

Laporan Praktikum Pemrograman Web Lanjut
Jobsheet 3 : MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, dan
ELOQUENT ORM



Oleh :
Danica Nasywa Putriniair (2341760122)
Kelas SIB 2B / 05

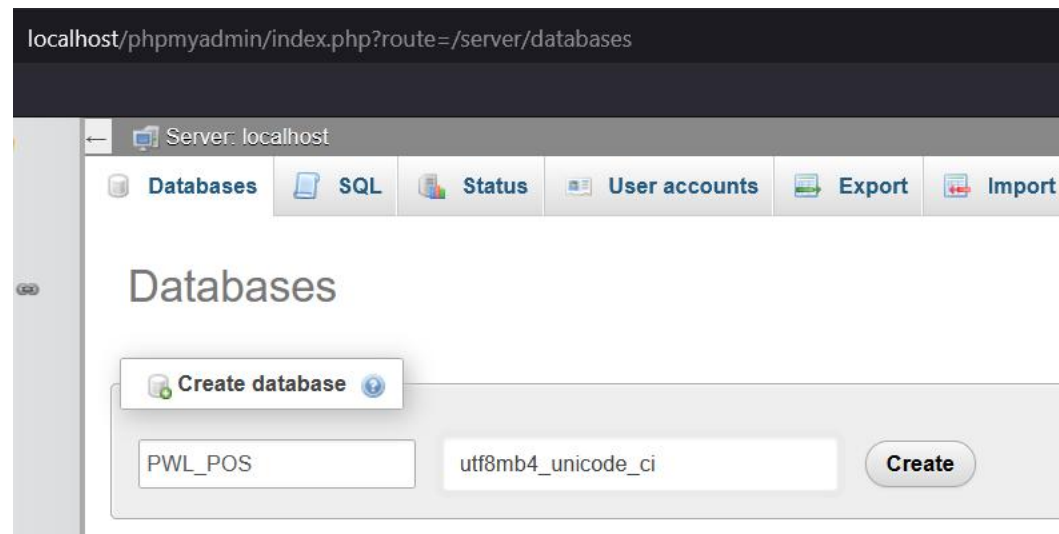
PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS JURUSAN
TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

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

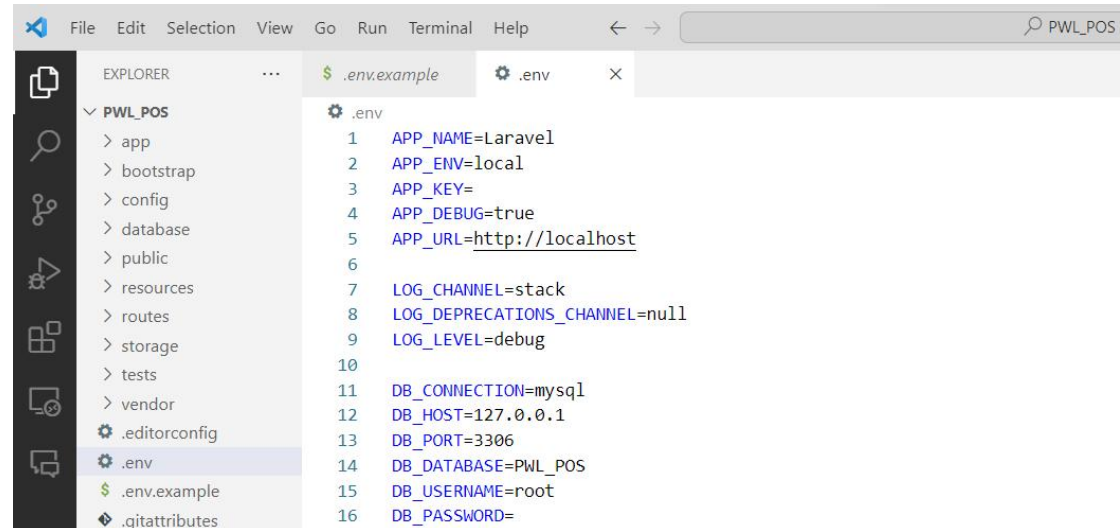
A. Pengaturan Database

Praktikum 1 - Pengaturan Database

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



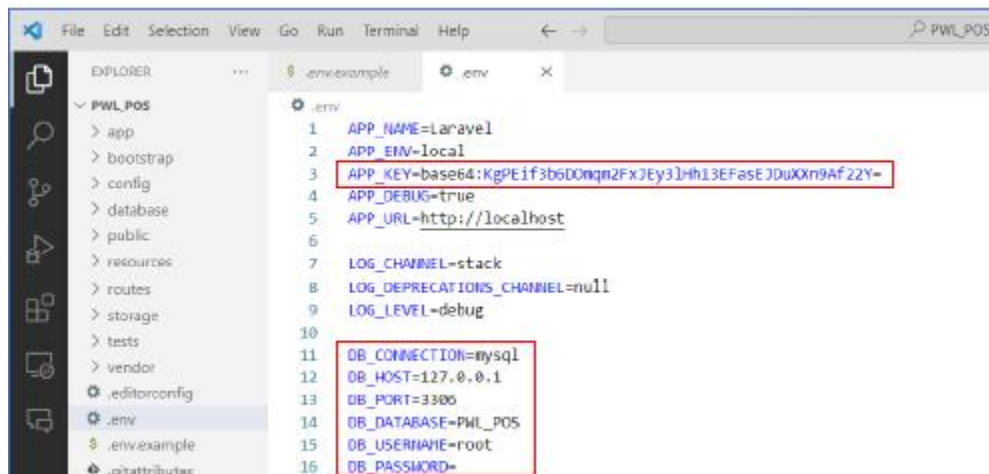
2. Buka aplikasi VSCode dan buka folder project PWL_POS yang sudah kita buat
3. Copy file .env.example menjadi .env
4. Buka file .env dan pastikan konfigurasi APP_KEY bernilai. Jika belum bernilai silahkan kalian *generate* menggunakan php artisan



Hasil :

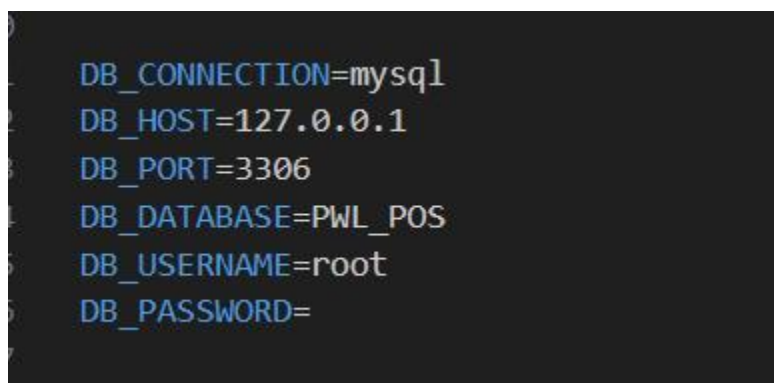


5. Edit file .env dan sesuaikan dengan database yang telah dibuat



```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:KgPEif3b6D0nqn2Fx7Ey3lHh13EFasE7DuXXn9Af22Y=
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=
```

Hasil :



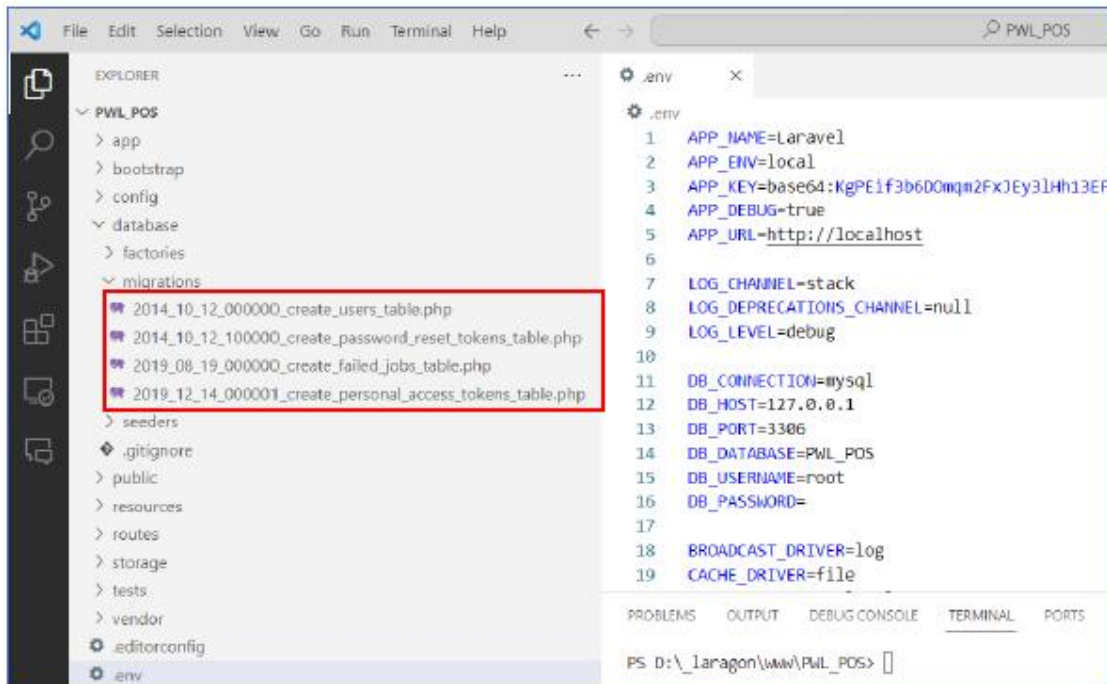
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=PWL_POS
DB_USERNAME=root
DB_PASSWORD=
```

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

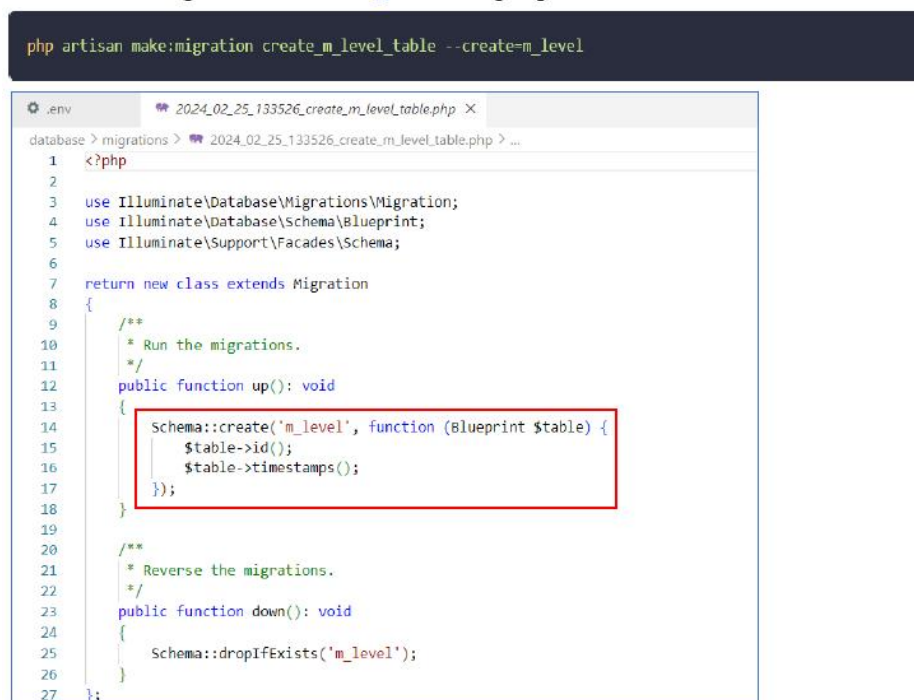
B. Migration

Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

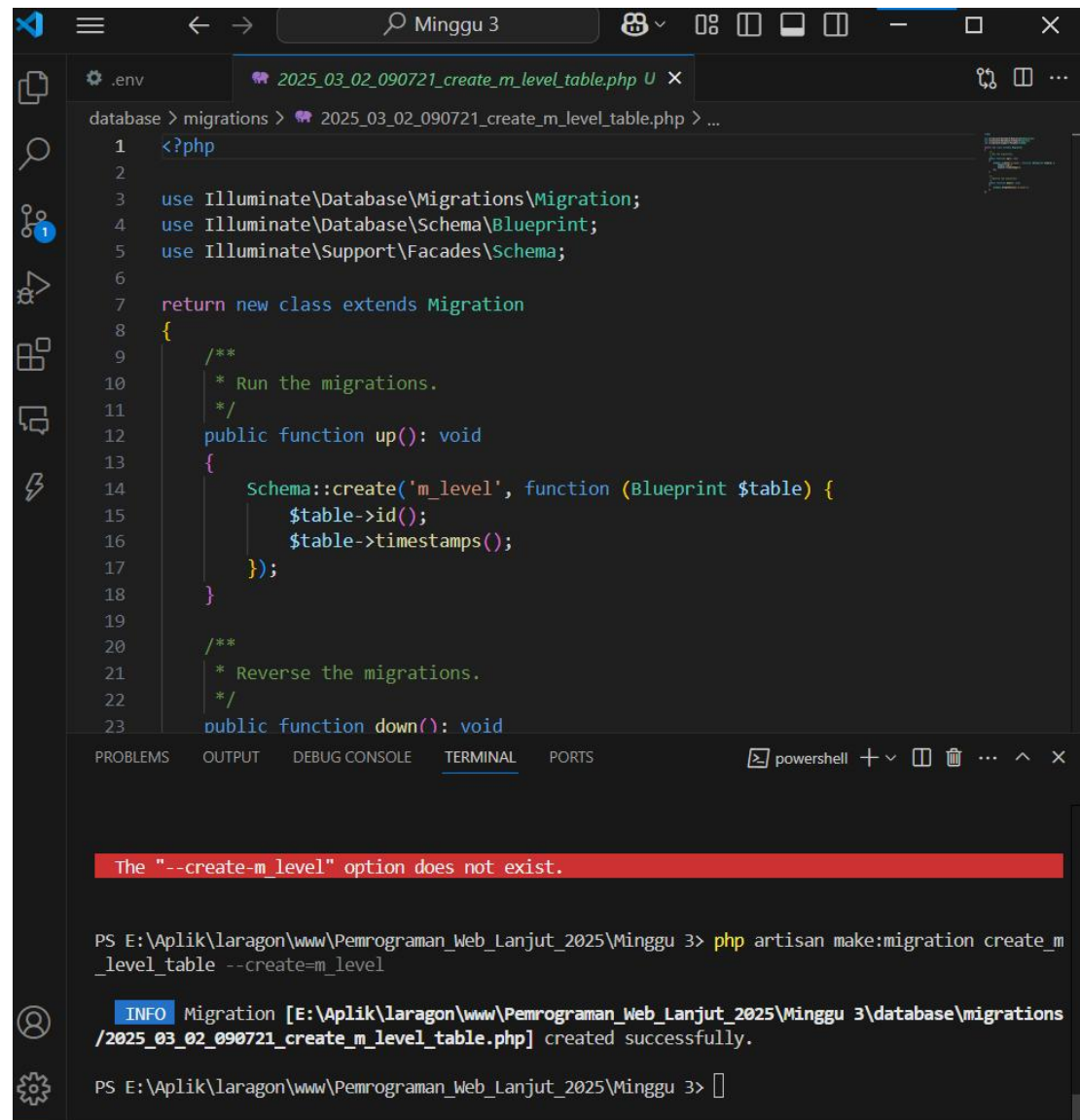
1. Buka *terminal* VSCode kalian, untuk yang dikotak merah adalah default dari laravel



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



Hasil :



The screenshot shows a Visual Studio Code editor window with a file named `2025_03_02_090721_create_m_level_table.php` open. The file contains a PHP migration class that extends `Illuminate\Database\Migrations\Migration`. The `up()` method uses `Schema::create` to create a table named `m_level` with columns `id` and `timestamps`. The `down()` method is also present but empty.

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
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();
16             $table->timestamps();
17         });
18     }
19
20     /**
21      * Reverse the migrations.
22      */
23     public function down(): void
```

Below the editor, the terminal shows the command `php artisan make:migration create_m_level_table --create=m_level` being executed. A red error message is displayed: "The "--create=m_level" option does not exist." Below this, an information message states: "Migration [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 3\database\Migrations\2025_03_02_090721_create_m_level_table.php] created successfully."

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 };

```

Hasil :

```

2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
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();
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 };

```

5. Simpan kode pada tahap 4 tersebut, kemudian jalankan perintah ini pada

terminal VSCode untuk melakukan migrasi

```
php artisan migrate
```

```
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>
```

Hasil :

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 3> php artisan migrate
```

```
INFO: Preparing database.
```

```
Creating migration table ..... 3,372ms DONE
```

```
INFO: Running migrations.
```

```
2014_10_12_000000_create_users_table ..... 699ms DONE
```

```
2014_10_12_100000_create_password_reset_tokens_table ..... 147ms DONE
```

```
2019_08_19_000000_create_failed_jobs_table ..... 233ms DONE
```

```
2019_12_14_000001_create_personal_access_tokens_table ..... 389ms DONE
```

```
2025_03_02_090721_create_m_level_table ..... 244ms DONE
```

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

Hasil :

Table	Action
<input type="checkbox"/> failed_jobs	★ Browse Structure Search Insert Empty
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty
<input type="checkbox"/> m_level	★ Browse Structure Search Insert Empty
<input type="checkbox"/> password_reset_tokens	★ Browse Structure Search Insert Empty
<input type="checkbox"/> personal_access_tokens	★ Browse Structure Search Insert Empty
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty
6 tables	Sum

7. Ok, table sudah dibuat di database

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

Hasil :

- Membuat migrasi untuk table *m_kategori*

```

2025_03_02_090721_create_m_level_table.php U  2025_03_02_151816_create_m_kategori_table.php U x
database > migrations > 2025_03_02_151816_create_m_kategori_table.php > ...
1  k?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create('m_kategori', function (Blueprint $table) {
15             $table->id();
16             $table->timestamps();
17         });
18     }
19
20     /**
21      * Reverse the migrations.
22      */
23     public function down(): void
24     {
25         Schema::dropIfExists('m_kategori');
26     }
27 }

```

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
2014_10_12_100000_create_password_reset_tokens_table ..... 147ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 233ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 389ms DONE
2025_03_02_090721_create_m_level_table ..... 244ms DONE

PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 3> php artisan make:migration create_m_kategori_table --create=m_kategori

INFO Migration [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 3\database\migrations\2025_03_02_151816_create_m_kategori_table.php] created successfully.

```

- Memodifikasi sesuai dengan database (*kategori_id*, *kategori_kode* String 10,

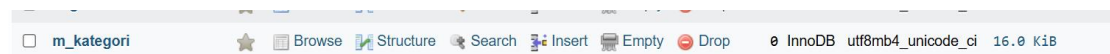
kategori_nama String 10)

```
*/  
public function up(): void  
{  
    Schema::create('m_kategori', function (Blueprint $table) {  
        $table->id();  
        $table->string('kategori_kode', 10)->unique();  
        $table->string('kategori_nama', 10);  
        $table->timestamps();  
    });  
}
```

- Lalu jalankan perintah php artisan migrate

```
php artisan migrate  
  
INFO Running migrations.  
2025_03_02_151816_create_m_kategori_table ..... 664ms DONE
```

- Lalu cek di phpMyAdmin



The screenshot shows the phpMyAdmin interface with the 'm_kategori' table selected. The table structure is displayed, showing columns for 'id', 'kategori_kode', 'kategori_nama', and 'timestamps'.

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

Praktikum 2.2 - Pembuatan file migrasi dengan relasi

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

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

Hasil :

```
PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 3> php artisan make:migration create_m_user_table --table=m_user  
  
INFO Migration [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 3\database\Migrations\2025_03_02_152404_create_m_user_table.php] created successfully.
```

2. 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 };

```

Hasil :

```

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('m_user', function (Blueprint $table) {
            $table->id('user_id'); // Membuat kolom user_id sebagai primary key
            $table->unsignedBigInteger('level_id')->index(); // Kolom level_id
            $table->string('username', 20)->unique(); // Kolom username yang unik
            $table->string('nama', 100); // Kolom nama
            $table->string('password'); // Kolom password
            $table->timestamps(); // Kolom created_at dan updated_at

            // Menambahkan foreign key constraint
            $table->foreign('level_id')->references('id')->on('m_level')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('m_user'); // Menghapus tabel m_user jika migrasi dibatalkan
    }
}

```

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

Hasil :

☐ m_kategori

☐ m_level

☐ m_user

pwl_pos m_level	
id	: bigint unsigned
level_kode	: varchar(10)
level_nama	: varchar(100)
created_at	: timestamp
updated_at	: timestamp

pwl_pos m_user	
user_id	: bigint unsigned
level_id	: bigint unsigned
username	: varchar(20)
nama	: varchar(100)
password	: varchar(255)
created_at	: timestamp
updated_at	: timestamp

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

m_barang
t_penjualan
t_stok
t_penjualan_detail

a. m_barang

```
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('m_barang', function (Blueprint $table) {
            $table->id('barang_id'); // Membuat kolom barang_id sebagai primary key
            $table->unsignedBigInteger('kategori_id')->index(); // Kolom kategori_id
            $table->string('barang_kode', 10)->unique(); // Kolom barang_kode yang unik
            $table->string('barang_nama', 100); // Kolom barang_nama
            $table->integer('harga_beli'); // Kolom harga beli
            $table->integer('harga_jual'); // Kolom harga jual

            $table->timestamps(); // Kolom created_at dan updated_at

            // Menambahkan foreign key constraint
            $table->foreign('kategori_id')->references('id')->on('m_kategori')->onDelete('cascade');
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('m_barang');
    }
};
```

☐ m_barang

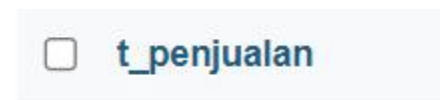
☐ m_kategori

b. t_penjualan

```
public function up(): void
{
    Schema::create('t_penjualan', function (Blueprint $table) {
        $table->id('penjualan_id'); // Membuat kolom barang_id sebagai primary key
        $table->unsignedBigInteger('user_id')->index(); // Kolom kategori_id
        $table->string('penjualan_kode', 50)->unique(); // Kolom barang_kode yang unik
        $table->dateTime('penjualan_tanggal');
        $table->integer('harga_jual'); // Kolom harga jual
        $table->timestamps(); // Kolom created_at dan updated_at

        // Menambahkan foreign key constraint
        $table->foreign('user_id')->references('user_id')->on('m_user')->onDelete('cascade');
    });

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('t_penjualan');
    }
}
```



c. t_stok

```
Schema::create('t_stok', function (Blueprint $table) {
    $table->id('stok_id'); // Membuat kolom barang_id sebagai primary key
    $table->unsignedBigInteger('barang_id')->index(); // Kolom kategori_id
    $table->unsignedBigInteger('user_id')->index(); // Kolom kategori_id
    $table->dateTime('stok_tanggal');
    $table->integer('stok_jual'); // Kolom harga jual
    $table->timestamps(); // Kolom created_at dan updated_at

    // Menambahkan foreign key constraint
    $table->foreign('user_id')->references('user_id')->on('m_user')->onDelete('cascade');
    $table->foreign('barang_id')->references('barang_id')->on('m_barang')->onDelete('cascade');
});
```



d. t_penjualan_detail

```
Schema::create('t_penjualan_detail', function (Blueprint $table) {
    $table->id('stok_id'); // Membuat kolom barang_id sebagai primary key
    $table->unsignedBigInteger('barang_id')->index(); // Kolom kategori_id
    $table->unsignedBigInteger('penjualan_id')->index(); // Kolom kategori_id
    $table->integer('harga'); // Kolom harga jual
    $table->integer('jumlah'); // Kolom harga jual
    $table->timestamps(); // Kolom created_at dan updated_at

    // Menambahkan foreign key constraint
    $table->foreign('barang_id')->references('barang_id')->on('m_barang')->onDelete('cascade');
    $table->foreign('penjualan_id')->references('penjualan_id')->on('t_penjualan')->onDelete('cascade');
});
```



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
```

Hasil :

```
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7
8 class LevelSeeder extends Seeder
9 {
10     /**
11      * Run the database seeds.
12      */
13     public function run(): void
14     {
15         //
16     }
17 }
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
↓ make:scope
↓ make:seeder
↓ make:test
↓ make:view

PS E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 3> php artisan make:seeder LevelSeeder

INFO Seeder [E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 3\database\seeders\LevelSeeder.php] created successfully.
```

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 }
```

```
11 /**
12  * Run the database seeds.
13  */
14 public function run(): void
15 {
16     $data = [
17         ['level_id' => 1, 'level_kode'=> 'DM', 'level_nama' => 'Administrator'],
18         ['level_id' => 2, 'level_kode'=> 'MNG', 'level_nama' => 'Manager'],
19         ['level_id' => 3, 'level_kode'=> 'STF', 'level_nama' => 'Staff/Kasir'],
20     ];
21     DB::table('m_level')->insert($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>
```

Hasil :

```
S E:\Aplik\laragon\www\Pemrograman_Web_Lanjut_2025\Minggu 3>
php artisan db:seed --class=LevelSeeder

INFO Seeding database.
```

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						

Hasil :

		level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	DM	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						

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 }
```

7. Jalankan perintah untuk mengeksekusi class UserSeeder

```
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 Copy Delete	1	1	admin	Administrator	\$2y\$12\$Tevu4dDO1CUAQpeM6H.Vp.LySwY.4oAKU7FzwS6fXV...
<input type="checkbox"/>	Edit Copy Delete	2	2	manager	Manager	\$2y\$12\$A/fns20/FdPTeUgghz31muEhIFaruLxkh5wvZ9NGRpu...
<input type="checkbox"/>	Edit Copy Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMY/BHnbJ9W...
<input type="checkbox"/> Check all With selected: Edit Copy Delete Export						

Hasil :

<div><div><div>⏪</div><div>⏩</div></div></div>									user_id	level_id	username	nama	password
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	1	admin	Administrator	\$2y\$12\$HvF7mT4UmPSrEC3cVHnhCuO9GSIQ1bcYClqU1j6xyg...					
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	2	manager	Manager	\$2y\$12\$CgLFsNsOkjGwrG9Pse1ig.rPMGyl5npJlToQxG6qqmM...					
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	3	staff	Staff/Kasir	\$2y\$12\$26NCxawX6bN1TwaPxG8goeilWp77Z3yJL90vsSxNC9t...					

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

a. KategoriSeeder

<div><div><div><div></div><div></div><div></div></div></div><div></div></div>				kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<div><div><div></div></div></div>	<div><div><div></div></div></div> Edit	<div><div><div></div></div></div> Copy	<div><div><div></div></div></div> Delete	1	PKN	Pakaian	NULL	NULL
<div><div><div></div></div></div>	<div><div><div></div></div></div> Edit	<div><div><div></div></div></div> Copy	<div><div><div></div></div></div> Delete	2	MKN	Makanan	NULL	NULL
<div><div><div></div></div></div>	<div><div><div></div></div></div> Edit	<div><div><div></div></div></div> Copy	<div><div><div></div></div></div> Delete	3	MNM	Minuman	NULL	NULL
<div><div><div></div></div></div>	<div><div><div></div></div></div> Edit	<div><div><div></div></div></div> Copy	<div><div><div></div></div></div> Delete	4	ATK	AlTuKan	NULL	NULL
<div><div><div></div></div></div>	<div><div><div></div></div></div> Edit	<div><div><div></div></div></div> Copy	<div><div><div></div></div></div> Delete	5	ELT	Elektronik	NULL	NULL

b. BarangSeeder

		barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	3	1	HDP	Headphone	500000	700000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	2	KTS	Kaos	70000	100000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	2	CLJ	Celana Jeans	150000	200000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	3	RTI	Roti	10000	15000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	3	NSB	Nasi Bungkus	20000	25000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	4	AMN	Air Mineral	5000	8000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	4	SUS	Susu	12000	17000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	5	BKT	Buku Tulis	15000	20000	NULL	NULL

c. StokSeeder

		stok_id	barang_id	user_id	stok_tanggal	stok_jual	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	3	3	1	2025-02-02 00:00:00	20	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	4	1	2025-02-02 00:00:00	100	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	5	1	2025-02-02 00:00:00	80	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	6	1	2025-02-02 00:00:00	200	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	7	1	2025-02-02 00:00:00	150	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	8	1	2025-02-02 00:00:00	300	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	9	1	2025-02-02 00:00:00	120	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	10	1	2025-02-02 00:00:00	90	NULL	NULL

d. PenjualanSeeder

```

class PenjualanSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $penjualan = [];
        for ($i = 1; $i <= 10; $i++) {
            $penjualan[] = [
                'user_id' => 1, // Sesuaikan dengan user yang ada
                'pembeli' => 'Pelanggan ' . $i,
                'penjualan_kode' => 'PJI00' . $i,
                'penjualan_tanggal' => now(),
                'created_at' => now(),
                'updated_at' => now(),
            ];
        }

        DB::table('t_penjualan')->insert($penjualan);
    }
}

```

e. PenjualanDetailSeeder


```

class PenjualanDetailSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $penjualan_detail = [];
        for ($i = 1; $i <= 10; $i++) {
            for ($j = 1; $j <= 3; $j++) { // 3 barang per transaksi
                $penjualan_detail[] = [
                    'penjualan_id' => $i,
                    'barang_id' => rand(1, 10),
                    'harga' => rand(1100, 12000),
                    'jumlah' => rand(1, 5),
                    'created_at' => now(),
                    'updated_at' => now(),
                ];
            }
        }

        DB::table('t_penjualan_detail')->insert($penjualan_detail);
    }
}

```

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
```

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']);
```

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

```
LevelController.php  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 }
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

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

```
LevelController.php x 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 }
```

8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table

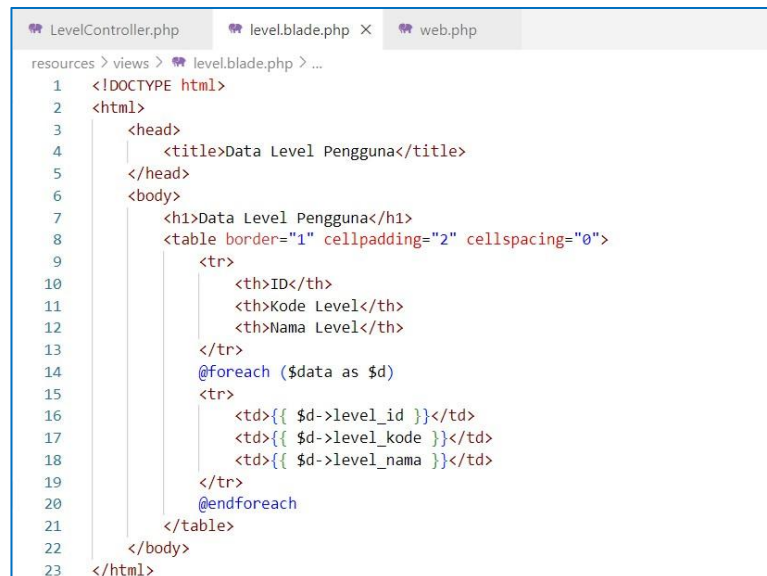
`m_level`. Kita modifikasi file `LevelController` seperti berikut

```

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

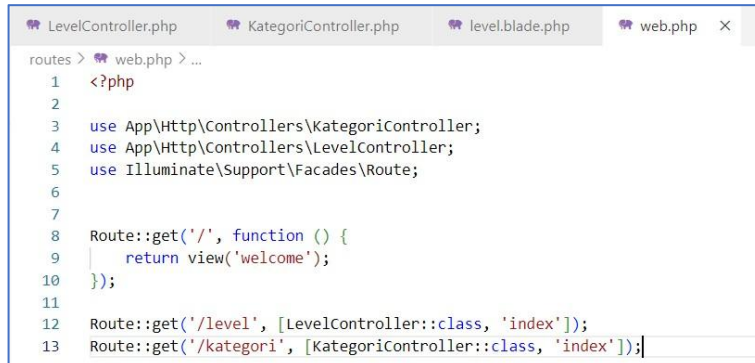
E. QUERY BUILDER

Praktikum 5 - Implementasi Query Builder

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

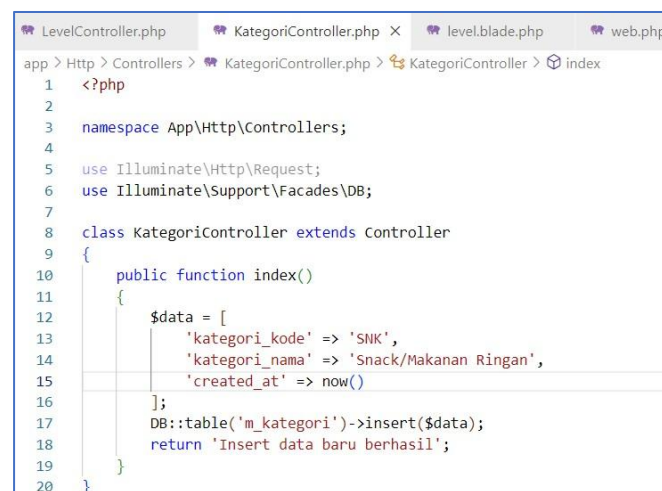
```
php artisan make:controller KategoriController
```

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



```
LevelController.php KategoriController.php level.blade.php web.php X
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

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

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     }

```

8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table

9. 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 }

```

10. 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>

```

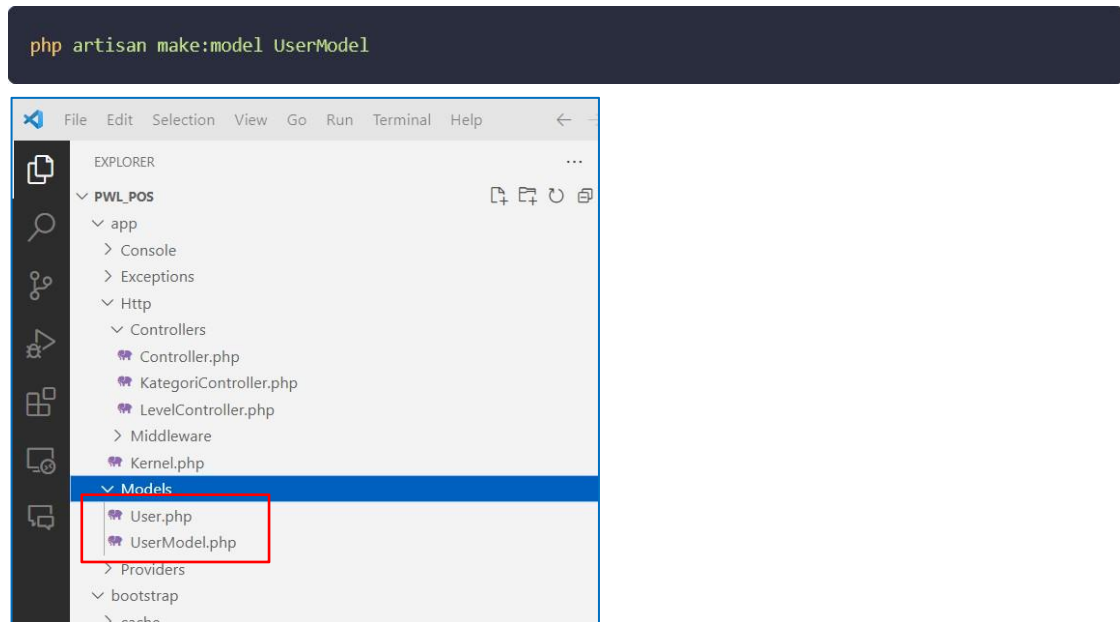
11. Silahkan dicoba pada browser dan amati apa yang terjadi.

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

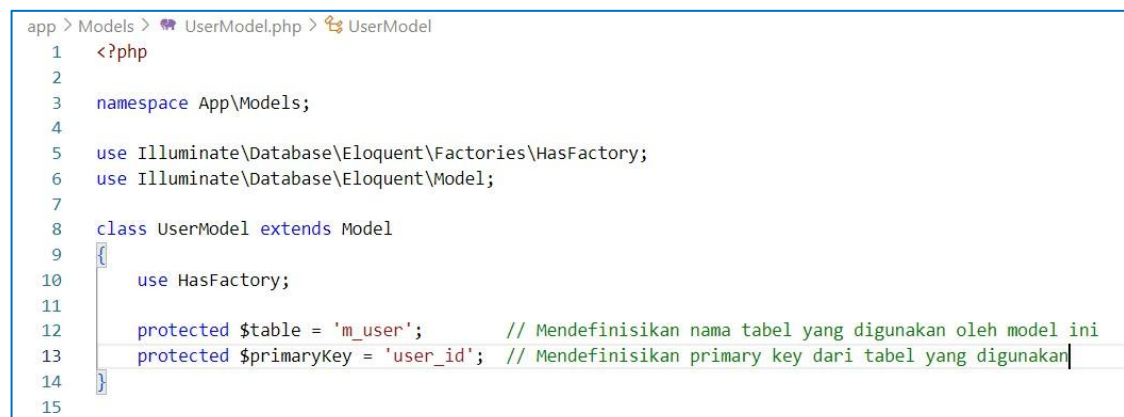
F. ELOQUENT ORM

Praktikum 6 - Implementasi Eloqurnt ORM

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



2. 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`



3. Kita buka file `UserModel.php` dan modifikasi seperti berikut

```

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']);

```

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

```

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

6. Kemudian kita buat view user.blade.php

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

```

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 }

```

8. Setelah itu, kita modifikasi lagi file UserController

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

```
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 }
```

10. Kita modifikasi lagi file UserController menjadi seperti beri

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?
 - APP_KEY digunakan untuk enkripsi dalam Laravel, seperti hashing password dan enkripsi data sensitif. Key ini memastikan keamanan dengan mengenkripsi session dan token yang digunakan dalam autentikasi
2. Pada Praktikum 1, bagaimana kita men-generate nilai untuk APP_KEY?
 - Menggunakan 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 4 file migrasi yaitu:
 - create_users_table.php (untuk menyimpan data pengguna)
 - create_password_reset_tokens_table.php (untuk menyimpan token reset password)
 - create_failed_jobs_table.php (menyimpan antrian pekerjaan (jobs) yang gagal dieksekusi.)
 - create_personal_access_tokens_table.php (untuk API Authentication dengan Laravel Sanctum.)
4. Secara default, file migrasi terdapat kode \$table->timestamps();, apa tujuan/output dari fungsi tersebut?
 - Fungsi ini menambahkan dua kolom created_at dan updated_at secara otomatis untuk mencatat waktu pembuatan dan pembaruan record dalam tabel.
5. Pada File Migrasi, terdapat fungsi \$table->id(); Tipe data apa yang dihasilkan dari fungsi tersebut?
 - Fungsi \$table->id(); menghasilkan kolom dengan tipe data BIGINT yang bersifat auto-increment dan menjadi primary key. Apa bedanya hasil migrasi pada table m_level, antara menggunakan \$table->id();
6. dengan menggunakan \$table->id('level_id'); ?
 - \$table->id(); akan membuat kolom id dengan tipe data BIGINT sebagai primary key.
 - \$table->id('level_id'); akan membuat kolom level_id dengan tipe data BIGINT sebagai primary key, bukan id.

7. Pada migration, Fungsi `->unique()` digunakan untuk apa?
- Fungsi `->unique()` digunakan untuk memastikan bahwa nilai dalam kolom tersebut unik, artinya tidak ada data yang duplikat dalam kolom tersebut.
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')` ?
- Pada tabel `m_level`, `$table->id('level_id')`; digunakan untuk membuat primary key.
 - Pada tabel `m_user`, `$table->unsignedBigInteger('level_id')`; digunakan untuk membuat foreign key yang merujuk ke `level_id` pada `m_level`.
9. Pada Praktikum 3 - Tahap 6, apa tujuan dari Class Hash? dan apa maksud dari kode program `Hash::make('1234');`?
- Class Hash digunakan untuk mengenkripsi password agar tidak tersimpan dalam bentuk teks biasa. `Hash::make('1234')`; akan menghasilkan password yang terenkripsi menggunakan algoritma hashing seperti `bcrypt`.
10. Pada Praktikum 4 - Tahap 3/5/7, pada query builder terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?
- Tanda ? digunakan sebagai placeholder dalam query untuk menghindari SQL Injection dan membuat query lebih aman. Nilai dari placeholder ini akan diisi secara otomatis ketika query dijalankan.
11. Pada Praktikum 6 - Tahap 3, apa tujuan penulisan kode `protected $table = 'm_user';` dan `protected $primaryKey = 'user_id';` ?
- `protected $table = 'm_user';` menentukan bahwa model terkait dengan tabel `m_user`.
 - `protected $primaryKey = 'user_id';` menentukan bahwa primary key tabel ini adalah `user_id`, bukan `id`.
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (DB Façade / Query Builder / Eloquent ORM) ? jelaskan
- DB Façade: Lebih cepat dan langsung menggunakan raw query, tetapi kode bisa sulit dibaca.
 - Query Builder: Memudahkan pembuatan query dengan fleksibilitas tinggi tanpa harus menulis raw SQL.
 - Eloquent ORM: Paling mudah digunakan karena berbasis model OOP, tetapi sedikit lebih lambat dibandingkan Query Builder.