



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Sistem Informasi Bisnis
Semester : 4 (empat)
Pertemuan ke- : 1 (satu)
Nama : Aqila Nur Azza

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

```
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'; // Mendefinisikan nama tabel yang digunakan
13     protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari ta
14
15     /**
16      * The attributes that are mass assignable.
17      *
18      * @var array
19     */
20     protected $fillable = ['level_id', 'username', 'nama', 'password'];
21 }
22
```

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



```
app > Http > Controllers > UserController.php > UserController > index
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         // tambah data user dengan Eloquent Model
14         $data = [
15             'level_id' => 2,
16             'username' => 'manager_dua',
17             'nama' => 'Manager 2',
18             'password' => Hash::make('12345'),
19         ];
20
21         UserModel::create($data); //menambah data user
22
23         // coba akses model UserModel
24         $user = UserModel::all(); // ambil semua data dari tabel m_user
25         return view('user', ['data' => $user]);
26     }
27 }
```

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
5	manager_dua	Manager 2	2

Penjelasan :

Menambahkan data user dengan fungsi create

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()
{
    // tambah data user dengan Eloquent Model
    $data = [
        'level_id' => 2,
        'username' => 'manager_tiga',
        'nama' => 'Manager 3',
        'password' => Hash::make('12345'),
    ];
    UserModel::create($data); //menambah data user

    // coba akses model UserModel
    $user = UserModel::all(); // ambil semua data dari tabel m_user
    return view('user', ['data' => $user]);
}
```

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

```
13 // tambah data user dengan Eloquent Model
14     $data = [
15         'level_id' => 2,
16         'username' => 'manager_tiga',
17         'nama' => 'Manager 3',
18         'password' => Hash::make('12345'),
19
20     ];
21     UserModel::create($data); //menambah data user
--
```

Penjelasan :

Terjadi error pada kode program karna ketika mencoba menyimpan password, tetapi dalam **UserModel**, kolom password tidak diizinkan dalam **mass assignment** karena tidak termasuk dalam **\$fillable**.



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

```
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         $user = UserModel::find(1);
14         return view('user', ['data' => $user]);
15     }
16 }
```



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 pada *browser* dan amati apa yang terjadi dan beri penjelasan dalam laporan

Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1

Penjelasan :

Fungsi Find akan mencari satu data berdasarkan primary key

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

```
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	Administrator	1



Penjelasan :

where('level_id', 1)->first() akan mencari semua data dengan level_id = 1, lalu mengambil **satu data pertama**

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

```
{  
    $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	Administrator	1

Penjelasan :

`firstWhere('level_id', 1)` sama seperti `where(...)->first()`, tetapi lebih singkat. Langsung mencari **satu data pertama** dengan `level_id = 1`

Kesimpulan Perbedaan :

Metode	Penjelasan
<code>find(1)</code>	Mencari satu data berdasarkan primary key (ID).
<code>where('level_id', 1)->first()</code>	Mencari semua data dengan <code>level_id = 1</code> , lalu mengambil satu data pertama
<code>firstWhere('level_id', 1)</code>	Sama seperti <code>where(...)->first()</code> , tetapi lebih singkat. Langsung mencari satu data pertama dengan <code>level_id = 1</code>

Terkadang Anda mungkin ingin melakukan beberapa tindakan lain jika tidak ada hasil yang ditemukan. Metode `findOneOr` dan `firstOneOr` 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::findOneOr(1, 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	Administrator	

Penjelasan :

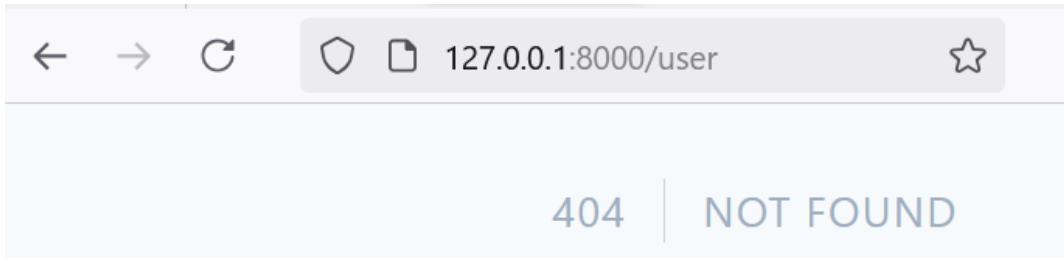
- `findOr()` berguna untuk mencari data berdasarkan ID.
- Jika data tidak ditemukan, bisa menjalankan **callback sebagai alternatif**.

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



```
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         $user = UserModel::findOrFail(20,['username', 'nama'], function() {
14             | abort(404);
15         });
16         return view('user', ['data' => $user]);
17     }
18 }
```

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



Penjelasan :

- Data dengan primary key 20 memang tidak ada dan tidak ditemukan akan menampilkan tampilan diatas.
- Fungsi `abort(404)` di Laravel digunakan untuk menghentikan eksekusi skrip dan mengembalikan HTTP response dengan status code 404 (Not Found).
- Laravel otomatis akan menampilkan halaman error 404, yang berarti sumber daya tidak ditemukan

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 *rute* 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

```
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

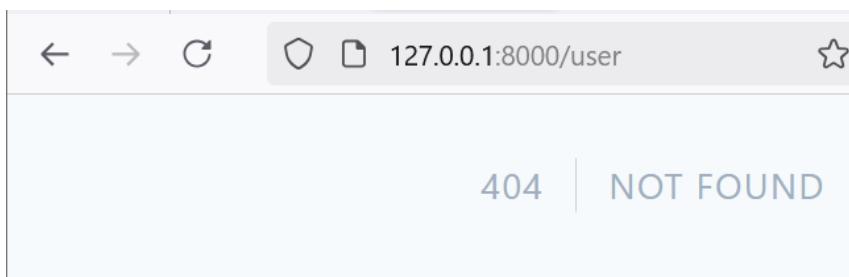
Penjelasan :

- Jika **ID 1 ada** di database, maka data akan diambil dan ditampilkan pada tampilan (view).
 - Jika **ID 1 tidak ada**, Laravel akan otomatis menampilkan **error 404 Not Found**.
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]);
}
```

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



Penjelasan :

- Mencari user dengan username **manager9**.
- Jika ditemukan, mengambil baris pertama yang cocok.
- Jika tidak ditemukan, Laravel akan langsung melemparkan error **404 Not Found** menggunakan `firstOrFail()`.

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

Praktikum 2.3 – Retreiving 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');
```

- 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
```

Penjelasan :

Debugging (dd(\$user);) akan menghentikan Eksekusi dd(\$user); adalah fungsi **dump and die**, jadi ketika dieksekusi, Laravel akan **berhenti** di situ dan tidak melanjutkan ke return view().

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

➤ Controller

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

➤ view

```
resources > views > user.blade.php > html > head > title
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="UTF-8">
5         <meta name="viewport" content="width=device-width, initial-scale=1.0">
6         <title>Data User</title>
7         <style>
8             table {
9                 border-collapse: collapse;
10                width: 200px;
11            }
12            th, td {
13                border: 1px solid black;
14                padding: 8px;
15                text-align: center;
16            }
17        </style>
18    </head>
19    <body>
20        <h1>Data User</h1>
21        <table>
22            <tr>
23                <th>Jumlah Pengguna</th>
24            </tr>
25            <tr>
26                <td>{{ $data }}</td>
27            </tr>
28        </table>
29    </body>
30 </html>
```



Data User

Jumlah Pengguna
2

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

Praktikum 2.4 – Retreiving 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 datadase tersebut sesuai dengan nilai yang dimasukkan.

Metode `firstOrNew`, 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 `firstOrNew` belum disimpan ke database. Anda perlu memanggil metode `save()` secara manual untuk menyimpannya:

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

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

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

Penjelasan :

- `firstOrCreate()` digunakan untuk mencari data berdasarkan kondisi tertentu. Jika tidak ditemukan, Laravel akan otomatis membuatnya.



- Jika sudah ada user dengan username = 'manager' dan nama = 'Manager', maka data tidak akan di-insert ulang.
- Jika belum ada, Laravel akan membuat data baru dengan nilai tersebut.

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

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

> Pastikan sudah terdapat password pada fillable UserModel

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



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

Penjelasan :

- Laravel pertama-tama akan mencari data dengan username = "manager22".
 - Jika tidak ditemukan, maka Laravel akan menambahkan data baru dengan nama, password, dan level_id.
 - Jika sudah ada, maka Laravel hanya akan mengambil data tersebut tanpa mengupdate nilai lainnya.
6. Ubah file controller dengan nama `UserController.php` dan ubah *script* seperti gambar di bawah ini

```
lass 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

Penjelasan :



`firstOrNew()` digunakan ketika jika data ditemukan, Laravel akan mengambilnya sebagai objek Eloquent. Namun, jika data tidak ditemukan, Laravel akan membuat objek baru tanpa langsung menyimpannya ke database.

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:: firstOrNew(
            [
                'username' => 'manager33',
                'nama' => 'Manager Tiga Tiga',
                'password' => Hash::make('12345'),
                'level_id' => 2
            ],
            [
                'data' => $user
            ]
        );
        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
	manager33	Manager Tiga Tiga	2

← ↑ →	▼	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/>	1	1	admin	Administrator	\$2y\$12\$uxsRMuPzbBC3QuuduxLleyKYK2DP1WYqVv7w5kNS7...	NULL	NULL	NULL
<input type="checkbox"/>	2	2	manager	Manager	\$2y\$12\$ya.EH3ZHzPrGD95Vui06leEtVsWIVBuByAjpy17eOym...	NULL	NULL	NULL
<input type="checkbox"/>	3	3	staff	Staff/Kasir	\$2y\$12\$6kktJK0h0GxuZL2lje4HesyhP/p6eutx/NOAR6UgJ2...	NULL	NULL	NULL
<input type="checkbox"/>	5	2	manager_dua	Manager 2	\$2y\$12\$nwFhR3lg07Jm2Us6Mgzn3/iCyUJ/YndUJM6u.Rt6F...	2025-03-07 13:29:25	2025-03-07 13:29:25	2025-03-07 13:29:25
<input type="checkbox"/>	6	2	manager22	Manager Dua Dua	\$2y\$12\$uaySea1JSL2je96xFrpF.qkh4zW9IRoMnGXQIAQ1...	2025-03-07 17:51:05	2025-03-07 17:51:05	2025-03-07 17:51:05

Penjelasan :

- Laravel akan mencari username "33" di database.
 - Karena username "33" tidak ditemukan, Laravel akan membuat objek baru dengan data username "33".
 - Tapi, data ini belum masuk ke database, hanya objek sementara di dalam kode.
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:: 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
7	manager33	Manager Tiga Tiga	2

#	user_id	level_id	username	nama	password	created_at	updated_at
1	1	1	admin	Administrator	\$2y\$12\$uxsRMuPzbBC3QuuduxLleyKYK2DP1WYQvV7w5kNS7... Original length 60	NULL	NULL
2	2	2	manager	Manager	\$2y\$12\$ya.EH3HzPrGD... Original length 60	NULL	NULL
3	3	3	staff	Staff/Kasir	\$2y\$12\$6kkJK0Hc0GxuZL2je4HesyhP/p6eubx/NOAR6UgJ2...	NULL	NULL
5	5	2	manager_dua	Manager 2	\$2y\$12\$nwFrR3lg07Jm2Us6Mgnz3.iCyUJYrndUJM6u.RtBF...	2025-03-07 13:29:25	2025-03-07 13:29:25
6	6	2	manager22	Manager Dua Dua	\$2y\$12\$uaySea1SL2je96xFrpFqkh4zWPIRcoMnGXQIAQ1I...	2025-03-07 17:51:05	2025-03-07 17:51:05
7	7	2	manager33	Manager Tiga Tiga	\$2y\$12\$FW3z0d5KlnFj7zlIMmqvFuJ66lWHuz/g5/KCQialRzp...	2025-03-07 18:07:39	2025-03-07 18:07:39

Penjelasan :

- `$user->save();` adalah perintah dalam Eloquent Laravel untuk menyimpan data ke database.
- Jika menggunakan `firstOrNew` hanya akan membuat objek baru namun tidak tersimpan ke database, maka dibutuhkan `$user->save();` untuk menyimpan ke database.

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



Praktikum 2.5 – Attribute Changes

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

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

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

$user->username = 'manager45';

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

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

$user->save();

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

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



```
app > Http > Controllers > UserController.php > UserController > f index
  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          $user = UserModel::create([
14              'username' => 'manager55',
15              'nama' => 'Manager55',
16              'password' => Hash::make('12345'),
17              'level_id' => 2,
18          ]);
19
20          $user->username = 'manager56';
21
22          $user->isDirty(); // true
23          $user->isDirty('username'); // true
24          $user->isDirty('nama'); // false
25          $user->isDirty(['nama', 'username']); // true
26
27          $user->isClean(); // false
28          $user->isClean('username'); // false
29          $user->isClean('nama'); // true
30          $user->isClean(['nama', 'username']); // false
31
32          $user->save();
33
34          $user->isDirty(); // false
35          $user->isClean(); // true
36          dd($user->isDirty());
37      }
}
```

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

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

Penjelasan :

Output false pada app\Http\Controllers\UserController.php:36 adalah hasil dari perintah dd(\$user->isDirty());, yang menunjukkan bahwa objek \$user tidak memiliki perubahan setelah save() dipanggil.

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

```
7 use Illuminate\Support\Facades\Hash;
8
9 class UserController extends Controller
10 {
11     public function index()
12     {
13         $user = UserModel::create([
14             'username' => 'manager11',
15             'nama' => 'Manager11',
16             'password' => Hash::make('12345'),
17             'level_id' => 2,
18             //Data berhasil disimpan ke database dengan username manager11
19         ]);

20
21         $user->username = 'manager12';
22         // Sekarang ada perubahan, tetapi belum disimpan ke database.
23
24         $user->save();
25         //Perubahan username sudah tersimpan di database.
26
27         $user->wasChanged(); //true (Mengembalikan true karena ada perubahan (username))
28         $user->wasChanged('username'); //true (Mengembalikan true karena username memang berubah)
29         $user->wasChanged(['username', 'level_id']); //true (Mengembalikan true karena username berubah (meskipun level_id tidak berubah, tapi karena sa
30         $user->wasChanged('nama'); //false (Mengembalikan false karena nama tidak berubah)
31         dd($user->wasChanged(['nama', 'username'])); //true (Mengembalikan true karena username berubah (meskipun nama tidak berubah, karena salah satu)
```

4. 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
```

Penjelasan :

- `wasChanged()` digunakan untuk memeriksa apakah ada perubahan setelah `save()`.
- Jika tidak ada perubahan, `wasChanged()` akan mengembalikan `false`.
- Jika ada perubahan di salah satu atribut yang dicek, hasilnya `true`.

Hasil output yang ditampilkan adalah `true`, yang berarti ada perubahan yang berhasil disimpan ke database



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>

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 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>
            <td>ID</td>
            <td>Username</td>
            <td>Nama</td>
            <td>ID Level Pengguna</td>
            <td>Aksi</td>
        </tr>
        @foreach ($data as $d)
        <tr>
            <td>{{ $d->user_id }}</td>
            <td>{{ $d->username }}</td>
            <td>{{ $d->nama }}</td>
            <td>{{ $d->level_id }}</td>
            <td>
                <a href="/user/ubah/{{ $d->user_id }}">Ubah</a> |
                <a href="/user/hapus/{{ $d->user_id }}">Hapus</a>
            </td>
        </tr>
        @endforeach
    </table>
</body>
```

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

```
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
5	manager_dua	Manager 2	2	Ubah Hapus
6	manager22	Manager Dua Dua	2	Ubah Hapus
7	manager33	Manager Tiga Tiga	2	Ubah Hapus
8	manager56	Manager55	2	Ubah Hapus
9	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

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Tambah Data User</title>
5  </head>
6  <body>
7      <h1>Form Tambah Data User</h1>
8      <form method="post" action="/user/tambah_simpan">
9
10         {{ csrf_field() }}
11
12         <label>Username</label>
13         <input type="text" name="username" placeholder="Masukan Username">
14         <br>
15
16         <label>Nama</label>
17         <input type="text" name="nama" placeholder="Masukan Nama">
18         <br>
19
20         <label>Password</label>
21         <input type="password" name="password" placeholder="Masukan Password">
22         <br>
23
24         <label>Level ID</label>
25         <input type="number" name="level_id" placeholder="Masukan ID Level">
26         <br><br>
27
28         <input type="submit" class="btn btn-success" value="Simpan">
29     </form>
30  </body>
31
32  </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

```
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         $user = UserModel::all();
14         return view('user', ['data' => $user]);
15     }
16
17     public function tambah()
18     {
19         return view('user_tambah');
20     }
}
```

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

Username

Nama

Password

Level ID

Penjelasan :

- Saat membuka /user, data user akan diambil dari database lalu ditampilkan dalam tabel (melalui `user.blade.php`).
- Saat membuka /user/tambah, pengguna akan melihat form tambah user (melalui `user_tambah.blade.php`).
- Route /user/tambah akan memanggil `UserController@tambah` untuk menampilkan form tambah user.

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



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>

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

```
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         $user = UserModel::all();
14         return view('user', ['data' => $user]);
15     }
16
17     public function tambah()
18     {
19         return view('user_tambah');
20     }
21
22     public function tambah_simpan(Request $request)
23     {
24         UserModel::create([
25             'username' => $request->username,
26             'nama' => $request->nama,
27             'password' => Hash::make($request->password),
28             'level_id' => $request->level_id
29         ]);
30
31     }
32 }
```

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
5	manager_dua	Manager 2	2	Ubah Hapus
6	manager22	Manager Dua Dua	2	Ubah Hapus
7	manager33	Manager Tiga Tiga	2	Ubah Hapus
8	manager56	Manager55	2	Ubah Hapus
9	manager12	Manager11	2	Ubah Hapus
13	admin2	Aqila	1	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

```
resources > views > user_ubah.blade.php > html
1  <!DOCTYPE html>
2  <html>
3  <head>
4  |   <title>Ubah Data User</title>
5  </head>
6  <body>
7  |   <h1>Form Ubah Data User</h1>
8  |   <a href="#">Kembali</a>
9  |   <br><br>
10 |   <form method="post" action="/user/ubah_simpan/{{ $data->user_id }}">
11 |       {{ csrf_field() }}
12 |       {{ method_field('PUT') }}
13 |
14 |       <label>Username</label>
15 |       <input type="text" name="username" placeholder="Masukan Username" value="{{ $data->username }}">
16 |       <br>
17 |
18 |       <label>Nama</label>
19 |       <input type="text" name="nama" placeholder="Masukan Nama" value="{{ $data->nama }}">
20 |       <br>
21 |
22 |       <label>Password</label>
23 |       <input type="password" name="password" placeholder="Masukan Password" value="{{ $data->password }}">
24 |       <br>
25 |
26 |       <label>Level ID</label>
27 |       <input type="number" name="level_id" placeholder="Masukan ID Level" value="{{ $data->level_id }}">
28 |       <br><br>
29 |
30 |       <input type="submit" class="btn btn-success" value="Ubah">
31 |   </form>
32 | </body>
33 | </html>
```

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

Form Ubah Data User

[Kembali](#)

Username

Nama

Password

Level ID

Penjelasan :

Akan menampilkan form edit

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

Form Ubah Data User

[Kembali](#)

Username

Nama

Password

Level ID

[Ubah](#)

-	-	-	-	-	-
13	admin1	Aqila	1	Ubah	Hapus

Penjelasan :

- Ubah digunakan untuk menampilkan form ubah data
- Ubah_simpan digunakan untuk menyimpan data perubahan

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); // Mencari user berdasarkan ID
    $user->delete(); // Menghapus user dari database

    return redirect('/user'); // Redirect kembali ke halaman daftar 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

➤ Sebelum :

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
5	manager_dua	Manager 2	2	Ubah Hapus
6	manager22	Manager Dua Dua	2	Ubah Hapus
7	manager33	Manager Tiga Tiga	2	Ubah Hapus
8	manager56	Manager55	2	Ubah Hapus
9	manager12	Manager11	2	Ubah Hapus
13	admin1	Aqila	1	Ubah Hapus

➤ Sesudah :

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
5	manager_dua	Manager 2	2	Ubah Hapus
6	manager22	Manager Dua Dua	2	Ubah Hapus
7	manager33	Manager Tiga Tiga	2	Ubah Hapus
8	manager56	Manager55	2	Ubah Hapus
9	manager12	Manager11	2	Ubah Hapus

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 scriptnya menjadi seperti di bawah ini

➤ Menambahkan LevelModel

```
app > Models > LevelModel.php > LevelModel
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6  use Illuminate\Database\Eloquent\Relations\HasMany;
7
8
9
10 class LevelModel extends Model
11 {
12     protected $table = 'm_level';
13     protected $primaryKey = 'level_id';
14     protected $fillable = ['level_kode', 'level_name']; //Foreign key
15
16     public function users(): HasMany
17     {
18         return $this->hasMany(UserModel::class, 'level_id', 'level_id');
19     }
20 }
```

➤ Modifikasi UserModel



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>

```
7 | use Illuminate\Database\Eloquent\Relations\BelongsTo;
8 |
9 | class UserModel extends Model
10| {
11|     use HasFactory;
12|
13|     protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan oleh model ini
14|     protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
15|
16|     /**
17|      * The attributes that are mass assignable.
18|      *
19|      * @var array
20|     */
21|     protected $fillable = ['level_id', 'username', 'nama', 'password'];
22|
23|     public function level(): BelongsTo {
24|         return $this->belongsTo(LevelModel::class, 'level_id', 'level_id');
25|     }
26| }
```

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

Penjelasan :

```
Illuminate\Database\Eloquent\Collection {#318 ▼ // app\Http\Controllers\UserController.php:59
    #items: array:8 [▶]
    #escapeWhenCastingToString: false
}
```

- Query UserModel::with('level')->get(); berhasil mengembalikan sebuah Collection yang berisi 8 item.
- Laravel hanya menampilkan struktur data

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_name }}</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	Staff/Kasir	3	STF	Staff/Kasir	Ubah Hapus
5	manager_dua	Manager 2	2	MNG	Manager	Ubah Hapus
6	manager22	Manager Dua Dua	2	MNG	Manager	Ubah Hapus
7	manager33	Manager Tiga Tiga	2	MNG	Manager	Ubah Hapus
8	manager56	Manager55	2	MNG	Manager	Ubah Hapus
9	manager12	Manager11	2	MNG	Manager	Ubah Hapus

Penjelasan :

Akan menampilkan semua data user dengan menambahkan Kode Level dan nama Level dari foreign key tabel level.

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>

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