LAPORAN PRAKTIKUM
PEMROGRAMAN WEB LANJUT
MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, dan ELOQUENT ORM

Disusun Oleh :

Qusnul Diah Mawanti 2341760035

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

Jurusan Teknologi Informasi – Politeknik Negeri Malang

Jobsheet 3 : Migration, Sheeder, DB Façade, Query Builder, dan Eloquent Orm

Mata Kuliah : Pemrograman Web Lanjut

Kelas : 2A – SIB

Nama : Qusnul Diah Mawanti

No absen : 25

NIM : 2341760035

Februari 2025

PERSIAPAN

Sebelumnya kita sudah membahas mengenai Routing, Controller, dan View yang ada di Laravel. Sebelum kita masuk pada pembuatan aplikasi berbasis website, alangkah baiknya kita perlu menyiapkan Basis data sebagai tempat menyimpan data-data pada aplikasi kita nanti. Selain itu, umumnya kita perlu menyiapkan juga data awal yang kita gunakan sebelum membuat aplikasi, seperti data user administrator, data pengaturan sistem, dll. Untuk itu, kita memerlukan teknik untuk merancang/membuat table basis data sebelum membuat aplikasi. Laravel memiliki fitur dalam pengelolaan basis data seperti, migration, seeder, model, dll.

Sebelum kita masuk materi, kita buat dulu project baru yang akan kita gunakan untuk membangun aplikasi sederhana dengan topik Point of Sales (PoS), sesuai dengan **Studi Kasus PWL.pdf**. Jadi kita bikin project Laravel 10 dengan nama **PWL_POS**.

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

```
C:\laragon\www>composer create-project "laravel/laravel:^10.0" PWL_POS
Creating a "laravel/laravel:^10.0" project at "./PWL_POS"
Installing laravel/laravel (v10.3.3)
  - Installing laravel/laravel (v10.3.3): Extracting archive
Created project in C:\laragon\www\PWL_POS
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 111 installs, 0 updates, 0 removals
  - Locking brick/math (0.12.3)
```

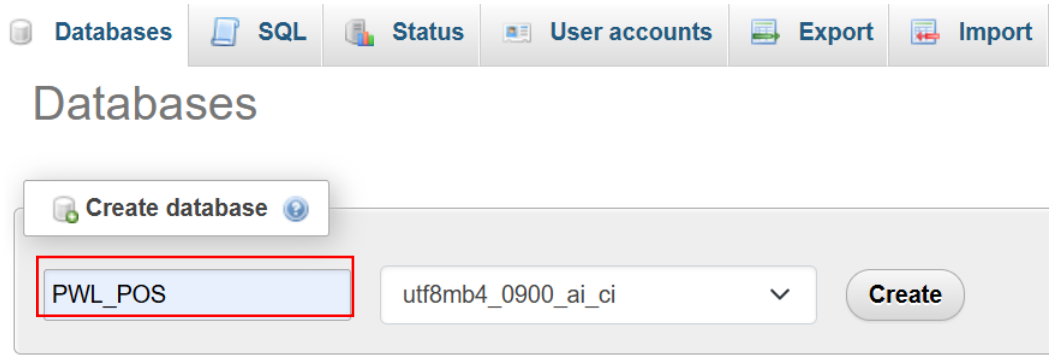
DASAR TEORI

A. Pengaturan Database

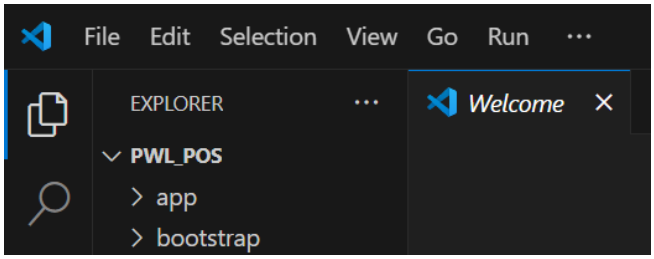
Database atau basis data menjadi komponen penting dalam membangun sistem. Hal ini dikarenakan database menjadi tempat untuk menyimpan data-data transaksi yang ada pada sistem. Koneksi ke database perlu kita atur agar sesuai dengan database yang kita gunakan.

1. Praktikum 1: Pengaturan Database

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

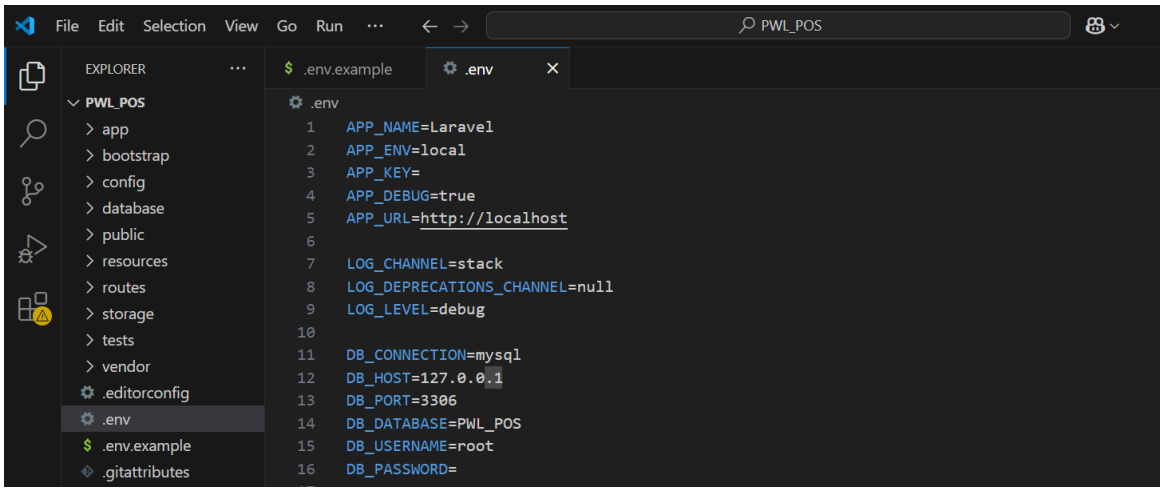


b. Buka aplikasi VSCode dan buka folder project **PWL_POS** yang sudah kita buat.

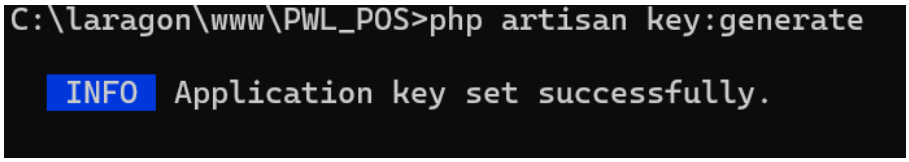


c. Copy file **.env.example** menjadi **.env**

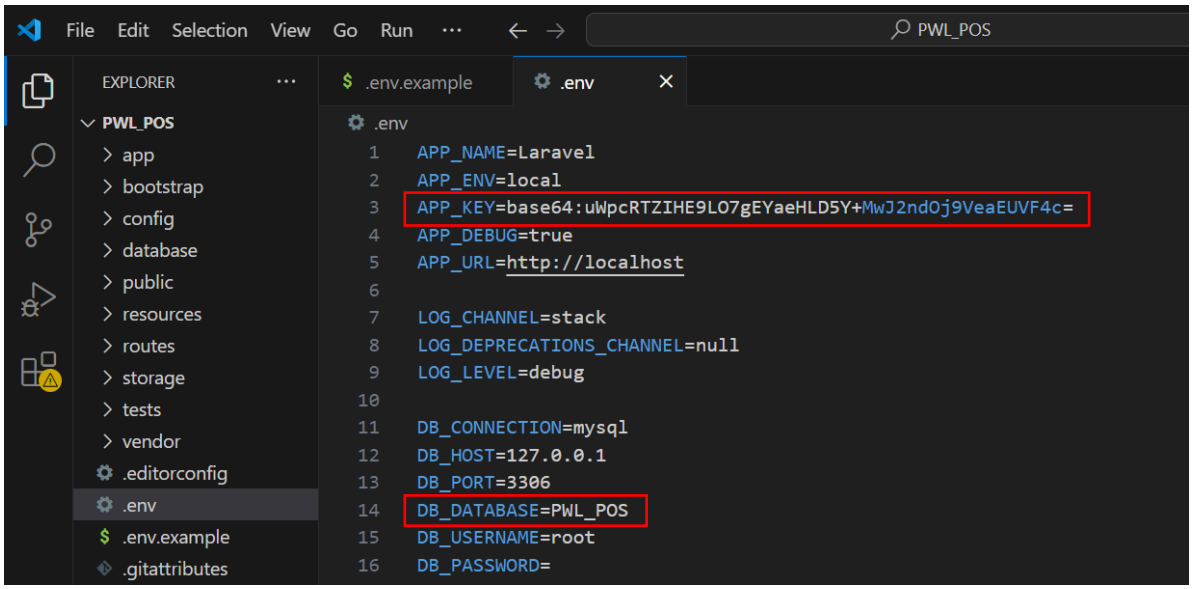
d. Buka file **.env**, dan pastikan konfigurasi **APP_KEY** bernilai.



Jika belum bernilai silahkan kalian generate menggunakan **php artisan**.



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



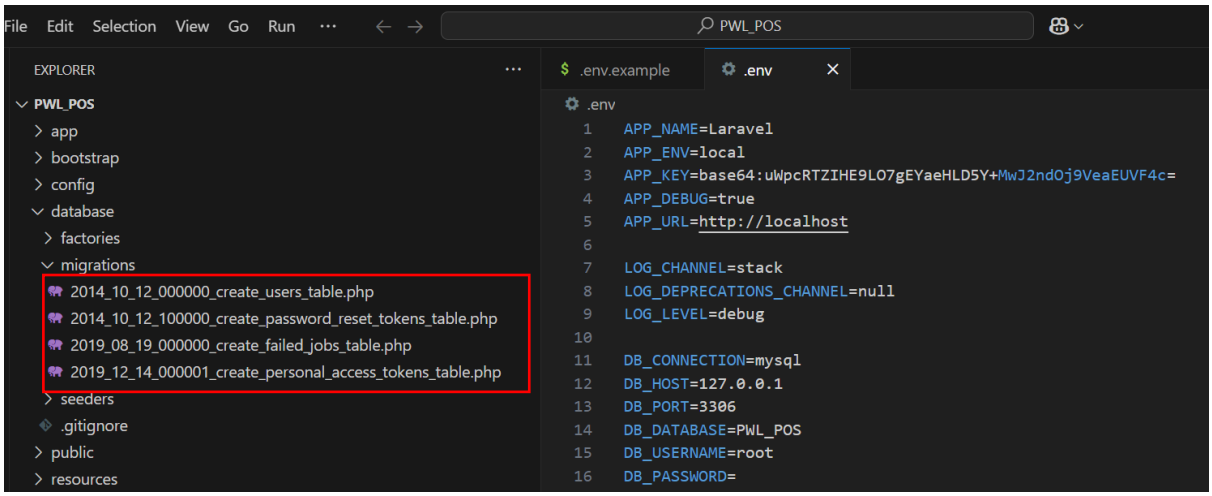
f. Laporkan hasil Praktikum-1 ini dan commit perubahan pada git.

B. Migration

Migration pada Laravel merupakan sebuah fitur yang dapat membantu kita mengelola database secara efisien dengan menggunakan kode program. Migration membantu kita dalam membuat (create), mengubah (edit), dan menghapus (delete) struktur tabel dan kolom pada database yang sudah kita buat dengan cepat dan mudah. Dengan Migration, kita juga dapat melakukan perubahan pada struktur database tanpa harus menghapus data yang ada. Salah satu keunggulan menggunakan migration adalah mempermudah proses instalasi aplikasi kita, Ketika aplikasi yang kita buat akan diimplementasikan di server/komputer lain.

1. Praktikum 2.1: Pembuatan File Migrasi Tanpa Relasi

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



b. Kita abaikan dulu yang di kotak merah (jangan di hapus).

c. Kita buat file migrasi untuk table **m_level** dengan perintah.

```
C:\laragon\www\PWL_POS>php artisan make:migration create_m_level_table

  INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2025_03_05_032139_create_m_level_table.php] created successfully.
```

database > migrations > 2025_03_05_032139_create_m_level_table.php > ...

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('m_level', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }
```

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

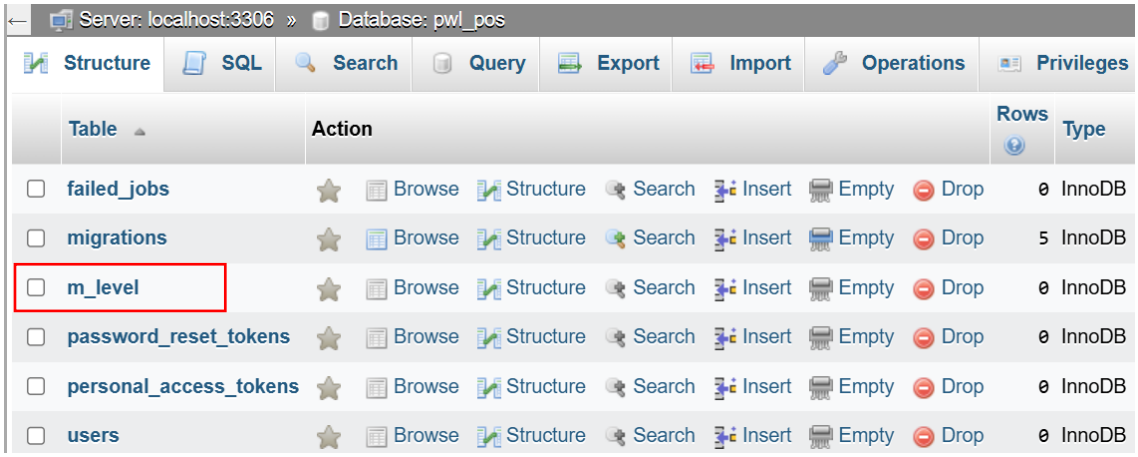
database > migrations > 2025_03_05_032139_create_m_level_table.php > ...

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('m_level', function (Blueprint $table) {
            $table->id('level_id');
            $table->string('level_kode', 10)->unique();
            $table->string('level_nama', 100);
            $table->timestamps();
        });
    }

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

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

```
C:\laragon\www\PWL_POS>php artisan migrate

  INFO  Preparing database.

Creating migration table ..................... 72ms DONE

  INFO  Running migrations.

2014_10_12_000000_create_users_table ............ 33ms DONE
2014_10_12_100000_create_password_reset_tokens_table  11ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 21ms DONE
2019_12_14_000001_create_personal_access_tokens_table  25ms DONE
2025_03_05_032139_create_m_level_table ......... 20ms DONE
```

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



g. Ok, table sudah dibuat di database.

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

- file migrasi untuk table **m_kategori**

```
C:\laragon\www\PWL_POS>php artisan make:migration create_m_kategori_table

  INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2025_03_05_
034628_create_m_kategori_table.php] created successfully.
```

```php
public function up(): void
{
    Schema::create('m_kategori', function (Blueprint $table) {
        $table->bigIncrements('kategori_id');
        $table->string('kategori_kode', 10)->unique();
        $table->string('kategori_nama', 100);
        $table->timestamps();
    });
}
```

- *Migrasi*

```
C:\laragon\www\PWL_POS>php artisan migrate

  INFO  Running migrations.

  2025_03_05_034628_create_m_kategori_table ..........................
..................................................... 67ms DONE
```

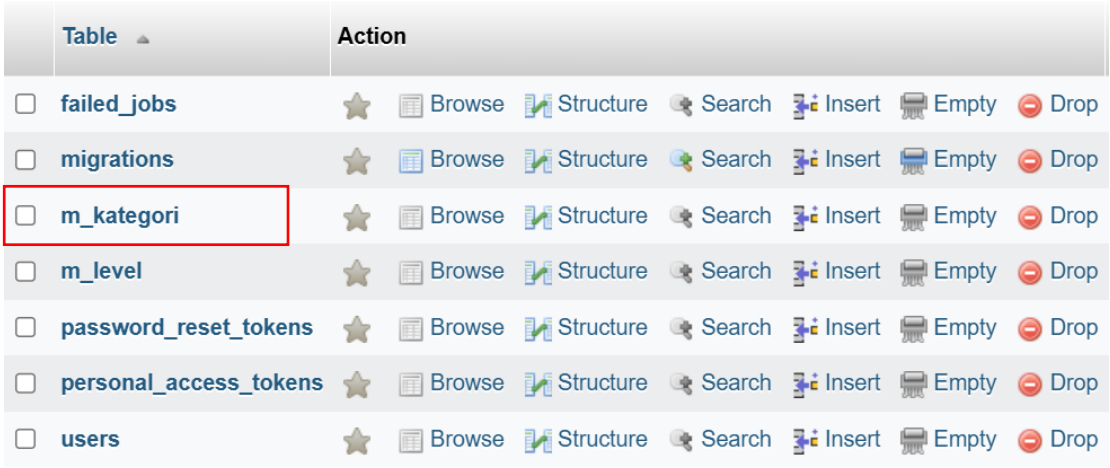


	Table	Action
☐	failed_jobs	Browse Structure Search Insert Empty Drop
☐	migrations	Browse Structure Search Insert Empty Drop
☐	m_kategori	Browse Structure Search Insert Empty Drop
☐	m_level	Browse Structure Search Insert Empty Drop
☐	password_reset_tokens	Browse Structure Search Insert Empty Drop
☐	personal_access_tokens	Browse Structure Search Insert Empty Drop
☐	users	Browse Structure Search Insert Empty Drop

i. Laporan hasil Praktikum-2.1 ini dan commit perubahan pada *git*.

2. Praktikum 2.2: Pembuatan File Migrasi Dengan Relasi

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

```
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/2025_03_05_04075
8_create_m_user_table.php] created successfully.
```

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

```php
public function up(): void {
    if (!Schema::hasTable('m_user')) {
        Schema::create('m_user', function (Blueprint $table) {
            $table->bigIncrements('user_id');
            $table->unsignedBigInteger('level_id');
            $table->string('username', 20)->unique();
            $table->string('nama', 100);
            $table->string('password', 255);
            $table->timestamps();

            // Foreign key ke tabel m_level
            $table->foreign('level_id')->references('level_id')->on('m_level')->onDelete('cascade');
        });
    }
}
```

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

```
C:\laragon\www\PWL_POS>php artisan migrate

  INFO  Running migrations.

  2025_03_05_090111_create_m_user_table ..... 6ms DONE
```

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

m_barang
t_penjualan
t_stok
t_penjualan_detail

- **m_barang**

  file migrasi

```
C:\laragon\www\PWL_POS>php artisan make:migration create_m_barang_table


  INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2025_03_05
_083522_create_m_barang_table.php] created successfully.
```

  Modifikasi file

```php
public function up(): void {
    Schema::create('m_barang', function (Blueprint $table) {
        $table->bigIncrements('barang_id');
        $table->unsignedBigInteger('kategori_id');
        $table->string('barang_kode', 10)->unique();
        $table->string('barang_nama', 100);
        $table->integer('harga_beli');
        $table->integer('harga_jual');
        $table->timestamps();

        // Foreign key ke tabel m_kategori
        $table->foreign('kategori_id')->references('kategori_id')->on('m_kategori')->onDelete('cascade');
    });
}
```

  Migrate

```
C:\laragon\www\PWL_POS>php artisan migrate

  INFO  Running migrations.

  2025_03_05_090949_create_m_barang_table ......... 109ms DONE
```

- **t_penjualan**

  file migrasi

```
C:\laragon\www\PWL_POS>php artisan make:migration create_t_penjualan_table


  INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2025_03_05_
083758_create_t_penjualan_table.php] created successfully.
```

  Modifikasi file

```php
public function up(): void
{
    Schema::create('t_penjualan', function (Blueprint $table) {
        $table->id();
        $table->date('tanggal_penjualan');
        $table->unsignedBigInteger('user_id');
        $table->integer('total_harga');
        $table->timestamps();

        // Foreign Key
        $table->foreign('user_id')->references('id')->on('m_user');
    });
}
```

Migrate

```
C:\laragon\www\PWL_POS>php artisan migrate

  INFO  Running migrations.

  2025_03_05_091144_create_t_penjualan_table ..... 88ms DONE
```

- **t_stok**

  File migrasi

```
C:\laragon\www\PWL_POS>php artisan make:migration create_t_stok_table

  INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2025_03
_05_091715_create_t_stok_table.php] created successfully.
```

  Modifikasi File

```php
public function up(): void {
    Schema::create('t_stok', function (Blueprint $table) {
        $table->bigIncrements('stok_id');
        $table->unsignedBigInteger('barang_id');
        $table->unsignedBigInteger('user_id');
        $table->dateTime('stok_tanggal');
        $table->integer('stok_jumlah');
        $table->timestamps();

        // Foreign key ke tabel m_barang dan m_user
        $table->foreign('barang_id')->references('barang_id')->on('m_barang')->onDelete('cascade');
        $table->foreign('user_id')->references('user_id')->on('m_user')->onDelete('cascade');
    });
}
```

  Migrate

```
C:\laragon\www\PWL_POS>php artisan migrate

  INFO  Running migrations.

  2025_03_05_091715_create_t_stok_table .................. 86ms DONE
```

- **t_penjualan_detail**

  File Migrasi

```
C:\laragon\www\PWL_POS>php artisan make:migration create_t_penjualan_d
etail_table

  INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2025_03
_05_092359_create_t_penjualan_detail_table.php] created successfully.
```

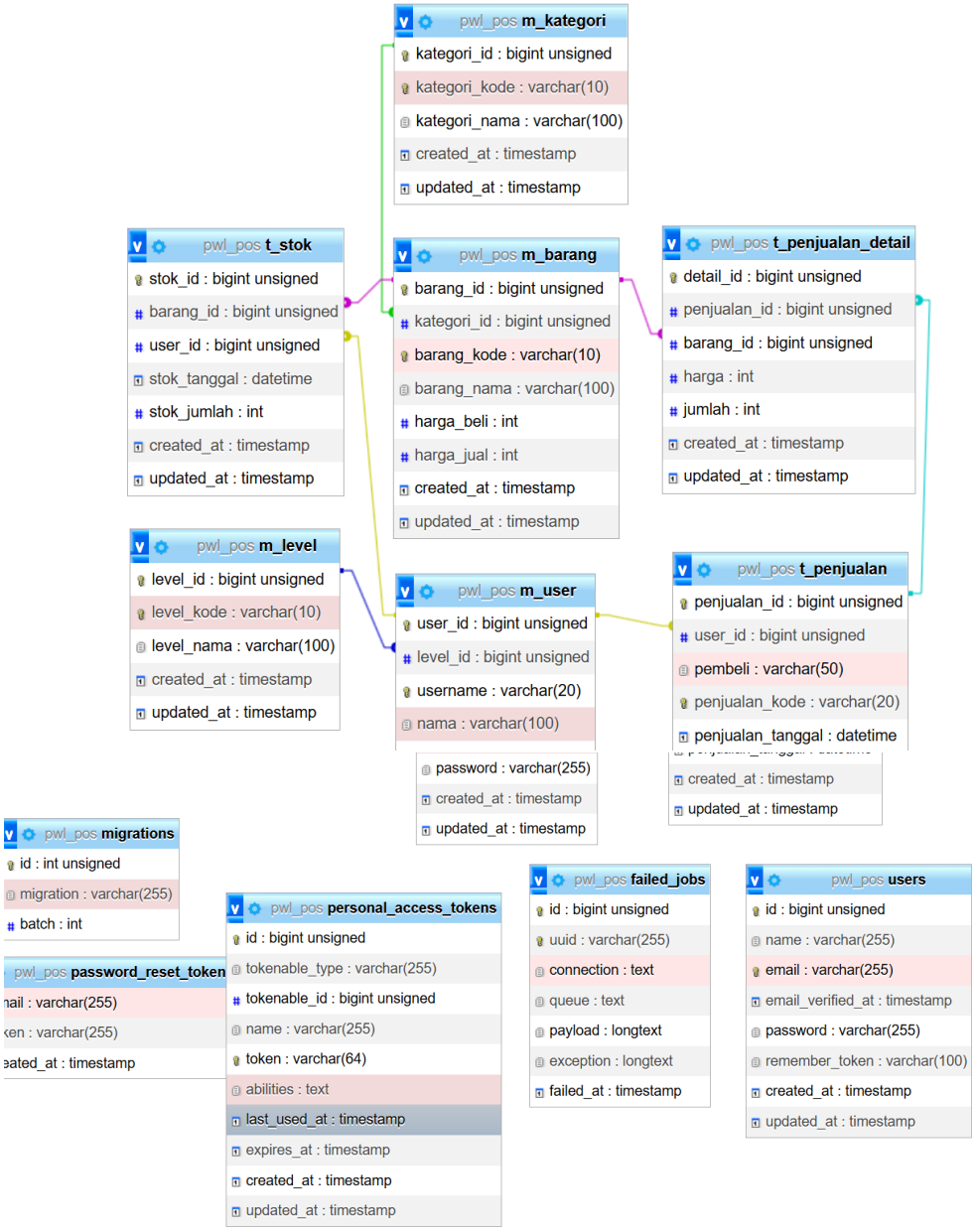
  Modifikasi File

```php
public function up(): void {
    Schema::create('t_penjualan_detail', function (Blueprint $table) {
        $table->bigIncrements('detail_id');
        $table->unsignedBigInteger('penjualan_id');
        $table->unsignedBigInteger('barang_id');
        $table->integer('harga');
        $table->integer('jumlah');
        $table->timestamps();

        // Foreign key ke tabel t_penjualan dan m_barang
        $table->foreign('penjualan_id')->references('penjualan_id')->on('t_penjualan')->onDelete('cascade');
        $table->foreign('barang_id')->references('barang_id')->on('m_barang')->onDelete('cascade');
    });
}
```

Migrate

```
C:\laragon\www\PWL_POS>php artisan migrate

  INFO  Running migrations.

  2025_03_05_092359_create_t_penjualan_detail_table ..... 135ms DONE
```

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



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

C. Seeder

Seeder merupakan sebuah fitur yang memungkinkan kita untuk mengisi database kita dengan data awal atau data *dummy* yang telah ditentukan. Seeder memungkinkan kita untuk membuat data awal yang sama untuk setiap penggunaan dalam pembangunan aplikasi. Umumnya, data yang sering dibuat *seeder* adalah data penggunna karena data tersebut akan digunakan saat aplikasi pertama kali di jalankan dan membutuhkan aksi *login*.

1. Praktikum 3: Membuat File *Seeder*

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

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

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

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

database > seeders > LevelSeeder.php > ...

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class LevelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run()
    {
        $data = [
            ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
            ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
            ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staf/Kasir'],
        ];
        DB::table('m_level')->insert($data);
    }
}
```

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

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=LevelSeeder

  INFO  Seeding database.
```

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

	level_id	level_kode	level_nama	created_at	updated_at
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staf/Kasir	NULL	NULL

Check all With selected: Edit Copy Delete Export

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

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

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

f. Modifikasi file **class UserSeeder** seperti berikut.

```php
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run()
    {
        $data = [
            [
                'user_id' => 1,
                'level_id' => 1,
                'username' => 'admin',
                'nama' => 'Administrator',
                'password' => Hash::make('12345'), // class untuk mengenkripsi/hash password
            ],
            [
                'user_id' => 2,
                'level_id' => 2,
                'username' => 'manager',
                'nama' => 'Manager',
                'password' => Hash::make('12345'),
            ],
            [
                'user_id' => 3,
                'level_id' => 3,
                'username' => 'staf',
                'nama' => 'Staf/Kasir',
                'password' => Hash::make('12345'),
            ],
        ];
        DB::table('m_user')->insert($data);
    }
}
```

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

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=UserSeeder

  INFO  Seeding database.
```

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

	user_id	level_id	username	nama	password
Edit Copy Delete	1	1	admin	Administrator	$2y$12$njFEs2WS04WuYQoKLPnFguW1hFTOz9o.TZeyfdFTl.0...
Edit Copy Delete	2	2	manager	Manager	$2y$12$Knb1OTxo6ufl4KH9P.oOCOdAvIdBxiUCICyxWXtS/P...
Edit Copy Delete	3	3	staf	Staf/Kasir	$2y$12$XryQK9sK4uY7mmO0TSsDQeuCXU5PjnYk7xnrczMFCBQ...

i. Ok, data seeder berhasil di masukkan ke database.
j. 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

- **KategorySeeder.php**

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

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

database > seeders > KategoriSeeder.php > ...

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class KategoriSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run()
    {
        $data = [
            ['kategori_id' => 1, 'kategori_kode' => 'ELK', 'kategori_nama' => 'Elektronik'],
            ['kategori_id' => 2, 'kategori_kode' => 'FAS', 'kategori_nama' => 'Fashion'],
            ['kategori_id' => 3, 'kategori_kode' => 'MKN', 'kategori_nama' => 'Makanan'],
            ['kategori_id' => 4, 'kategori_kode' => 'ATK', 'kategori_nama' => 'Alat Tulis'],
            ['kategori_id' => 5, 'kategori_kode' => 'SPT', 'kategori_nama' => 'Sport'],
        ];
        DB::table('m_kategori')->insert($data);
    }
}
```

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=KategoriSeeder

  INFO  Seeding database.
```

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
Edit Copy Delete	1	ELK	Elektronik	NULL	NULL
Edit Copy Delete	2	FAS	Fashion	NULL	NULL
Edit Copy Delete	3	MKN	Makanan	NULL	NULL
Edit Copy Delete	4	ATK	Alat Tulis	NULL	NULL
Edit Copy Delete	5	SPT	Sport	NULL	NULL

- **BarangSeeder.php**

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

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

```php
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class BarangSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run()
    {
        $data = [
            ['barang_id' => 1, 'kategori_id' => 1, 'barang_kode' => 'TV001', 'barang_nama' => 'Televisi', 'harga_beli
            ['barang_id' => 2, 'kategori_id' => 1, 'barang_kode' => 'HP001', 'barang_nama' => 'Handphone', 'harga_bel
            ['barang_id' => 3, 'kategori_id' => 2, 'barang_kode' => 'SW001', 'barang_nama' => 'Sweater', 'harga_beli'
            ['barang_id' => 4, 'kategori_id' => 2, 'barang_kode' => 'JNS001', 'barang_nama' => 'Jeans', 'harga_beli'
            ['barang_id' => 5, 'kategori_id' => 3, 'barang_kode' => 'MSG001', 'barang_nama' => 'Mie Instan', 'harga_b
            ['barang_id' => 6, 'kategori_id' => 3, 'barang_kode' => 'SUS001', 'barang_nama' => 'Susu Kotak', 'harga_b
            ['barang_id' => 7, 'kategori_id' => 4, 'barang_kode' => 'PEN001', 'barang_nama' => 'Pulpen', 'harga_beli'
            ['barang_id' => 8, 'kategori_id' => 4, 'barang_kode' => 'BKU001', 'barang_nama' => 'Buku Tulis', 'harga_b
            ['barang_id' => 9, 'kategori_id' => 5, 'barang_kode' => 'BOL001', 'barang_nama' => 'Bola Sepak', 'harga_b
            ['barang_id' => 10, 'kategori_id' => 5, 'barang_kode' => 'RKT001', 'barang_nama' => 'Raket Badminton', 'h
        ];
        DB::table('m_barang')->insert($data);
    }
}
```

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=BarangSeeder

  INFO  Seeding database.
```

	barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
Edit Copy Delete	1	1	TV001	Televisi	2000000	2500000	NULL	NULL
Edit Copy Delete	2	1	HP001	Handphone	3000000	3500000	NULL	NULL
Edit Copy Delete	3	2	SW001	Sweater	150000	200000	NULL	NULL
Edit Copy Delete	4	2	JNS001	Jeans	200000	250000	NULL	NULL
Edit Copy Delete	5	3	MSG001	Mie Instan	2500	3500	NULL	NULL
Edit Copy Delete	6	3	SUS001	Susu Kotak	5000	7000	NULL	NULL
Edit Copy Delete	7	4	PEN001	Pulpen	2000	4000	NULL	NULL
Edit Copy Delete	8	4	BKU001	Buku Tulis	5000	8000	NULL	NULL
Edit Copy Delete	9	5	BOL001	Bola Sepak	100000	150000	NULL	NULL
Edit Copy Delete	10	5	RKT001	Raket Badminton	150000	200000	NULL	NULL

- **StokSeeder.php**

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

  INFO  Seeder [C:\laragon\www\PWL_POS\database\seeders\StokSeeder.ph
p] created successfully.
```

```php
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class StokSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run()
    {
        $data = [];
        for ($i = 1; $i <= 10; $i++) {
            $data[] = [
                'stok_id' => $i,
                'barang_id' => $i,
                'user_id' => rand(1, 3),
                'stok_tanggal' => now(),
                'stok_jumlah' => rand(10, 50),
            ];
        }
        DB::table('t_stok')->insert($data);
    }
}
```

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=StokSeeder

  INFO  Seeding database.
```

	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
Edit Copy Delete	1	1	3	2025-03-05 12:17:25	43	NULL	NULL
Edit Copy Delete	2	2	3	2025-03-05 12:17:25	23	NULL	NULL
Edit Copy Delete	3	3	1	2025-03-05 12:17:25	31	NULL	NULL
Edit Copy Delete	4	4	1	2025-03-05 12:17:25	38	NULL	NULL
Edit Copy Delete	5	5	1	2025-03-05 12:17:25	17	NULL	NULL
Edit Copy Delete	6	6	2	2025-03-05 12:17:25	43	NULL	NULL
Edit Copy Delete	7	7	3	2025-03-05 12:17:25	41	NULL	NULL
Edit Copy Delete	8	8	1	2025-03-05 12:17:25	39	NULL	NULL
Edit Copy Delete	9	9	3	2025-03-05 12:17:25	36	NULL	NULL
Edit Copy Delete	10	10	3	2025-03-05 12:17:25	10	NULL	NULL

- **PenjualanSeeder.php**

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

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

```php
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class PenjualanSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run()
    {
        $data = [];
        for ($i = 1; $i <= 10; $i++) {
            $data[] = [
                'penjualan_id' => $i,
                'user_id' => rand(1, 3),
                'pembeli' => 'Pembeli ' . $i,
                'penjualan_kode' => 'TRX00' . $i,
                'penjualan_tanggal' => now(),
            ];
        }
        DB::table('t_penjualan')->insert($data);
    }
}
```

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=PenjualanSeeder

  INFO  Seeding database.
```

	penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
Edit Copy Delete	1	2	Pembeli 1	TRX001	2025-03-05 12:20:42	NULL	NULL
Edit Copy Delete	2	3	Pembeli 2	TRX002	2025-03-05 12:20:42	NULL	NULL
Edit Copy Delete	3	3	Pembeli 3	TRX003	2025-03-05 12:20:42	NULL	NULL
Edit Copy Delete	4	2	Pembeli 4	TRX004	2025-03-05 12:20:42	NULL	NULL
Edit Copy Delete	5	1	Pembeli 5	TRX005	2025-03-05 12:20:42	NULL	NULL
Edit Copy Delete	6	1	Pembeli 6	TRX006	2025-03-05 12:20:42	NULL	NULL
Edit Copy Delete	7	1	Pembeli 7	TRX007	2025-03-05 12:20:42	NULL	NULL
Edit Copy Delete	8	1	Pembeli 8	TRX008	2025-03-05 12:20:42	NULL	NULL
Edit Copy Delete	9	1	Pembeli 9	TRX009	2025-03-05 12:20:42	NULL	NULL
Edit Copy Delete	10	1	Pembeli 10	TRX0010	2025-03-05 12:20:42	NULL	NULL

- **PenjualanDetailSeeder.php**

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

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

```php
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class PenjualanDetailSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run()
    {
        $data = [];
        for ($i = 1; $i <= 10; $i++) {
            for ($j = 1; $j <= 3; $j++) {
                $barang_id = rand(1, 10);
                $harga = DB::table('m_barang')->where('barang_id', $barang_id)->value('harga_jual');

                $data[] = [
                    'detail_id' => (($i - 1) * 3) + $j,
                    'penjualan_id' => $i,
                    'barang_id' => $barang_id,
                    'harga' => $harga,
                    'jumlah' => rand(1, 5),
                ];
            }
        }
        DB::table('t_penjualan_detail')->insert($data);
    }
}
```

```
C:\laragon\www\PWL_POS>php artisan db:seed --class=PenjualanDetailSeeder

  INFO  Seeding database.
```

	detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
Edit Copy Delete	1	1	10	200000	5	NULL	NULL
Edit Copy Delete	2	1	3	200000	5	NULL	NULL
Edit Copy Delete	3	1	10	200000	2	NULL	NULL
Edit Copy Delete	4	2	1	2500000	4	NULL	NULL
Edit Copy Delete	5	2	4	250000	2	NULL	NULL
Edit Copy Delete	6	2	7	4000	4	NULL	NULL
Edit Copy Delete	7	3	6	7000	4	NULL	NULL
Edit Copy Delete	8	3	10	200000	4	NULL	NULL
Edit Copy Delete	9	3	6	7000	5	NULL	NULL
Edit Copy Delete	10	4	5	3500	1	NULL	NULL
Edit Copy Delete	11	4	1	2500000	3	NULL	NULL
Edit Copy Delete	12	4	8	8000	5	NULL	NULL
Edit Copy Delete	13	5	9	150000	2	NULL	NULL

☐	Edit	Copy	Delete	14	5	10	200000	1	NULL	NULL
☐	Edit	Copy	Delete	15	5	2	3500000	1	NULL	NULL
☐	Edit	Copy	Delete	16	6	9	150000	4	NULL	NULL
☐	Edit	Copy	Delete	17	6	2	3500000	1	NULL	NULL
☐	Edit	Copy	Delete	18	6	2	3500000	1	NULL	NULL
☐	Edit	Copy	Delete	19	7	10	200000	4	NULL	NULL
☐	Edit	Copy	Delete	20	7	2	3500000	2	NULL	NULL
☐	Edit	Copy	Delete	21	7	2	3500000	2	NULL	NULL
☐	Edit	Copy	Delete	22	8	9	150000	2	NULL	NULL
☐	Edit	Copy	Delete	23	8	6	7000	4	NULL	NULL
☐	Edit	Copy	Delete	24	8	4	250000	5	NULL	NULL
☐	Edit	Copy	Delete	25	9	10	200000	4	NULL	NULL

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

D. DB Facade

DB Façade merupakan fitur dari Laravel yang digunakan untuk melakukan query secara langsung dengan mengetikkan perinta SQL secara utuh (raw query). Disebut raw query (query mentah) karena penulisan query pada DB Façade langsung ditulis sebagaimana yang biasa dituliskan pada database, seperti **"select * from m_user"** atau **"insert into m_user…"** atau **"update m_user set … Where …"**

Raw query adalah cara paling dasar dan tradisional yang ada di Laravel. Raw query terasa familiar karena biasa kita pakai ketika melakukan query langsung ke database.

1. Praktikum 4: Implementasi DB Facade

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

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

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

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

routes > web.php > ...

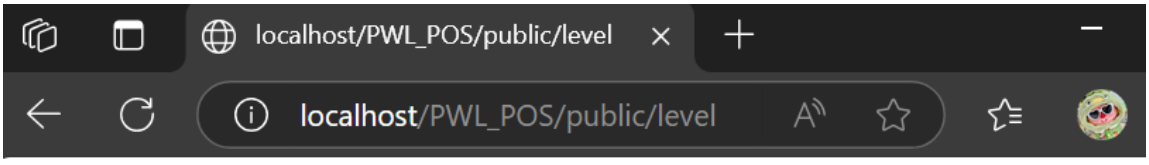
```php
<?php

use App\Http\Controllers\LevelController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Route::get('/level', [LevelController::class, 'index']);
```

c. Selanjutnya, kita modifikasi file **LevelController** untuk menambahkan 1 data ke table **m_level**

app > Http > Controllers > LevelController.php > LevelController > index

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController extends Controller
{
    public function index() {
        DB::insert('insert into m_level(level_kode, level_nama, created_at)
                    values(?,?,?)', ['CUS', 'Pelanggan', now()]);
        return 'Insert data baru berhasil';
    }
}
```

d. Kita coba jalankan di browser dengan url **localhost/PWL_POS/public/level**



Insert data baru berhasil

Dan amati apa yang terjadi pada table **m_level** di database

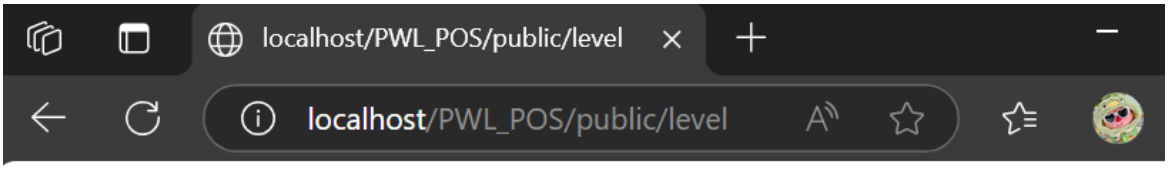
	level_id	level_kode	level_nama	created_at	updated_at
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staf/Kasir	NULL	NULL
Edit Copy Delete	4	CUS	Pelanggan	2025-03-05 13:00:43	NULL

screenshot perubahan yang ada pada table **m_level**

e. Selanjutanya, kita modifikasi lagi file **LevelController** untuk meng-update data di table **m_level** seperti berikut

```
class LevelController extends Controller
{
    public function index() {
        // Menambahkan Data
        // DB::insert('insert into m_level(level_kode, level_nama, created_at)
        //         values(?,?,?)', ['CUS', 'Pelanggan', now()]);
        // return 'Insert data baru berhasil';

        // Meng-update data
        $row = DB::update('update m_level set level_nama = ? where level_kode = ?',
                          ['Customer', 'CUS']);
        return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';
    }
}
```

f. Kita coba jalankan di browser dengan url **localhost/PWL_POS/public/level** lagi



Update data berhasil. Jumlah data yang diupdate: 1 baris

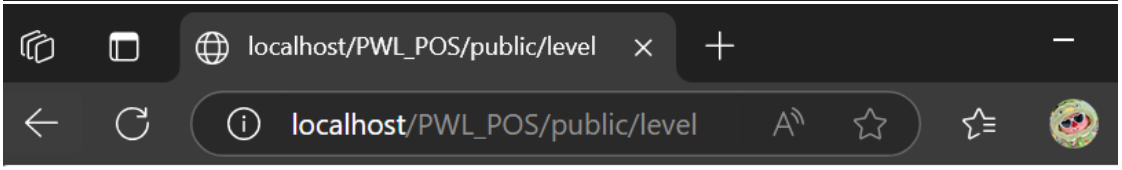
dan amati apa yang terjadi pada table **m_level** di database,

	level_id	level_kode	level_nama	created_at	updated_at
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staf/Kasir	NULL	NULL
Edit Copy Delete	4	CUS	Customer	2025-03-05 13:00:43	NULL

screenshot perubahan yang ada pada table **m_level.**

g. Kita coba modifikasi lagi file **LevelController** untuk melakukan proses hapus data

```php
class LevelController extends Controller
{
    public function index() {
        // Menambahkan Data
        // DB::insert('insert into m_level(level_kode, level_nama, created_at)
        //         values(?,?,?)', ['CUS', 'Pelanggan', now()]);
        // return 'Insert data baru berhasil';

        // Meng-update data
        // $row = DB::update('update m_level set level_nama = ? where level_kode = ?',
        //          ['Customer', 'CUS']);
        // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';

        // Hapus data
        $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
        return 'Delete data berhasil. Jumlah data yang dihapus: ' .$row.' baris';
    }
}
```



Delete data berhasil. Jumlah data yang dihapus: 1 baris

	level_id	level_kode	level_nama	created_at	updated_at
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staf/Kasir	NULL	NULL

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

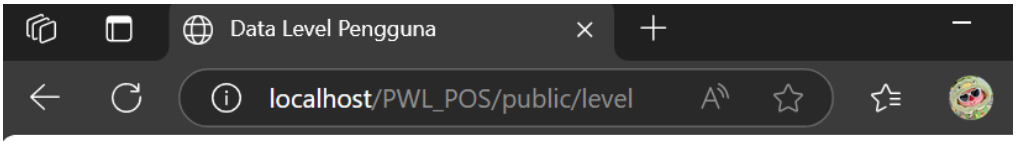
```php
class LevelController extends Controller
{
    public function index() {
        // Menambahkan Data
        // DB::insert('insert into m_level(level_kode, level_nama, created_at)
        //         values(?,?,?)', ['CUS', 'Pelanggan', now()]);
        // return 'Insert data baru berhasil';

        // Meng-update data
        // $row = DB::update('update m_level set level_nama = ? where level_kode = ?',
        //          ['Customer', 'CUS']);
        // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row.' baris';

        // Hapus data
        // $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
        // return 'Delete data berhasil. Jumlah data yang dihapus: ' .$row.' baris';

        //Menampilkan data
        $data = DB::select('select * from m_level');
        return view('level' , ['data' => $data]);
    }
}
```

i. 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 > ...

```
<!DOCTYPE html>
<html>
    <head>
        <title>Data Level Pengguna</title>
    </head>
    <body>
        <h1>Daftar Level Pengguna</h1>
        <table border="1" cellpadding="2" cellpacing="0">
            <tr>
                <th>ID</th>
                <th>Kode</th>
                <th>Nama Level</th>
            </tr>
            @foreach ($data as $d)
            <tr>
                <td>{{ $d->level_id }}</td>
                <td>{{ $d->level_kode }}</td>
                <td>{{ $d->level_nama }}</td>
            </tr>
            @endforeach
        </table>
    </body>
</html>
```

j. Silahkan dicoba pada browser dan amati apa yang terjadi



Daftar Level Pengguna

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

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

E. Query Builder

Query builder adalah fitur yang disediakan Laravel untuk melakukan proses CRUD (create, retrieve/read, update, delete) pada database. Berbeda dengan raw query pada DB Facede yang mengharuskan kita menulis perintah SQL, pada query builder perintah SQL ini diakses menggunakan method. Jadi, kita tidak menulis perintah SQL secara langsung, melainkan cukup memanggil method-method yang ada di query builder.

1. Praktikum 5: Implementasi *Query Builder*

a. Kita buat controller dahuku untuk mengelola data pada table **m_kategori**

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

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

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

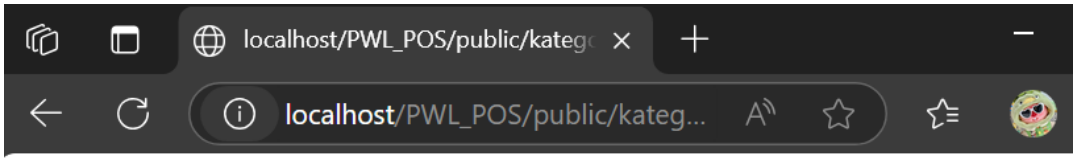
routes > web.php > ...

```php
<?php

use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Route::get('/level', [LevelController::class, 'index']);
Route::get('/kategori', [KategoriController::class, 'index']);
```

c. Selanjutnya, kita modifikasi file **KategoriController** untuk menambahkan 1 data ke table **m_kategori**

```php
app > Http > Controllers > KategoriController.php > ...
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class KategoriController extends Controller
{
    public function index() {
        $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snak/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil';
    }
}
```

d. Kita coba jalankan di browser dengan url **localhost/PWL_POS/public/kategori**



Insert data baru berhasil

dan amati apa yang terjadi pada table **m_kategori** di database,

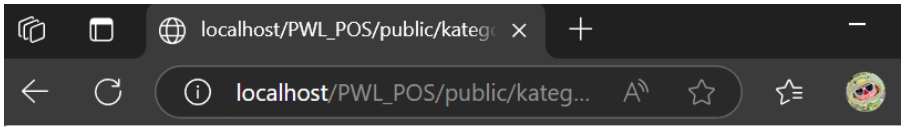
	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
Edit Copy Delete	1	ELK	Elektronik	NULL	NULL
Edit Copy Delete	2	FAS	Fashion	NULL	NULL
Edit Copy Delete	3	MKN	Makanan	NULL	NULL
Edit Copy Delete	4	ATK	Alat Tulis	NULL	NULL
Edit Copy Delete	5	SPT	Sport	NULL	NULL
Edit Copy Delete	6	SNK	Snak/Makanan Ringan	2025-03-05 13:48:10	NULL

screenshot perubahan yang ada pada table **m_kategori**

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

```php
class KategoriController extends Controller
{
    public function index() {

        // Menambah data
        /* $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snak/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil'; */

        // Update data
        $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')
                    ->update(['kategori_nama' => 'Camilan']);
        return 'Update data berhasil. Jumlah data yang diupdate: ' .$row.' baris';
    }
}
```

f. Kita coba jalankan di browser dengan url **localhost/PWL_POS/public/kategori**



Update data berhasil. Jumlah data yang diupdate: 1 baris

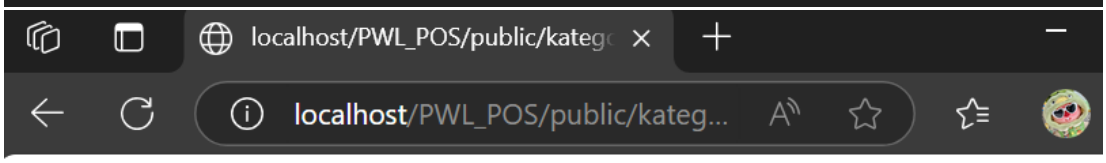
Dan amati apa yang terjadi pada table **m_kategori** di database,

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
Edit Copy Delete	1	ELK	Elektronik	NULL	NULL
Edit Copy Delete	2	FAS	Fashion	NULL	NULL
Edit Copy Delete	3	MKN	Makanan	NULL	NULL
Edit Copy Delete	4	ATK	Alat Tulis	NULL	NULL
Edit Copy Delete	5	SPT	Sport	NULL	NULL
Edit Copy Delete	6	SNK	Camilan	2025-03-05 13:48:10	NULL

screenshot perubahan yang ada pada table **m_kategori**

g. Kita coba modifikasi lagi file **KategoriController** untuk melakukan proses hapus data

```php
class KategoriController extends Controller
{
    public function index() {

        // Menambah data
        /* $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snak/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil'; */

        // Update data
        /* $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')
                    ->update(['kategori_nama' => 'Camilan']);
        return 'Update data berhasil. Jumlah data yang diupdate: ' .$row.' baris'; */

        //Hapus data
        $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
        return 'Delete data berhasil. Jumlah data yang dihapus: ' .$row.'baris';
    }
}
```



Delete data berhasil. Jumlah data yang dihapus: 1baris

←T→		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
☐	Edit Copy Delete	1	ELK	Elektronik	NULL	NULL
☐	Edit Copy Delete	2	FAS	Fashion	NULL	NULL
☐	Edit Copy Delete	3	MKN	Makanan	NULL	NULL
☐	Edit Copy Delete	4	ATK	Alat Tulis	NULL	NULL
☐	Edit Copy Delete	5	SPT	Sport	NULL	NULL

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

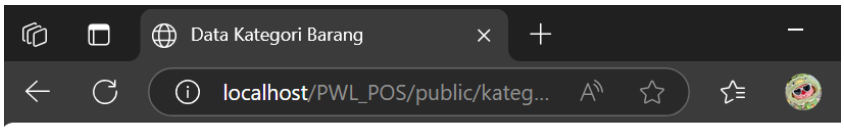
```php
class KategoriController extends Controller
{
    public function index() {

        // Menambah data
        /* $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snak/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil'; */

        // Update data
        /* $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')
                    ->update(['kategori_nama' => 'Camilan']);
        return 'Update data berhasil. Jumlah data yang diupdate: ' .$row.' baris'; */

        //Hapus data
        /* $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
        return 'Delete data berhasil. Jumlah data yang dihapus: ' .$row.'baris'; */

        // Menampilkan data
        $data = DB::table('m_kategori')->get();
        return view('kategori', ['data'=> $data]);
    }
}
```

i. 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 > ...

```blade
<!DOCTYPE html>
<html>
    <head>
        <title>Data Kategori Barang</title>
    </head>
    <body>
        <h1>Daftar Kategori Barang</h1>
        <table border="1" cellpadding="2" cellpacing="0">
            <tr>
                <th>ID</th>
                <th>Kode</th>
                <th>Nama Kategori</th>
            </tr>
            @foreach ($data as $d)
            <tr>
                <td>{{ $d->kategori_id }}</td>
                <td>{{ $d->kategori_kode }}</td>
                <td>{{ $d->kategori_nama }}</td>
            </tr>
            @endforeach
        </table>
    </body>
</html>
```

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



Daftar Kategori Barang

ID	Kode	Nama Kategori
1	ELK	Elektronik
2	FAS	Fashion
3	MKN	Makanan
4	ATK	Alat Tulis
5	SPT	Sport

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

F. Eloquent Orm

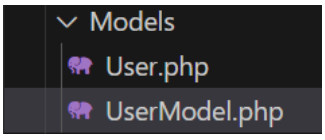
Eloquent ORM adalah fitur bawaan dari laravel. Eloquent ORM adalah cara pengaksesan database dimana setiap baris tabel dianggap sebagai sebuah object. Kata ORM sendiri merupakan singkatan dari Object-relational mapping, yakni suatu teknik programming untuk mengkonversi data ke dalam bentuk object.

1. Praktikum 6: Implementasi Eloquent ORM

a. Kita buat file model untuk tabel **m_user** dengan mengetikkan perintah

```
C:\laragon\www\PWL_POS>php artisan make:model UserModel

  INFO  Model [C:\laragon\www\PWL_POS\app\Models\UserModel.php] created
successfully.
```

b. 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**



c. Kita buka file **UserModel.php** dan modifikasi seperti berikut.

app > Models > UserModel.php > ...

```php
<?php

namespace App\Models;

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

class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user';        //Mendefinisikan nama tabel yang digunakan oleh model ini
    protected $primaryKey = 'user_id';  //Mendefinisikan primary key dari tabel yang digunakan
}
```

d. Kita modifikasi route **web.php** untuk mencoba routing ke controller **UserController.**

routes > web.php > ...

```php
<?php

use App\Http\Controllers\KategoriController;
use App\Http\Controllers\LevelController;
use App\Http\Controllers\UserController;
use Illuminate\Support\Facades\Route;

/*
|--------------------------------------------------------------------------
| Web Routes
|--------------------------------------------------------------------------
|
| Here is where you can register web routes for your application. These
| routes are loaded by the RouteServiceProvider and all of them will
| be assigned to the "web" middleware group. Make something great!
|
*/

Route::get('/', function () {
    return view('welcome');
});

Route::get('/level', [LevelController::class, 'index']);
Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);
```

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

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

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

app > Http > Controllers > UserController.php > ...

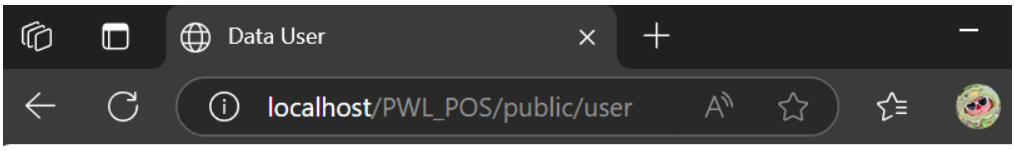
```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;

class UserController extends Controller
{
    public function index() {
        // coba akses model UserModel
        $user = UserModel::all(); // ambil semua data dari tabel m_user
        return view('user', ['data' => $user]);
    }
}
```

f. Kemudian kita buat view **user.blade.php**

resources > views > user.blade.php > ...

```
<!DOCTYPE html>
<html>
    <head>
        <title>Data User</title>
    </head>
    <body>
        <h1>Data User</h1>
        <table border="1" cellpadding="2" cellpacing="0">
            <tr>
                <th>ID</th>
                <th>username</th>
                <th>Nama</th>
                <th>ID Level Pengguna</th>
            </tr>
            @foreach ($data as $d)
            <tr>
                <td>{{ $d->user_id }}</td>
                <td>{{ $d->username }}</td>
                <td>{{ $d->nama }}</td>
                <td>{{ $d->level_id }}</td>
            </tr>
            @endforeach
        </table>
    </body>
</html>
```

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



Data User

ID	username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staf	Staf/Kasir	3

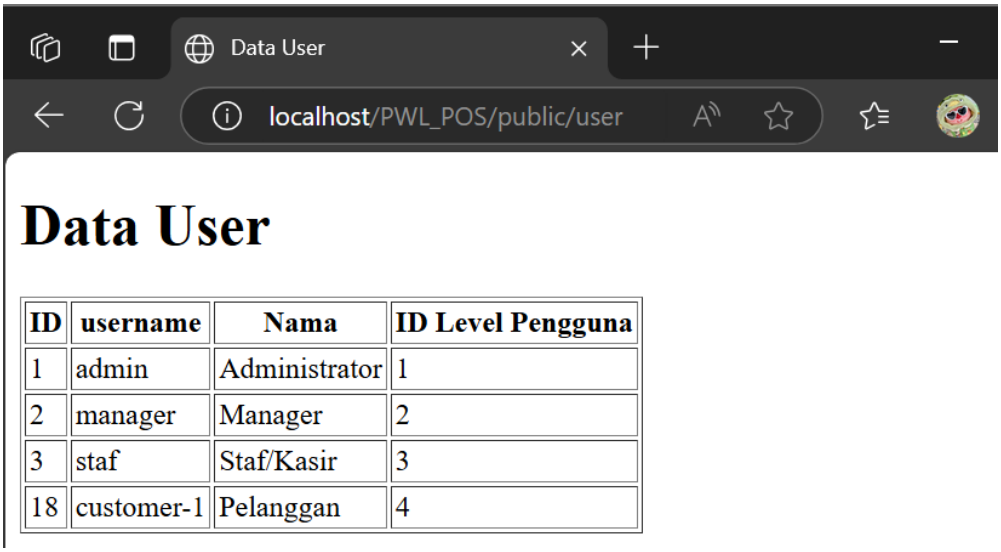
h. Setelah itu, kita modifikasi lagi file **UserController**

app > Http > Controllers > UserController.php > ...

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        // tambah data user dengan Eloquent Model
        $data = [
            'username' => 'customer-1',
            'nama' => 'Pelanggan',
            'password' => Hash::make('12345'),
            'level_id' => 4
        ];
        UserModel::insert($data); // tambahkan data ke tabel m_user

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

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

Menambah data user



Data User

ID	username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staf	Staf/Kasir	3
18	customer-1	Pelanggan	4

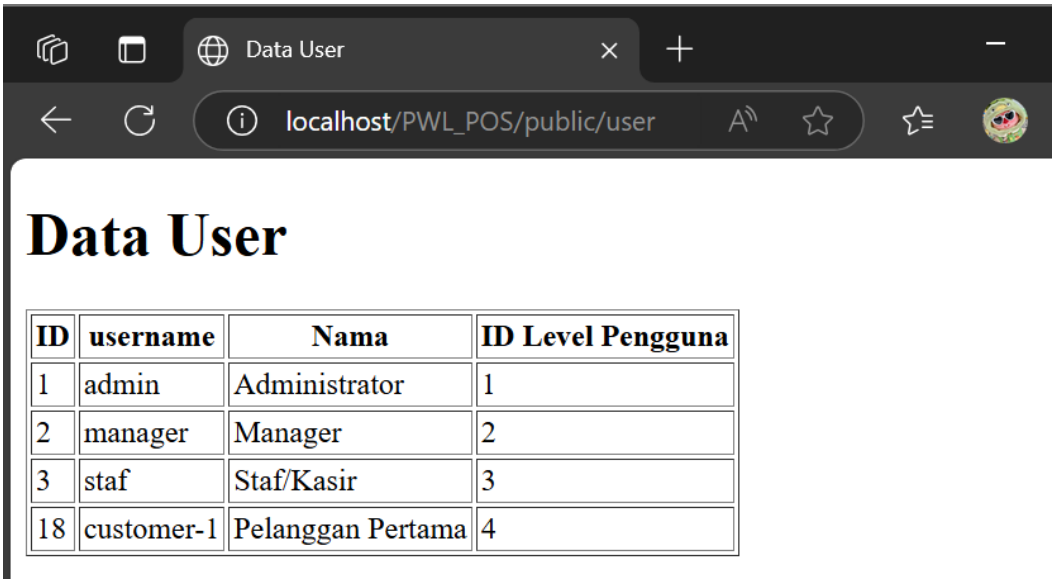
j. Kita modifikasi lagi file **UserController** menjadi seperti berikut

app > Http > Controllers > UserController.php > ...

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        // tambah data user dengan Eloquent Model
        $data = [
            'nama' => 'Pelanggan Pertama',
        ];
        UserModel::where('username', 'customer-1')->update($data); // update data user

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

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

Update data User



Data User

ID	username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staf	Staf/Kasir	3
18	customer-1	Pelanggan Pertama	4

l. 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?
   Jawaban:
   APP_KEY digunakan untuk **enkripsi data sensitif** dalam Laravel, seperti session, token autentikasi, dan password. Nilai ini digunakan oleh Laravel untuk mengenkripsi serta mendekripsi data yang tersimpan dengan aman.
2. Pada **Praktikum 1**, bagaimana kita men-generate nilai untuk APP_KEY?
   Jawaban:
   Nilai *APP_KEY* dapat di-generate menggunakan perintah berikut di terminal:
   php artisan key:generate
   Perintah ini akan mengupdate nilai APP_KEY di file **.env**.
3. Pada **Praktikum 2.1 - Tahap 1**, secara default Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?
   Jawaban:
   Secara default, Laravel memiliki **tiga file migrasi**, yaitu:
   - **create_users_table.php** → Membuat tabel users.
   - **create_password_resets_table.php** → Membuat tabel password_resets untuk fitur reset password.
   - **create_failed_jobs_table.php** → Membuat tabel failed_jobs untuk menyimpan data job yang gagal dalam queue Laravel.
4. Secara default, file migrasi terdapat kode $table->timestamps();, apa tujuan*/output* dari fungsi tersebut?
   Jawaban:
   Fungsi $table->timestamps(); secara otomatis menambahkan **dua kolom**:
   - **created_at** → Menyimpan waktu saat data dibuat.
   - **updated_at** → Menyimpan waktu saat data terakhir diperbarui.
5. Pada File Migrasi, terdapat fungsi $table->id(); Tipe data apa yang dihasilkan dari fungsi tersebut?
   Jawaban:
   Fungsi $table->id(); akan menghasilkan **kolom primary key** dengan tipe data **bigint unsigned** yang bersifat **auto-increment**.
6. Apa bedanya hasil migrasi pada table m_level, antara menggunakan $table->id(); dengan menggunakan $table->id('level_id'); ?
   Jawaban:
   - $table->id(); → Secara default membuat kolom **id** sebagai **primary key**.
   - $table->id('level_id'); → Membuat kolom **level_id** sebagai **primary key**, sesuai dengan kebutuhan tabel.
7. Pada migration, Fungsi ->unique() digunakan untuk apa?
   Jawaban:
   Fungsi ->unique() digunakan untuk **mencegah duplikasi nilai dalam suatu kolom**.

8. Pada **Praktikum 2.2 - Tahap 2**, kenapa kolom level_id pada tabel m_user menggunakan $tabel->unsignedBigInteger('level_id'), sedangkan kolom level_id pada tabel m_level menggunakan $tabel->id('level_id') ?
   Jawaban:
   - Pada tabel **m_level**, level_id dibuat dengan $table->id('level_id'); yang berarti **primary key** dengan tipe **bigint unsigned**.
   - Pada tabel **m_user**, level_id harus disesuaikan dengan tipe primary key di m_level, sehingga menggunakan $table->unsignedBigInteger('level_id'); agar bisa menjadi **foreign key** yang sesuai.
9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class Hash? dan apa maksud dari kode program Hash::make('1234');?
   Jawaban:
   - Class Hash digunakan untuk **mengenkripsi password** sebelum disimpan ke database.
   - Hash::make('1234'); → Mengubah string '1234' menjadi **hash bcrypt** yang tidak bisa dikembalikan ke bentuk aslinya.
10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya *(?),* apa kegunaan dari tanda tanya (?) tersebut?
    Jawaban:
    Tanda tanya (?) digunakan untuk **binding parameter** dalam query builder, sehingga:
    - **Mencegah SQL Injection**
    - **Memudahkan pembacaan query**
11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode protected $table = ‘m_user’; dan protected $primaryKey = ‘user_id’; ?
    Jawaban:
    - protected $table = 'm_user'; → Menentukan bahwa model ini terkait dengan tabel m_user.
    - protected $primaryKey = 'user_id'; → Menentukan bahwa **primary key** pada tabel ini adalah user_id, bukan id (default Laravel).
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database *(DB Façade / Query Builder / Eloquent ORM)* ? jelaskan
    Jawaban:
    Eloquent ORM lebih mudah digunakan untuk operasi CRUD karena:
    - **Lebih simpel** → Tidak perlu menulis SQL secara manual.
    - **Lebih aman** → Menggunakan fitur bawaan Laravel seperti mass assignment protection.
    - **Lebih terstruktur** → Data diperlakukan sebagai **objek**, bukan array hasil query.