



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

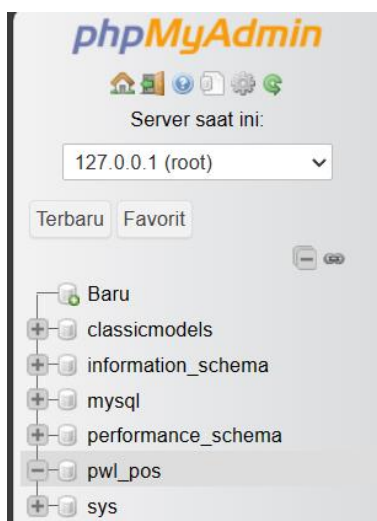
Nama : Muhammad Bagus Indrawan
Kelas : TI-2H
NIM : 2241720217

JOBSHEET 03

MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, dan ELOQUENT ORM

A. PENGATURAN DATABASE

Praktikum ¹ - pengaturan database:



¹ . Buka aplikasi phpMyAdmin, dan buat database baru dengan nama **PWL_POS**



2. Buka aplikasi VSCode dan buka folder project **PWL_POS** yang sudah kita buat
3. Copy file **.env.example** menjadi **.env**
4. Buka file **.env**, dan pastikan konfigurasi **APP_KEY** bernilai. Jika belum bernilai silahkan kalian *generate* menggunakan **php artisan**.

```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=PWL_POS
15 DB_USERNAME=root
16 DB_PASSWORD=
```

5. Edit file **.env** dan sesuaikan dengan database yang telah dibuat

```
.env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:KgPEif3b6D0mqm2FxJey3lHh13EFaseJDuxXn9Af22Y=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=PWL_POS
15 DB_USERNAME=root
16 DB_PASSWORD=
```



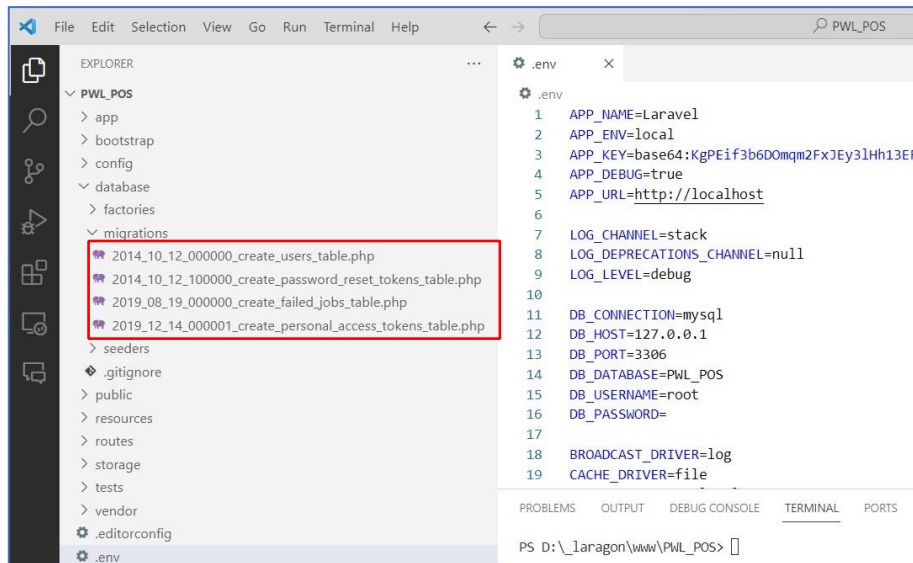
```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:ORfGr1b8vXqPDxmy73/Ib6wnEw6Zda8/uCl8k5PTHs8=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=pwl_pos
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
25 MEMCACHED_HOST=127.0.0.1
26
27 REDIS_HOST=127.0.0.1
28 REDIS_PASSWORD=null
29 REDIS_PORT=6379
30
31 MAIL_MAILER=smtp
32 MAIL_HOST=mailpit
33 MAIL_PORT=1025
34 MAIL_USERNAME=null
35 MAIL_PASSWORD=null
36 MAIL_ENCRYPTION=null
37 MAIL_FROM_ADDRESS="hello@example.com"
```

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

B. MIGRATION

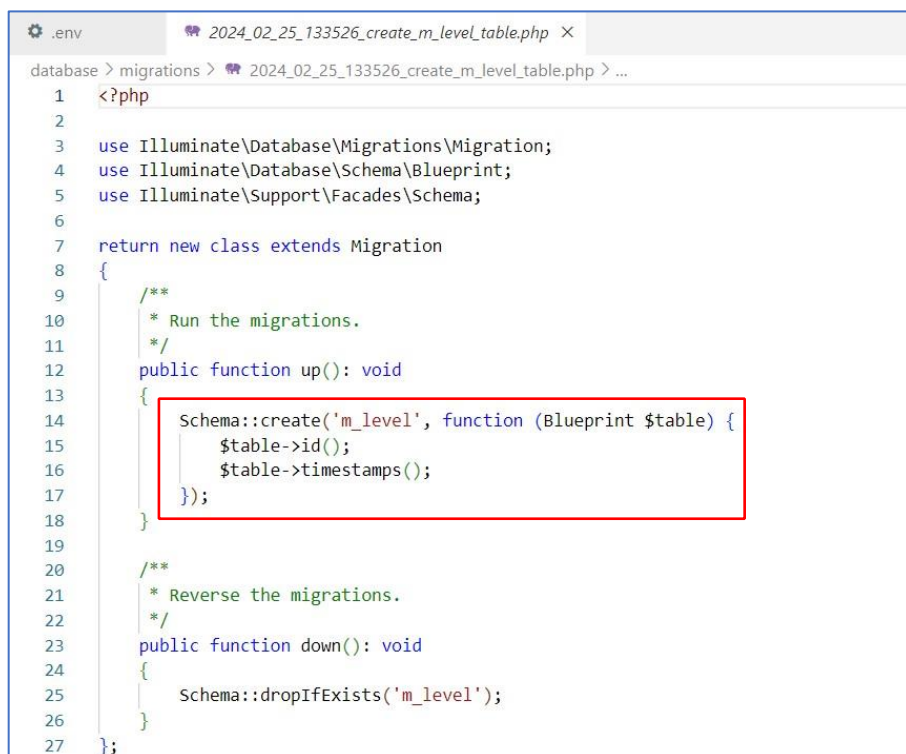
Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

1. Buka *terminal* VSCode kalian, untuk yang di kotak merah adalah default dari laravel



2. Kita abaikan dulu yang di kotak merah (jangan di hapus)
3. Kita buat file migrasi untuk table `m_level` dengan perintah

```
php artisan make:migration create_m_level_table --create=m_level
```





```
PS C:\laragon\www> cd PWL_POS
PS C:\laragon\www\PWL_POS> php artisan make:migration create_m_level_table --create=m_level

INFO Migration [C:\laragon\www\PWL_POS\database\Migrations\2024_03_06_070950_create_m_level_table.php] created successfully.

PS C:\laragon\www\PWL_POS>
```

4. Kita perhatikan bagian yang di kotak merah, bagian tersebut yang akan kita modifikasi sesuai desain database yang sudah ada

```
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('m_level', function (Blueprint $table) {
15             $table->id('level_id');
16             $table->string('level_kode', 10)->unique();
17             $table->string('level_nama', 100);
18             $table->timestamps();
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      */
25     public function down(): void
26     {
27         Schema::dropIfExists('m_level');
28     }
29 };
```

```
PWL_POS > database > migrations > 2024_03_06_070950_create_m_level_table.php
1 Illuminate\Database\Migrations\Migration;
2 use Illuminate\Database\Schema\Blueprint;
3 use Illuminate\Support\Facades\Schema;
4
5 return new class extends Migration
6 {
7     /**
8      * Run the migrations.
9      */
10    public function up(): void
11    {
12        Schema::create('m_level', function (Blueprint $table) {
13            $table->id('level_id');
14            $table->string('level_kode', 10)->unique();
15            $table->string('level_nama', 100);
16            $table->timestamps();
17        });
18    }
19
20    /**
21     * Reverse the migrations.
22     */
23    public function down(): void
24    {
25        Schema::dropIfExists('m_level');
```



5. Simpan kode pada tahapan 4 tersebut, kemudian jalankan perintah ini pada terminal VSCode untuk melakukan migrasi

```
php artisan migrate
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\laragon\www\PWL_POS> php artisan migrate

INFO Preparing database.

Creating migration table 12ms DONE

INFO Running migrations.

2014_10_12_000000_create_users_table 16ms DONE

2014_10_12_100000_create_password_reset_tokens_table 6ms DONE

2019_08_19_000000_create_failed_jobs_table 42ms DONE

2019_12_14_000001_create_personal_access_tokens_table 15ms DONE

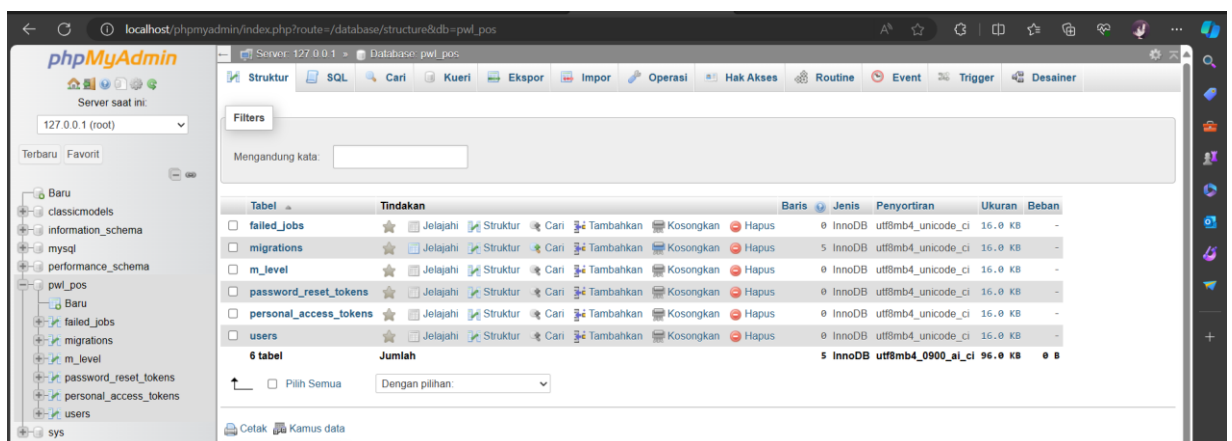
2024_02_25_133526_create_m_level_table 13ms DONE

PS D:\laragon\www\PWL_POS>

6. Kemudian kita cek di phpMyAdmin apakah table sudah ter-generate atau belum

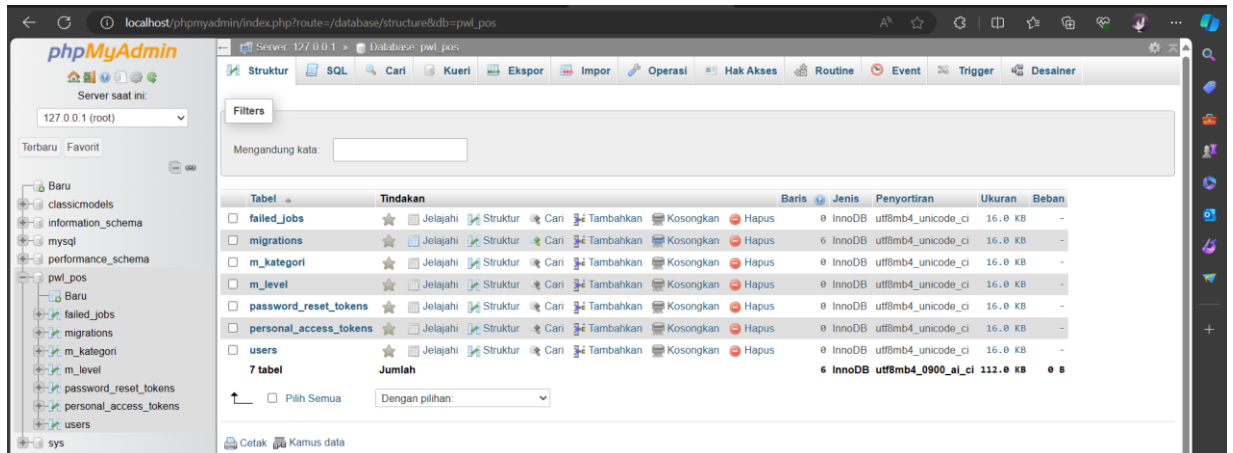
Table	Action	Rows
<input type="checkbox"/> failed_jobs	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	5
<input type="checkbox"/> m_level	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> password_reset_tokens	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> personal_access_tokens	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0

7. Ok, table sudah dibuat di database





8. Buat table *database* dengan *migration* untuk table **m_kategori** yang sama-sama tidak memiliki *foreign key*



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

Praktikum 2.³ - Pembuatan file migrasi dengan relasi

```
php artisan make:migration create_m_user_table --table=m_user
```

```
PS C:\laragon\www\PWL_POS> php artisan make:migration create_m_user_table --table=m_user

INFO Migration [C:\laragon\www\PWL_POS\database\Migrations\2024_03_06_073046_create_m_user_table.php] created successfully.
```

². Buka *terminal* VSCode kalian, dan buat file migrasi untuk table **m_user**

³. Buka file migrasi untuk table **m_user**, dan modifikasi seperti berikut

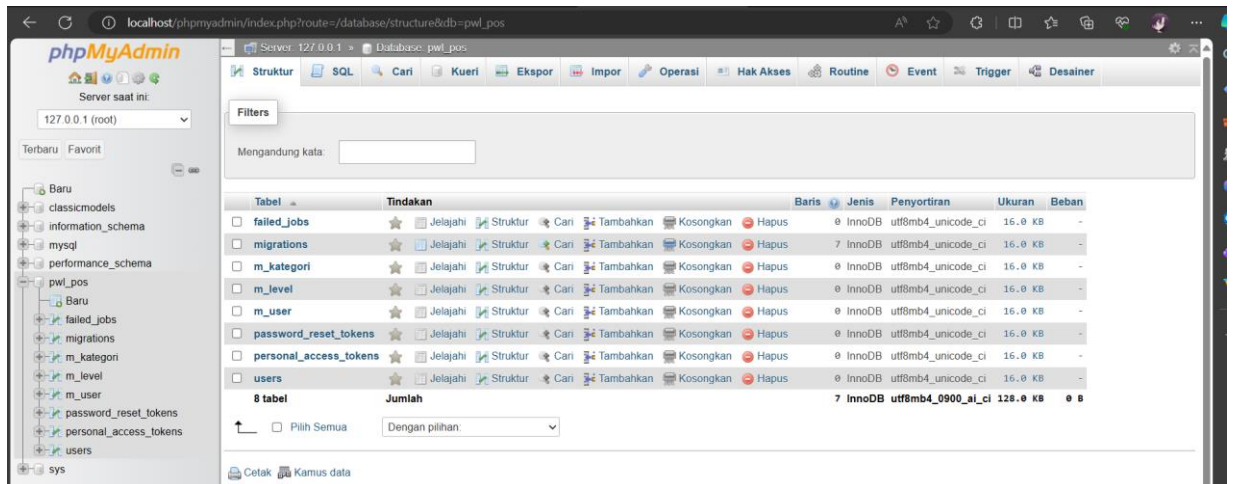


```
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('m_user', function (Blueprint $table) {
15             $table->id('user_id');
16             $table->unsignedBigInteger('level_id')->index(); // indexing untuk ForeignKey
17             $table->string('username', 20)->unique(); // unique untuk memastikan tidak ada username yang sama
18             $table->string('nama', 100);
19             $table->string('password');
20             $table->timestamps();
21
22             // Mendefinisikan Foreign Key pada kolom level_id mengacu pada kolom level_id di tabel m_level
23             $table->foreign('level_id')->references('level_id')->on('m_level');
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      */
30     public function down(): void
31     {
32         Schema::dropIfExists('m_user');
33     }
34 };
```

```
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('m_user', function (Blueprint $table) {
15             $table->id('user_id');
16             $table->unsignedBigInteger('level_id')->index();
17             $table->string('username', 20)->unique();
18             $table->string('nama', 100);
19             $table->string('password');
20             $table->timestamps();
21         });
22     }
23
24     /**
25      * Reverse the migrations.
26      */
27     public function down(): void
28     {
29         Schema::dropIfExists('m_user');
30     }
31 };
```



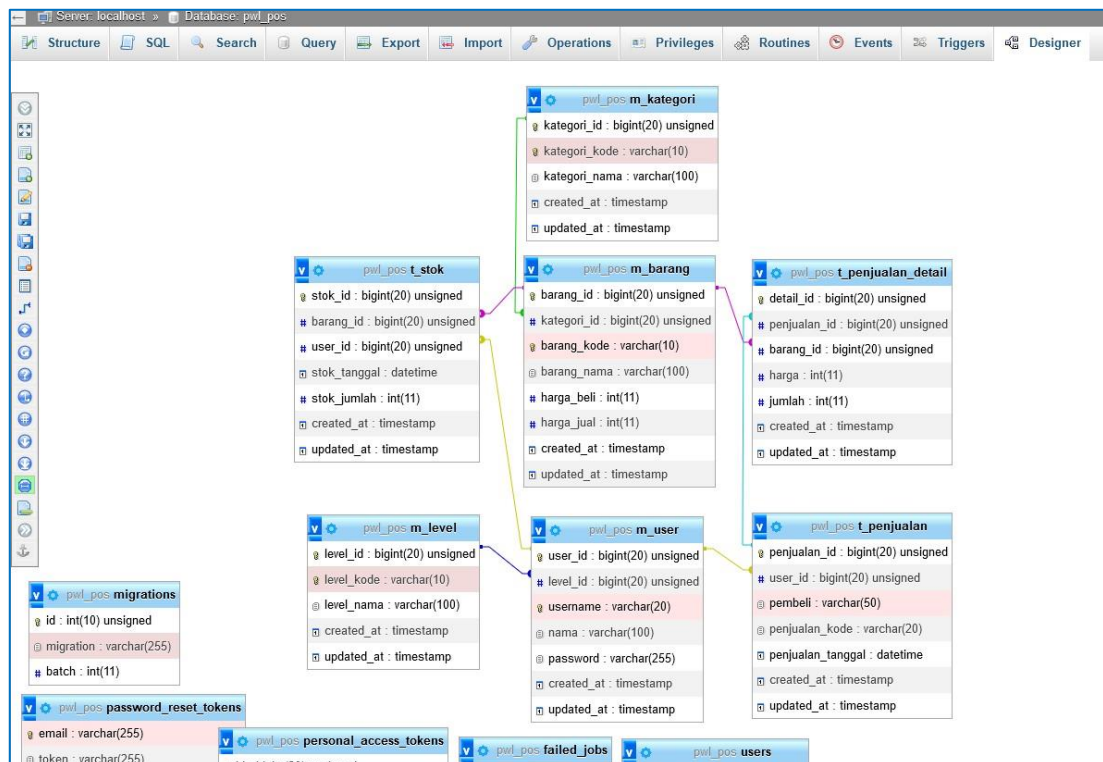

3. Simpan kode program Langkah 2, dan jalankan perintah **php artisan migrate**. Amati apa yang terjadi pada database.

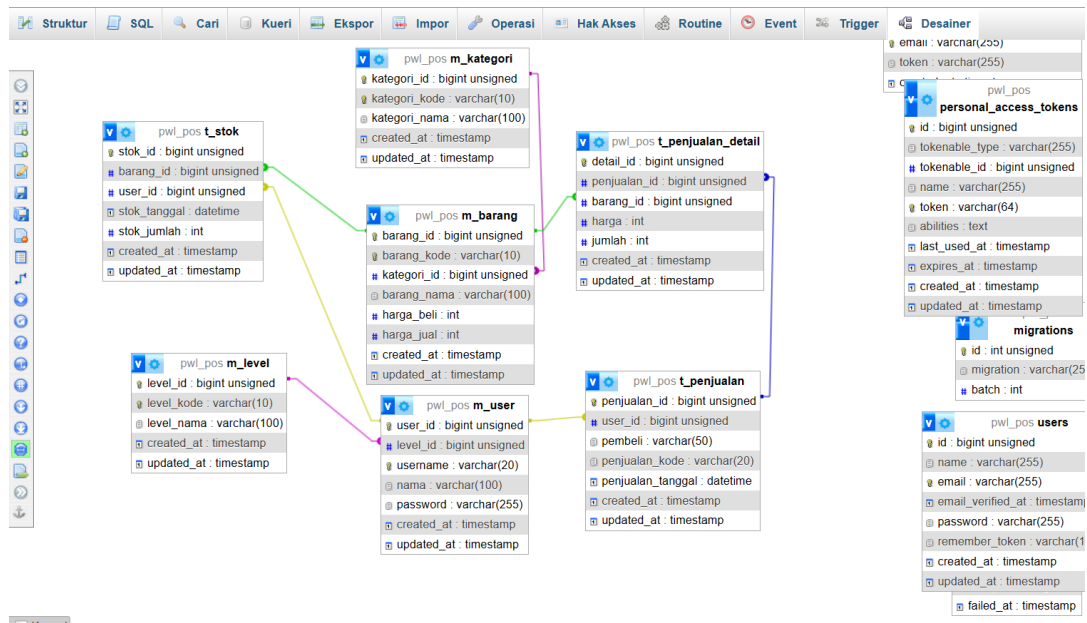


4. Buat table *database* dengan *migration* untuk table-table yang memiliki *foreign key*

m_barang
t_penjualan
t_stok
t_penjualan_detail

5. Jika semua file migrasi sudah di buat dan dijalankan maka bisa kita lihat tampilan *designer* pada **phpMyAdmin** seperti berikut





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

C. SEEDER

Praktikum 3 – Membuat file *seeder*

1. Kita akan membuat file seeder untuk table **m_level** dengan mengetikkan perintah

```
php artisan make:seeder LevelSeeder
```

```
<?php
namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class LevelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        //
    }
}
```

```
PS C:\laragon\www\PWL_POS> php artisan make:seeder LevelSeeder

INFO Seeder [C:\laragon\www\PWL_POS\database\seeders\LevelSeeder.php] created successfully.

PS C:\laragon\www\PWL_POS>
```



2. Selanjutnya, untuk memasukkan data awal, kita modifikasi file tersebut di dalam function `run()`

```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class LevelSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
18             ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
19             ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
20         ];
21         DB::table('m_level')->insert($data);
22     }
23 }
```

3. Selanjutnya, kita jalankan file *seeder* untuk table `m_level` pada terminal

```
php artisan db:seed --class=LevelSeeder
```

```
PS D:\_laragon\www\PWL_POS> php artisan db:seed --class=LevelSeeder
INFO Seeding database.
PS D:\_laragon\www\PWL_POS>
```

4. Ketika *seeder* berhasil dijalankan maka akan tampil data pada table `m_level`

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL
↑ <input type="checkbox"/> Check all With selected: Edit Copy Delete Export					



<div>←T→</div>				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>		Ubah	Salin Hapus	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>		Ubah	Salin Hapus	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>		Ubah	Salin Hapus	3	STF	Staff/Kasir	NULL	NULL

5. Sekarang kita buat file *seeder* untuk table `m_user` yang me-refer ke table `m_level`

```
php artisan make:seeder UserSeeder
```

6. Modifikasi file `class UserSeeder` seperti berikut

```
9 class UserSeeder extends Seeder
10 {
11     public function run(): void
12     {
13         $data = [
14             [
15                 'user_id' => 1,
16                 'level_id' => 1,
17                 'username' => 'admin',
18                 'nama' => 'Administrator',
19                 'password' => Hash::make('12345'), // class untuk mengenkripsi/hash password
20             ],
21             [
22                 'user_id' => 2,
23                 'level_id' => 2,
24                 'username' => 'manager',
25                 'nama' => 'Manager',
26                 'password' => Hash::make('12345'),
27             ],
28             [
29                 'user_id' => 3,
30                 'level_id' => 3,
31                 'username' => 'staff',
32                 'nama' => 'Staff/Kasir',
33                 'password' => Hash::make('12345'),
34             ],
35         ];
36         DB::table('m_user')->insert($data);
37     }
38 }
```



```
php artisan db:seed --class=UserSeeder
```

8. Perhatikan hasil seeder pada table **m_user**

	user_id	level_id	username	nama	password
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$Tevu4dDO1CUAQpeM6H.Vp.LySwY.4oAKU7FzwS6fXV...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$Ajfns20/FdPTeUgghz31muEhIFarULxkh5wvZ9NGRpu...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMY/BHnbJ9W...

☐ Check all With selected: ☐ Edit ☐ Copy ☐ Delete ☐ Export

7. Jalankan perintah untuk mengeksekusi class **UserSeeder**

	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/> Ubah <input type="checkbox"/> Salin <input type="checkbox"/> Hapus	1	1	admin	Administrator	\$2y\$12\$2yeNlg8LTgWPOvKaM42cteBUnGFuQch7oqwsbdSAOF6...	NULL	NULL
<input type="checkbox"/> Ubah <input type="checkbox"/> Salin <input type="checkbox"/> Hapus	2	2	manager	Manager	\$2y\$12\$T/GcmUJtLleZpMcBFDHfLeJhVR.CeWIM2DVC37xM3Sv...	NULL	NULL
<input type="checkbox"/> Ubah <input type="checkbox"/> Salin <input type="checkbox"/> Hapus	3	3	staff	Staff	\$2y\$12\$NxoTQpRAjnfetudu.qOqH.6bJa1aVa3B7qD099QpkL1...	NULL	NULL

9. Ok, data seeder berhasil di masukkan ke database.

10. Sekarang coba kalian masukkan data *seeder* untuk table yang lain, dengan ketentuan seperti berikut

No	Nama Tabel	Jumlah Data	Keterangan
1	m_kategori	5	5 kategori barang
2	m_barang	10	10 barang yang berbeda
3	t_stok	10	Stok untuk 10 barang
4	t_penjualan	10	10 transaksi penjualan
5	t_penjualan_detail	30	3 barang untuk setiap transaksi penjualan

11. Jika sudah, laporkan hasil Praktikum-3 ini dan *commit* perubahan pada *git*

D. DB FACADE

Praktikum 4 – Implementasi DB Facade

1. Kita buat controller dahulu untuk mengelola data pada table **m_level**

```
php artisan make:controller LevelController
```

```
PS C:\laragon\www\PWL_POS> php artisan make:controller LevelController
```

INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\LevelController.php] created successfully.

2. Kita modifikasi dulu untuk *routing*-nya, ada di **PWL_POS/routes/web.php**



```
LevelController.php  web.php X
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\LevelController;
4  use Illuminate\Support\Facades\Route;
5
6
7  Route::get('/', function () {
8      return view('welcome');
9  });
10
11 Route::get('/level', [LevelController::class, 'index']);
```

```
routes > web.php
1  <?php
2
3  use App\Http\Controllers\LevelController;
4  use Illuminate\Support\Facades\Route;
5
6  /*
7  |-----
8  | Web Routes
9  |-----
10 |
11 | Here is where you can register web routes for your application. These
12 | routes are loaded by the RouteServiceProvider and all of them will
13 | be assigned to the "web" middleware group. Make something great!
14 |
15 */
16 |
17 Route::get('/', function () {
18     return view('welcome');
19 });
20
21 Route::get('/level', [LevelController::class, 'index']);
```

3. Selanjutnya, kita modifikasi file `LevelController` untuk menambahkan 1 data ke table `m_level`

```
LevelController.php X  web.php
app > Http > Controllers > LevelController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class LevelController extends Controller
9  {
10     public function index()
11     {
12         DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13
14         return 'Insert data baru berhasil';
15     }
16 }
```




4. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/level dan amati apa yang terjadi pada table `m_level` di database, *screenshot* perubahan yang ada pada table `m_level`

				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	4	CUS	Pelanggan	2024-02-26 08:20:00	NULL

localhost/PWL_POS/public/level x localhost / 127.0.0.1 / pwl_pos / x radeonaru/POS: Answers to Prac x

localhost/PWL_POS/public/level

Insert data baru berhasil

				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Ubah	Salin	Hapus	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	4	CUS	Pelanggan	2024-03-07 14:37:03	NULL

5. Selanjutnya, kita modifikasi lagi file `LevelController` untuk meng-*update* data di table `m_level` seperti berikut

```
LevelController.php x web.php
app > Http > Controllers > LevelController.php > ...
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index()
11     {
12         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13         // return 'Insert data baru berhasil';
14
15         $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
16         return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
17     }
18 }
```



6. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/level lagi dan amati apa yang terjadi pada table `m_level` di database, *screenshot* perubahan yang ada pada table `m_level`

Update data berhasil. Jumlah data yang diupdate: 1 baris

				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Ubah	Salin	Hapus	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	4	CUS	Customer	2024-03-07 14:37:03	NULL

7. Kita coba modifikasi lagi file `LevelController` untuk melakukan proses hapus data

```
LevelController.php | web.php
app > Http > Controllers > LevelController.php > LevelController > index
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class LevelController extends Controller
9  {
10     public function index()
11     {
12         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13         // return 'Insert data baru berhasil';
14
15         // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
16         // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
17
18         $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
19         return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
20     }
21 }
```

Delete data berhasil. Jumlah data yang dihapus: 1 baris



8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table `m_level`. Kita modifikasi file `LevelController` seperti berikut

```
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index()
11     {
12         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13         // return 'Insert data baru berhasil';
14
15         // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
16         // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
17
18         // $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
19         // return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
20
21         $data = DB::select('select * from m_level');
22         return view('level', ['data' => $data]);
23     }
24 }
```

9. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil `view('level')`, maka kita buat file view pada VSCode di

`PWL_POS/resources/view/level.blade.php`

```
resources > views > level.blade.php > ...
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Data Level Pengguna</title>
5     </head>
6     <body>
7         <h1>Data Level Pengguna</h1>
8         <table border="1" cellpadding="2" cellspacing="0">
9             <tr>
10                 <th>ID</th>
11                 <th>Kode Level</th>
12                 <th>Nama Level</th>
13             </tr>
14             @foreach ($data as $d)
15                 <tr>
16                     <td>{{ $d->level_id }}</td>
17                     <td>{{ $d->level_kode }}</td>
18                     <td>{{ $d->level_nama }}</td>
19                 </tr>
20             @endforeach
21         </table>
22     </body>
23 </html>
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi



ID	Kode Level	Nama Level
1	ADM	Administrator
2	MNG	Manager
3	STF	Staff/Kasir

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

E. QUERY BUILDER

Praktikum 5 – Implementasi *Query Builder*

1. Kita buat controller dahulu untuk mengelola data pada table `m_kategori`

```
php artisan make:controller KategoriController
```

```
PS C:\laragon\www\PWL_POS> php artisan make:controller KategoriController
```

```
INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\KategoriController.php] created successfully.
```

2. Kita modifikasi dulu untuk routing-nya, ada di `PWL_POS/routes/web.php`

```
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use Illuminate\Support\Facades\Route;
6
7
8  Route::get('/', function () {
9      return view('welcome');
10 });
11
12 Route::get('/level', [LevelController::class, 'index']);
13 Route::get('/kategori', [KategoriController::class, 'index']);
```

3. Selanjutnya, kita modifikasi file `KategoriController` untuk menambahkan 1 data ke table `m_kategori`



```
LevelController.php KategoriController.php X level.blade.php web.php
app > Http > Controllers > KategoriController.php > KategoriController > index
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController extends Controller
9  {
10     public function index()
11     {
12         $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($data);
18         return 'Insert data baru berhasil';
19     }
20 }
```

4. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/kategori dan amati apa yang terjadi pada table `m_kategori` di database, *screenshot* perubahan yang ada pada table `m_kategori`

localhost/PWL_POS/public/kategori

Insert data baru berhasil

				kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Ubah	Salin	Hapus	1	KTG001	Elektronik	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	2	KTG002	Pakaian	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	3	KTG003	Alat Rumah Tangga	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	4	KTG004	Otomotif	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	5	KTG005	Olahraga	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	6	SNK	Snack\Makanan Ringan	2024-03-07 15:22:20	NULL

5. Selanjutnya, kita modifikasi lagi file `KategoriController` untuk meng-*update* data di table `m_kategori` seperti berikut



```
app > Http > Controllers > KategoriController.php > KategoriController > index
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController extends Controller
9  {
10     public function index()
11     {
12         /* $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($data);
18         return 'Insert data baru berhasil'; */
19
20         $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
21         return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
22     }
23 }
```

6. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/kategori lagi dan amati apa yang terjadi pada table `m_kategori` di database, *screenshot* perubahan yang ada pada table `m_kategori`

localhost/PWL_POS/public/kategori

Update data berhasil. Jumlah data yang diupdate: 1 baris

			kategori_id	kategori_kode	kategori_nama	created_at	updated_at	
<input type="checkbox"/>	Ubah	Salin	Hapus	1	KTG001	Elektronik	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	2	KTG002	Pakaian	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	3	KTG003	Alat Rumah Tangga	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	4	KTG004	Otomotif	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	5	KTG005	Olahraga	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	6	SNK	Camilan	2024-03-07 15:22:20	NULL

7. Kita coba modifikasi lagi file `KategoriController` untuk melakukan proses hapus data



```
10 public function index()
11 {
12     /* $data = [
13         'kategori_kode' => 'SNK',
14         'kategori_nama' => 'Snack/Makanan Ringan',
15         'created_at' => now()
16     ];
17     DB::table('m_kategori')->insert($data);
18     return 'Insert data baru berhasil'; */
19
20     // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
21     // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';
22
23     $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
24     return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row.' baris';
25 }
```

localhost/PWL_POS/public/kateg x localhost / 127.0.0.1 / pwl_pos / x Commits · radeonaru/PWL_POS x

localhost/PWL_POS/public/kategori

Delete data berhasil. Jumlah data yang dihapus: 1baris

				kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Ubah	Salin	Hapus	1	KTG001	Elektronik	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	2	KTG002	Pakaian	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	3	KTG003	Alat Rumah Tangga	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	4	KTG004	Otomotif	NULL	NULL
<input type="checkbox"/>	Ubah	Salin	Hapus	5	KTG005	Olahraga	NULL	NULL

Pilih Semua Dengan pilihan: Ubah Salin Hapus Ekspor

8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table **m_kategori**. Kita modifikasi file **KategoriController** seperti berikut

```
10 public function index()
11 {
12     /* $data = [
13         'kategori_kode' => 'SNK',
14         'kategori_nama' => 'Snack/Makanan Ringan',
15         'created_at' => now()
16     ];
17     DB::table('m_kategori')->insert($data);
18     return 'Insert data baru berhasil'; */
19
20     // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
21     // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';
22
23     // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
24     // return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row.' baris';
25
26     $data = DB::table('m_kategori')->get();
27     return view('kategori', ['data' => $data]);
28 }
```



9. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil `view('kategori')`, maka kita buat file view pada VSCode di `PWL_POS/resources/view/kategori.blade.php`

```
resources > views > kategori.blade.php > html > body > table > tr > td
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Data Kategori Barang</title>
5   </head>
6   <body>
7     <h1>Data Kategori Barang</h1>
8     <table border="1" cellpadding="2" cellspacing="0">
9       <tr>
10        <th>ID</th>
11        <th>Kode Kategori</th>
12        <th>Nama Kategori</th>
13      </tr>
14      @foreach ($data as $d)
15        <tr>
16          <td>{{ $d->kategori_id }}</td>
17          <td>{{ $d->kategori_kode }}</td>
18          <td>{{ $d->kategori_nama }}</td>
19        </tr>
20      @endforeach
21    </table>
22  </body>
23 </html>
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi.

The screenshot shows a web browser window with the URL `localhost/PWL_POS/public/kategori`. The page displays the following table:

ID	Kode Kategori	Nama Kategori
1	KTG001	Elektronik
2	KTG002	Pakaian
3	KTG003	Alat Rumah Tangga
4	KTG004	Otomotif
5	KTG005	Olahraga

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

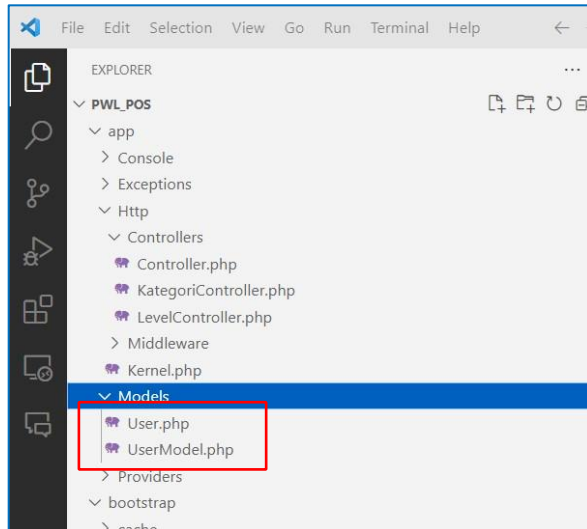
F. ELOQUENT ORM

Praktikum 6 – Implementasi Eloquent ORM

1. Kita buat file model untuk tabel `m_user` dengan mengetikkan perintah



```
php artisan make:model UserModel
```



- Setelah berhasil generate model, terdapat 2 file pada folder `model` yaitu file `User.php` bawaan dari laravel dan file `UserModel.php` yang telah kita buat. Kali ini kita akan menggunakan file `UserModel.php`
- Kita buka file `UserModel.php` dan modifikasi seperti berikut

```
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 oleh model ini
13     protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
14
15 }
```



```
app > Models > UserModel.php > PHP Intelephense > UserModel
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  2 references | 0 implementations
9  class UserModel extends Model
10 |
11 |     use HasFactory;
12 |
13 |     0 references
14 |     protected $table = 'm_user';
15 |     0 references
16 |     protected $primaryKey = 'user_id';
17 |
18 | }
```

4. Kita modifikasi route `web.php` untuk mencoba routing ke controller `UserController`

```
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use App\Http\Controllers\UserController;
6  use Illuminate\Support\Facades\Route;
7
8
9  Route::get('/', function () {
10 |     return view('welcome');
11 | });
12
13 Route::get('/level', [LevelController::class, 'index']);
14 Route::get('/kategori', [KategoriController::class, 'index']);
15 Route::get('/user', [UserController::class, 'index']);
16
```

```
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\LevelController;
4  use App\Http\Controllers\KategoriController;
5  use App\Http\Controllers\UserController;
6  use Illuminate\Support\Facades\Route;
7
8  /*
9  |-----
10 | Web Routes
11 |-----
12 |
13 | Here is where you can register web routes for your application. These
14 | routes are loaded by the RouteServiceProvider and all of them will
15 | be assigned to the "web" middleware group. Make something great!
16 |
17 | */
18
19 Route::get('/', function () {
20 |     return view('welcome');
21 | });
22
23 Route::get('/level', [LevelController::class, 'index']);
24 Route::get('/kategori', [KategoriController::class, 'index']);
25 Route::get('/user', [UserController::class, 'index']);
```

5. Sekarang, kita buat file controller `UserController` dan memodifikasinya seperti berikut



```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7
8  class UserController extends Controller
9  {
10     public function index()
11     {
12         // coba akses model UserModel
13         $user = UserModel::all(); // ambil semua data dari tabel m_user
14         return view('user', ['data' => $user]);
15     }
16 }
```

```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\UserModel;
7
8  2 references | 0 implementations
9  class UserController extends Controller
10 {
11     1 reference | 0 overrides
12     public function index()
13     {
14         $user = UserModel::all();
15         return view('user', ['data' => $user]);
16     }
17 }
```

6. Kemudian kita buat view `user.blade.php`



```
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        <th>ID</th>
11        <th>Username</th>
12        <th>Nama</th>
13        <th>ID Level Pengguna</th>
14      </tr>
15      @foreach ($data as $d)
16        <tr>
17          <td>{{ $d->user_id }}</td>
18          <td>{{ $d->username }}</td>
19          <td>{{ $d->nama }}</td>
20          <td>{{ $d->level_id }}</td>
21        </tr>
22      @endforeach
23    </table>
24  </body>
25 </html>
```

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

7. Jalankan di browser, catat dan laporkan apa yang terjadi



localhost / 127.0.0.1 / pwl_pos / i x | radeonaru (Indra) x

localhost/PWL_POS/public/user

Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff	3

8. Setelah itu, kita modifikasi lagi file `UserController`

```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Hash;
8
9  class UserController extends Controller
10 {
11     public function index()
12     {
13         // tambah data user dengan Eloquent Model
14         $data = [
15             'username' => 'customer-1',
16             'nama' => 'Pelanggan',
17             'password' => Hash::make('12345'),
18             'level_id' => 4
19         ];
20         UserModel::insert($data); // tambahkan data ke tabel m_user
21
22         // coba akses model UserModel
23         $user = UserModel::all(); // ambil semua data dari tabel m_user
24         return view('user', ['data' => $user]);
25     }
26 }
```



```
app > Http > Controllers > UserController.php > PHP Intelephense > UserController > index
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  2 references | 0 implementations
10 class UserController extends Controller
11 {
12     1 reference | 0 overrides
13     public function index()
14     {
15         $data = [
16             'username' => 'customer-1',
17             'nama' => 'Pelanggan',
18             'password' => Hash::make('12345'),
19             'level_id' => 4
20         ];
21         UserModel::insert($data);
22
23         $user = UserModel::all();
24         return view('user', ['data' => $user]);
25     }
26 }
```

9. Jalankan di browser, amati dan laporkan apa yang terjadi

The screenshot shows a web browser window with the address bar displaying `localhost/PWL_POS/public/user`. The page title is "Data User". Below the title is a table with the following data:

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff	3
6	customer-1	Pelanggan	4

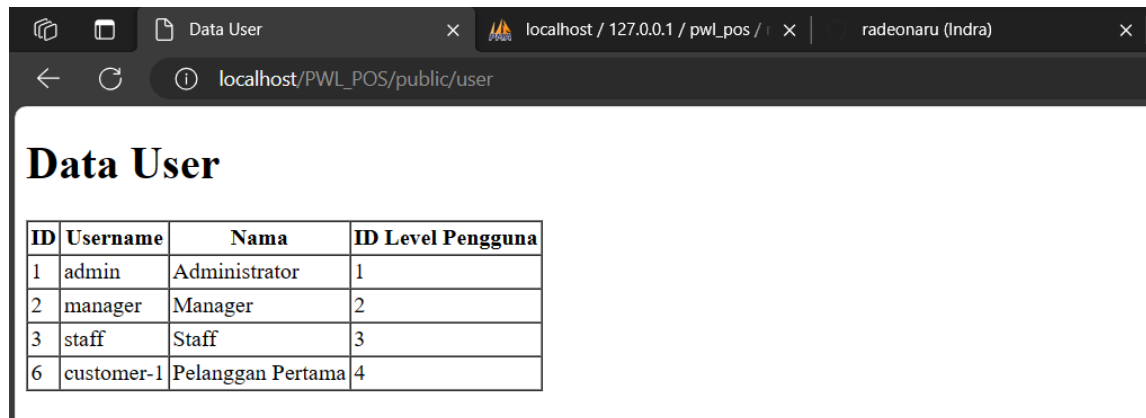
10. Kita modifikasi lagi file `UserController` menjadi seperti berikut



```
9 class UserController extends Controller
10 {
11     public function index()
12     {
13         // tambah data user dengan Eloquent Model
14         $data = [
15             'nama' => 'Pelanggan Pertama',
16         ];
17         UserModel::where('username', 'customer-1')->update($data); // update data user
18
19         // coba akses model UserModel
20         $user = UserModel::all(); // ambil semua data dari tabel m_user
21         return view('user', ['data' => $user]);
22     }
23 }
```

```
app > Http > Controllers > UserController.php > PHP Intelephense > UserController > index
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 2 references | 0 implementations
10 class UserController extends Controller
11 {
12     1 reference | 0 overrides
13     public function index()
14     {
15         // $data = [
16         //     'username' => 'customer-1',
17         //     'nama' => 'Pelanggan',
18         //     'password' => Hash::make('12345'),
19         //     'level_id' => 4
20         // ];
21         // UserModel::insert($data);
22
23         $data = [
24             'nama' => 'Pelanggan Pertama',
25         ];
26         UserModel::where('username', 'customer-1')->update($data);
27
28         $user = UserModel::all();
29         return view('user', ['data' => $user]);
30     }
31 }
```

11. Jalankan di browser, amati dan laporkan apa yang terjadi



ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff	3
6	customer-1	Pelanggan Pertama	4

12. Jika sudah, laporkan hasil Praktikum-6 ini dan *commit* perubahan pada *git*

G. Penutup

Jawablah pertanyaan berikut sesuai pemahaman materi di atas

1. Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari `APP_KEY` pada *file setting .env* Laravel?

Fungsi dari `APP_KEY` adalah untuk mengamankan sesi pengguna dan data yang dienkripsi oleh aplikasi Laravel yang digunakan untuk enkripsi dan dekripsi data sensitif seperti cookie dan data sesi.

2. Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk `APP_KEY`?

- 1) Buka terminal atau command prompt.
- 2) Arahkan terminal ke direktori proyek Laravel Anda.
- 3) Jalankan perintah: `php artisan key:generate`

3. Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?

Secara default, Laravel memiliki dua file migrasi yang terletak di dalam direktori `database/migrations`:

- 1) `2014_10_12_000000_create_users_table.php`
- 2) `2014_10_12_100000_create_password_resets_table.php`

File migrasi tersebut memiliki tujuan dan fungsi yang berbeda:

- 1) `2014_10_12_000000_create_users_table.php`: File migrasi ini bertanggung jawab untuk membuat tabel pengguna (`users`) dalam database Anda.
- 2) `2014_10_12_100000_create_password_resets_table.php`: File migrasi ini bertanggung jawab untuk membuat tabel reset password (`password_resets`).

4. Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/*output* dari fungsi tersebut?



Kode `$table->timestamps()`; dalam file migrasi Laravel memiliki tujuan untuk secara otomatis menambahkan dua kolom ke tabel yang dibuat, yaitu `created_at` dan `updated_at`.

Kolom `created_at` akan menampung tanggal dan waktu ketika sebuah rekaman dibuat pertama kali, sedangkan kolom `updated_at` akan menampung tanggal dan waktu ketika rekaman tersebut terakhir kali diubah.

5. Pada File Migrasi, terdapat fungsi `$table->id()`; Tipe data apa yang dihasilkan dari fungsi tersebut?

Fungsi `$table->id()`; dalam file migrasi Laravel menghasilkan kolom dengan tipe data `BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY`.

6. Apa bedanya hasil migrasi pada table `m_level`, antara menggunakan `$table->id()`; dengan menggunakan `$table->id('level_id')` ?

Apabila migrasi menggunakan `$table->id()`;; hasilnya akan membuat kolom dengan nama default `id`., dan apabila migrasi menggunakan `$table->id('level_id')`;; hasilnya akan membuat kolom dengan nama khusus `level_id`.

7. Pada migration, Fungsi `->unique()` digunakan untuk apa?

Fungsi `->unique()` digunakan untuk menetapkan kolom sebagai unik, yang berarti nilai-nilai di kolom tersebut harus unik atau tidak dapat memiliki duplikat nilai di setiap baris dalam tabel.

8. Pada **Praktikum 2.2 - Tahap 2**, kenapa kolom `level_id` pada tabel `m_user` menggunakan `$table->unsignedBigInteger('level_id')`, sedangkan kolom `level_id` pada tabel `m_level` menggunakan `$table->id('level_id')` ?

`$table->id('level_id')` pada tabel `m_level` sebenarnya menggunakan metode `id()` yang disediakan oleh Laravel untuk secara otomatis membuat kolom `id` sebagai primary key dan secara otomatis bertambah (`auto-increment`). Sedangkan `$table->unsignedBigInteger('level_id')` pada tabel `m_user` mendefinisikan kolom `level_id`



sebagai bilangan bulat besar tak bertanda (unsigned big integer) yang nantinya akan dijadikan sebagai foreign key dari tabel `m_level`.

9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');`?

Tujuan dari class `Hash` adalah untuk memudahkan penggunaan hashing, khususnya dalam hal mengenkripsi dan mendekripsi data.

`Hash::make('1234')` adalah metode yang digunakan untuk menghasilkan hash dari string tertentu. Dalam kode program diatas, string yang diberikan adalah password '1234'. Fungsi ini akan mengambil string tersebut dan menghasilkan hash unik berdasarkan algoritma hashing yang ditentukan di konfigurasi Laravel. Hasilnya akan menjadi string hash yang tidak bisa dibalik menjadi nilai aslinya dengan mudah.

10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?

Tanda tanya ? pada query builder digunakan sebagai parameter placeholder dalam query SQL. Parameter ini adalah cara aman untuk memasukkan nilai ke dalam query tanpa harus menyertakan nilai secara langsung di dalam string query.

11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode `protected $table =`

`'m_user';` dan `protected $primaryKey = 'user_id';` ?

Penulisan kode `protected` pada `protected $table = 'm_user';` dan `protected $primaryKey = 'user_id';` adalah untuk mendefinisikan properti-properti tersebut sebagai akses proteksi dalam penggunaannya di dalam `UserModel`.

12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Façade / Query Builder / Eloquent ORM*) ? jelaskan

Menurut saya, saya lebih mudah menggunakan Query Builder, karena saya lebih bisa memahami sintaks PHP yang digunakan pada PHP daripada operasi yang lain. Selain itu, penulisan query pada Query Builder menggunakan metode chaining PHP. Yang dimana metode ini memberikan tingkat kontrol yang lebih langsung terhadap query yang dihasilkan, terutama untuk operasi yang lebih kompleks atau kustom.