => 'manager11',
            'nama' => 'Manager11',
            'password' => Hash::make('12345'),
            'level_id' => 2,
        ]);

        $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





Hasilnya true, artinya proses insert berhasil dilakukan pada database dan didatabase seperti berikut

```
mysql> SELECT * FROM m_user;
```

user_id	level_id	username	nama	password
created_at		updated_at		
1	1	admin	Administrator	\$2y\$12\$iAKc0U2nsQAVaBzTU5Hq/ObxnbL10DyHe1Zhc6tmbjea8lf8335xW
NULL	2	manager	Manager	\$2y\$12\$3L/c7Xx3YGXuyML98Edo0hdGFD2t8jMEP42Ld2WfTwoHgJrEu9S
NULL	3	staff	Staf/Kasir	\$2y\$12\$uw1YvyYGGFTuoq4oJ37tF.jk.SeptYimMkM1F8wJHJOMk6JQxgMGy
NULL	4	customer-1	Pelanggan Pertama	\$2y\$12\$IOPmbp1t1ihb4PnmH4pE80.COh7Xzba8CKpqCeAAkZYSxf8QmLu3u
2025-03-08 04:25:25	2	manager_dua	Manager 2	\$2y\$12\$u6c9w9hUNMY2iFQAc9eLsdo/arRudB4r7cn5r609d95t5KqOqcYq
2025-03-08 15:17:55	2	manager22	Manager Dua Dua	\$2y\$12\$eeYVwRUQVlm5xP5SIgLjB.3SYbh/5CmSmGkeWSTdxgQsobBDD/n12
2025-03-08 15:44:29	2	manager23	Manager Tiga Tiga	\$2y\$12\$8TDKhNCTzwVPajZYpWq/w.ZuYy.GTRsPzodA0sdqmWQ8/DuXL6Ni.
2025-03-09 04:04:29	2	manager56	Manager55	\$2y\$12\$rWzndUDgkRoVxxsGtxSi.RiqnfHRULFN8rDCtYISxY.cLhYVphq
2025-03-09 04:22:53	2	manager12	Manager11	\$2y\$12\$Eo3mgkAk3nKemuyXLfG8OSTQ/0iPrsce1cPH5mEitadhMWiBMQq

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

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



Seperti yang telah kita ketahui, CRUD merupakan singkatan dari *Create*, *Read*, *Update* dan *Delete*. CRUD merupakan istilah untuk proses pengolahan data pada database, seperti input data ke database, menampilkan data dari database, mengedit data pada database dan menghapus data dari database. Ikuti langkah-langkah di bawah ini untuk melakukan CRUD dengan Eloquent

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

```
<body>
<h1>Data User</h1>
<a href="/user/tambah">+ Tambah User</a>
<table border="1" cellpadding="2" cellspacing="0">
  <tr>
    <td>ID</td>
    <td>Username</td>
    <td>Nama</td>
    <td>ID Level Pengguna</td>
    <td>Aksi</td>
  </tr>
  @foreach ($data as $d)
    <tr>
      <td>{{ $d->user_id }}</td>
      <td>{{ $d->username }}</td>
      <td>{{ $d->nama }}</td>
      <td>{{ $d->level_id }}</td>
      <td><a href="/user/ubah/{{ $d->user_id }}">Ubah</a> | <a href="/user/hapus/{{ $d->user_id }}">Hapus</a></td>
    </tr>
  @endforeach
</table>
</body>
```

2. Buka file controller pada `UserController.php` dan buat scripnya untuk *read* menjadi seperti di bawah ini

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

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

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	Staf/Kasir	3	Ubah Hapus
4	customer-1	Pelanggan Pertama	4	Ubah Hapus
5	manager_dua	Manager 2	2	Ubah Hapus
6	manager22	Manager Dua Dua	2	Ubah Hapus
7	manager23	Manager Tiga Tiga	2	Ubah Hapus
14	manager56	Manager55	2	Ubah Hapus
15	manager12	Manager11	2	Ubah Hapus



`+ Tambah User` membuat tombol Tambah User

`<td>user_id }}">Ubah | user_id }}">Hapus</td>` membuat tombol Ubah Hapus

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



```
<body>
<h1>Form Tambah Data User</h1>
<form method="post" action="/user/tambah_simpan">

    {{ csrf_field() }}

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

</form>
</body>
```

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

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

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

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

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

7. Simpan kode program Langkah 4 s/d 6. Kemudian jalankan pada *browser* dan klik link “+ **Tambah User**” amati apa yang terjadi dan beri penjelasan dalam laporan

Username	<input type="text" value="Masukkan Username"/>
Nama	<input type="text" value="Masukkan Nama"/>
Password	<input type="password" value="Masukkan Password"/>
Level ID	<input type="text" value="Masukkan ID Level"/>
<input type="button" value="Simpan"/>	

Tombol Tambah User mengarah ke halaman `user_tambah.blade` yang digunakan untuk



menginputkan data baru

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

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



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

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

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

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

Saya menginputkan sebagai berikut

Username	<input type="text" value="Customer2"/>
Nama	<input type="text" value="Customer ke-2"/>
Password	<input type="password" value="...."/>
Level ID	<input type="text" value="4"/>
<input type="button" value="Simpan"/>	

Saat klik tombol Simpan pada inputan tersebut berhasil ditambahkan

16	4	Customer2	Customer ke-2	\$2y\$12\$oxyFa62owT58HC4T1Ngyxu.azwcqOApY1sUx225z8P2...	2025-03-09 05:26:24	2025-03-09 05:26:24
----	---	-----------	---------------	--	---------------------	---------------------

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



```
<body>
  <h1>Form Ubah Data User</h1>
  <a href="/user">Kembali</a>
  <br><br>

  <form method="post" action="/user/ubah_simpan/{{ $data->user_id }}">
    {{ csrf_field() }}
    {{ method_field('PUT') }}

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

12. Tambahkan *script* pada *routes* dengan nama file [web.php](#). Tambahkan seperti gambar di bawah ini

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



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

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

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

Form Ubah Data User

[Kembali](#)

customer2

Nama

Password

Level ID

Saya mengubah data menjadi berikut namun muncul NOT FOUND

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

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

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

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

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

    $user->save();

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

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



Data User

[+ Tambah User](#)

ID	Username	Nama	ID Level Pengguna	Aksi
1	admin	Administrator	1	Ubah Hapus
2	manager	Manager	2	Ubah Hapus
3	staff	Staf/Kasir	3	Ubah Hapus
4	customer-1	Pelanggan Pertama	4	Ubah Hapus
5	manager_dua	Manager 2	2	Ubah Hapus
6	manager22	Manager Dua Dua	2	Ubah Hapus
7	manager23	Manager Tiga Tiga	2	Ubah Hapus
14	manager56	Manager55	2	Ubah Hapus
15	manager12	Manager11	2	Ubah Hapus
16	customer2	Pelanggan Kedua	4	Ubah Hapus

Customer2 berubah menjadi cutomer2

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	Staf/Kasir	3	Ubah Hapus
4	customer-1	Pelanggan Pertama	4	Ubah Hapus
5	manager_dua	Manager 2	2	Ubah Hapus
6	manager22	Manager Dua Dua	2	Ubah Hapus
7	manager23	Manager Tiga Tiga	2	Ubah Hapus
14	manager56	Manager55	2	Ubah Hapus
15	manager12	Manager11	2	Ubah Hapus

Data customer2 telah dihapus dari database

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

Praktikum 2.7 – Relationships

One to One

Hubungan satu-ke-satu adalah tipe hubungan database yang sangat mendasar. Misalnya, suatu `Usermodel` mungkin dikaitkan dengan satu model `Levelmodel`. Untuk mendefinisikan hubungan ini, kita akan menempatkan `Levelmodel` metode pada model `Usermodel`. Metode tersebut `Levelmodel` harus memanggil `hasOne` metode tersebut dan mengembalikan hasilnya. Metode ini `hasOne` tersedia untuk model Anda melalui kelas dasar model `Illuminate\Database\Eloquent\Model`:



```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasOne;

You, 1 second ago | 1 author (You)
class UserModel extends Model
{
    public function level(): HasOne
    {
        return $this->hasOne(LevelModel::class);
    }
}
```



Mendefinisikan Kebalikan dari Hubungan *One-to-one*

Jadi, kita dapat mengakses model `Levelmodel` dari model `Usermodel` kita. Selanjutnya, mari kita tentukan hubungan pada model `Levelmodel` yang memungkinkan kita mengakses user. Kita dapat mendefinisikan kebalikan dari suatu `hasOne` hubungan menggunakan `belongsTo` metode:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class LevelModel extends Model
{
    public function user(): BelongsTo
    {
        return $this->belongsTo(UserModel::class);
    }
}
```

One to Many

Hubungan satu-ke-banyak digunakan untuk mendefinisikan hubungan di mana satu model adalah induk dari satu atau lebih model turunan. Misalnya, 1 kategori mungkin memiliki jumlah barang yang tidak terbatas. Seperti semua hubungan Eloquent lainnya, hubungan satu-ke-banyak ditentukan dengan mendefinisikan metode pada model Eloquent Anda:

```
<?php

namespace App\Models;

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

class KategoriModel extends Model
{
    public function barang(): HasMany
    {
        return $this->hasMany(BarangModel::class, 'barang_id', 'barang_id');
    }
}
```



One to Many (Inverse) / Belongs To

Sekarang kita dapat mengakses semua barang, mari kita tentukan hubungan agar barang dapat mengakses kategori induknya. Untuk menentukan invers suatu **hasMany** hubungan, tentukan metode hubungan pada model anak yang memanggil **belongsTo** tersebut:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\BelongsTo;

class BarangModel extends Model
{
    public function kategori(): BelongsTo
    {
        return $this->belongsTo(KategoriModel::class, 'kategori_id', 'kategori_id');
    }
}
```

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

```
class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = ['level_id', 'username', 'nama', 'password'];

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

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



```
public function index()
{
    $user = UserModel::with('level')->get();
    dd($user);
}
```

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

```
Illuminate\Database\Eloquent\Collection (#320 ▼ // app\Http\Controllers\UserController.php:15)
  #items: array:10 (▼)
    0 => App\Models\UserModel (#327 ▼)
    1 => App\Models\UserModel (#328 ▼)
    2 => App\Models\UserModel (#329 ▼)
    3 => App\Models\UserModel (#330 ▼)
    4 => App\Models\UserModel (#331 ▼)
    5 => App\Models\UserModel (#332 ▼)
    6 => App\Models\UserModel (#333 ▼)
    7 => App\Models\UserModel (#334 ▼)
    8 => App\Models\UserModel (#335 ▼)
    9 => App\Models\UserModel (#336 ▼)
  #escapeWhenCastingToString: false
```

Fungsi `belongsToMany` digunakan untuk mendefinisikan relasi antara tabel `m_user` dan `m_level`

Fungsi `index` untuk membuat query yang menjoinkan kedua tabel diatas

Karena menggunakan `dd($user)` maka hasilnya seperti diatas diperoleh 10 baris

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

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

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

```
<body>
<h1>Data User</h1>
<a href="/user/tambah">+ Tambah User</a>
<table border="1" cellpadding="2" cellspacing="0">
  <tr>
    <td>ID</td>
    <td>Username</td>
    <td>Nama</td>
    <td>ID Level Pengguna</td>
    <td>Kode Level</td>
    <td>Nama Level</td>
    <td>Aksi</td>
  </tr>
  @foreach ($data as $d)
    <tr>
      <td>{{ $d->user_id }}</td>
      <td>{{ $d->username }}</td>
      <td>{{ $d->nama }}</td>
      <td>{{ $d->level_id }}</td>
      <td>{{ $d->level->level_kode }}</td>
      <td>{{ $d->level->level_nama }}</td>
      <td><a href="/user/ubah/{{ $d->user_id }}">Ubah</a> | <a href="/user/hapus/{{ $d->user_id }}">Hapus</a></td>
    </tr>
  @endforeach
</table>
</body>
```

6. Simpan kode program Langkah 4 dan 5. Kemudian jalankan link pada *browser*,



kemudian amati apa yang terjadi dan beri penjelasan dalam laporan

Data User

[+ Tambah User](#)

ID	Username	Nama	ID Level Pengguna	Kode Level	Nama Level	Aksi
1	admin	Administrator	1	ADM	Administrator	Ubah Hapus
2	manager	Manager	2	MNG	Manager	Ubah Hapus
3	staff	Staf/Kasir	3	STF	Staf/Kasir	Ubah Hapus
4	customer-1	Pelanggan Pertama	4	CUS	Pelanggan	Ubah Hapus
5	manager_dua	Manager 2	2	MNG	Manager	Ubah Hapus
6	manager22	Manager Dua Dua	2	MNG	Manager	Ubah Hapus
7	manager23	Manager Tiga Tiga	2	MNG	Manager	Ubah Hapus
14	manager56	Manager55	2	MNG	Manager	Ubah Hapus
15	manager12	Manager11	2	MNG	Manager	Ubah Hapus
17	customer2	Pelanggan Kedua	4	CUS	Pelanggan	Ubah Hapus

Ditampilkan hasil seperti diatas.

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



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
<http://www.polinema.ac.id>

Reza Angelina Febriyanti / 27 / SIB 2A

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