**LAPORAN PRAKTIKUM**

**MATA KULIAH PEMROGRAMAN WEB LANJUT**

Dosen Pengampu : Dimas Wahyu Wibowo, ST., MT.

**JOBSHEET 3: MIGRATION, SEEDER, DB FASCADE, QUERY BUILDER, dan ELOQUENT ORM**

****

Nama : Yonanda Mayla Rusdiaty

NIM : 2341760184

Prodi : D-IV Sistem Informasi Bisnis

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2024**

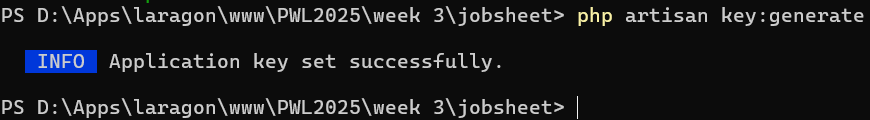
1. **PENGATURAN DATABASE**

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:**
2. Buka aplikasi phpMyAdmin, dan buat database baru dengan nama PWL\_POS

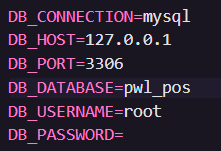


1. Buka aplikasi VSCode dan buka folder project PWL\_POS yang sudah kita buat
2. Copy file .env.example menjadi .env
3. Buka file .env, dan pastikan konfigurasi APP\_KEY bernilai. Jika belum bernilai silahkan kalian generate menggunakan php artisan





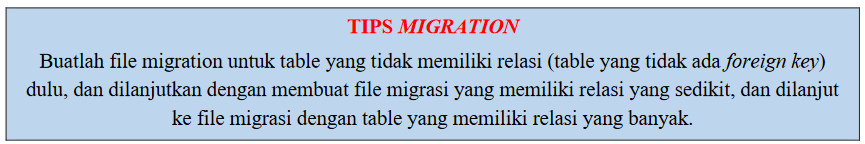
1. Edit file .env dan sesuaikan dengan database yang telah dibuat



1. Laporkan hasil Praktikum-1 ini dan commit perubahan pada git
2. **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

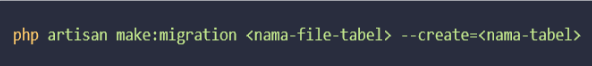
Salah satu keunggulan menggunakan migration adalah mempermudah proses instalasi aplikasi kita, Ketika aplikasi yang kita buat akan diimplementasikan di server/komputer lain





Pembuatan file migrasi bisa menggunakan 2 cara, yaitu:

1. Menggunakan artisan untuk membuat file migration

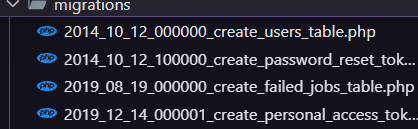


1. Menggunakan artisan untuk membuat file model + file migration



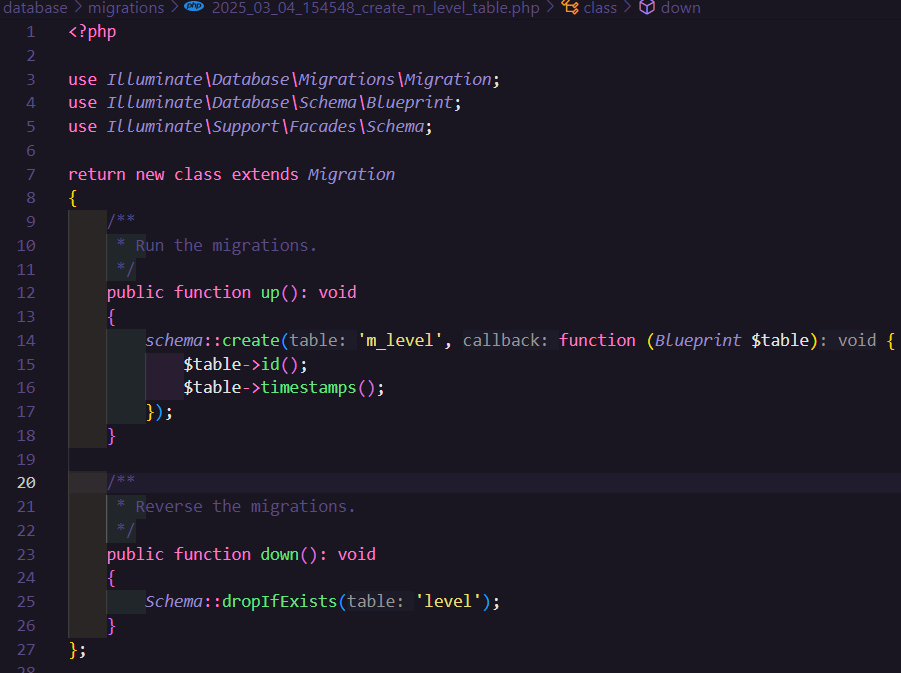
Perintah -m di atas adalah shorthand untuk opsi membuat file migrasi berdasarkan model yang dibuat.

1. **Praktikum 2.1 – Pembuatan File Migrasi Tanpa Relasi**
2. Buka terminal VSCode kalian, untuk yang di kotak merah adalah default dari laravel

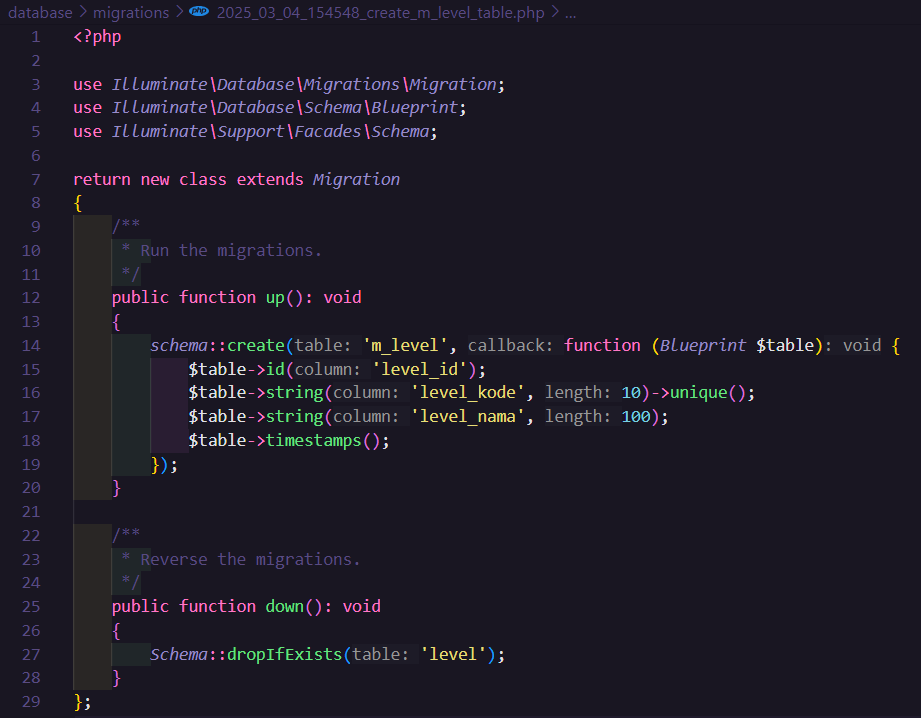


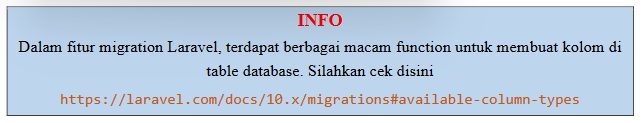
1. Kita abaikan dulu yang di kotak merah (jangan di hapus)
2. Kita buat file migrasi untuk table m\_level dengan perintah



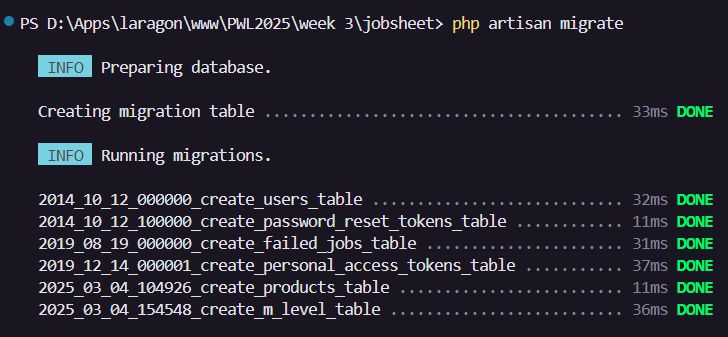


1. Kita perhatikan bagian yang di kotak merah, bagian tersebut yang akan kita modifikasi sesuai desain database yang sudah ada

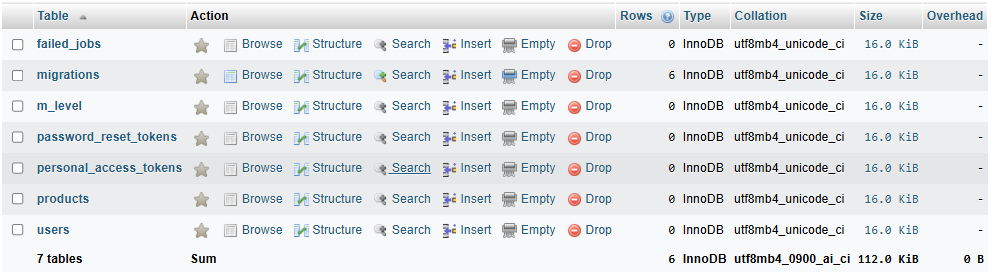




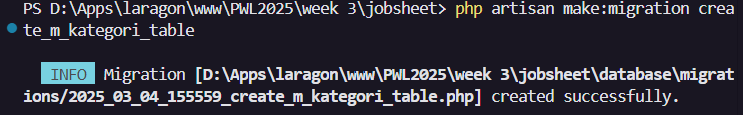
1. Simpan kode pada tahapan 4 tersebut, kemudian jalankan perintah ini pada terminal VSCode untuk melakukan migrasi

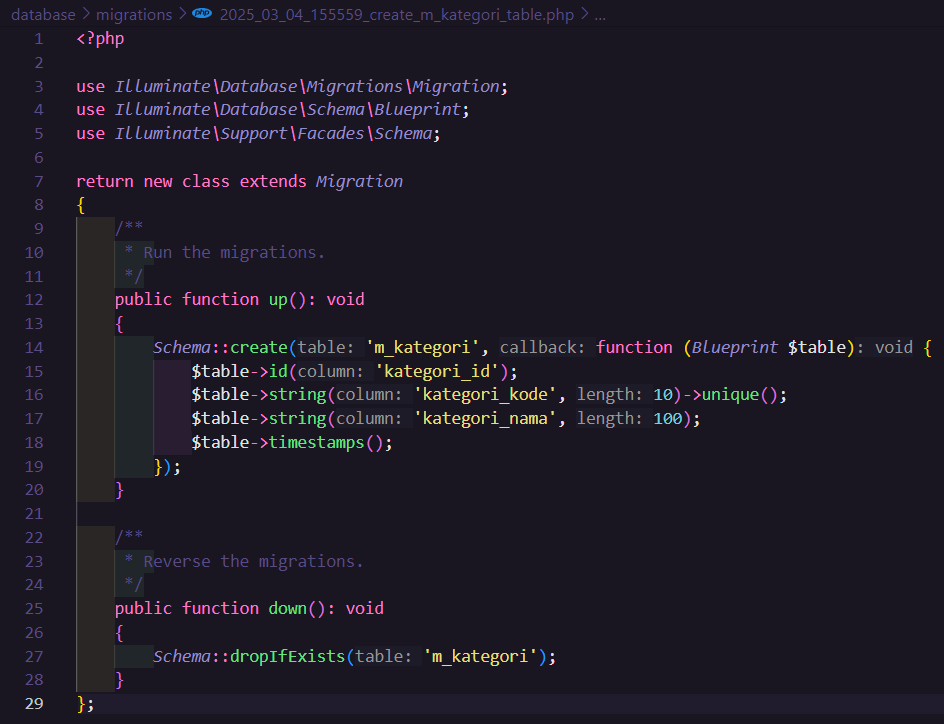


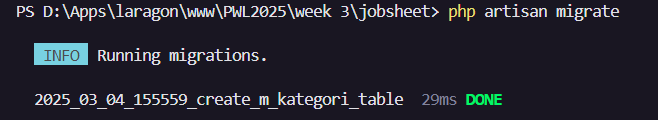
1. Kemudian kita cek di phpMyAdmin apakah table sudah ter-generate atau belum

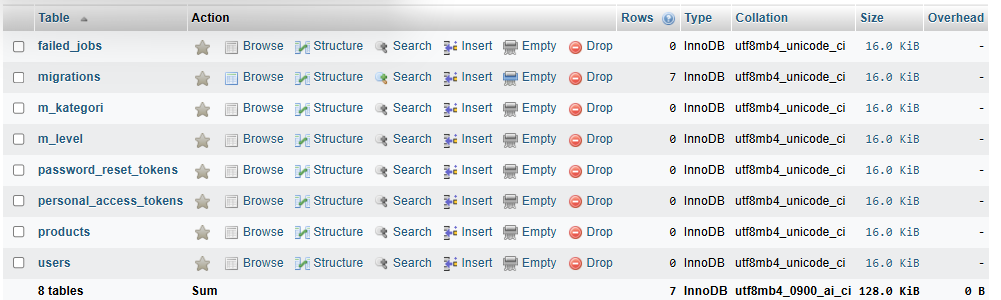


1. Ok, table sudah dibuat di database
2. Buat table database dengan migration untuk table m\_kategori yang sama-sama tidak memiliki foreign key

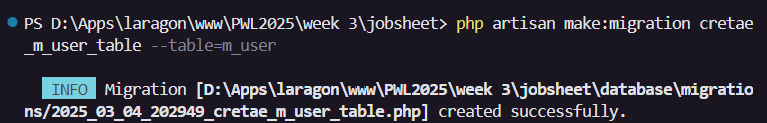




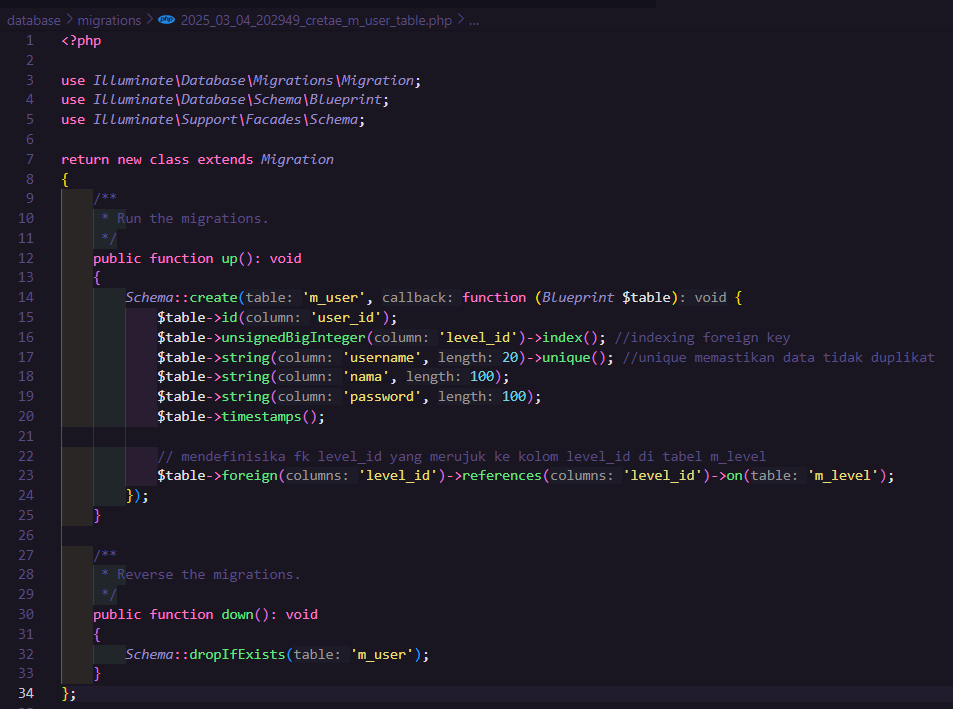




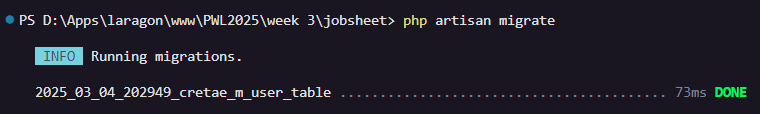
1. Laporkan hasil Praktikum-2.1 ini dan commit perubahan pada git.
2. **Praktikum 2.2 – Pembuatan File Migrasi dengan Relasi**
3. Buka terminal VSCode kalian, dan buat file migrasi untuk table m\_user



1. Buka file migrasi untuk table m\_user, dan modifikasi seperti berikut



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

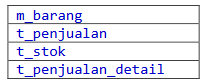


**Jawab:**

1. Fungsi up digunakan untuk mendefinisikan perubahan yang akan diterapkan pada database saat migration dijalankan. Dalam contoh ini, fungsi up membuat tabel m\_user dengan kolom-kolom berikut:

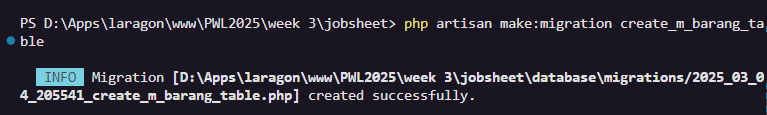
* user\_id: Kolom ID utama dengan tipe data bigint yang otomatis bertambah (auto\_increment).
* level\_id: Kolom unsignedBigInteger yang digunakan sebagai foreign key dan diindeks.
* username: Kolom string dengan panjang maksimal 20 karakter yang harus unik.
* nama: Kolom string dengan panjang maksimal 100 karakter.
* password: Kolom string dengan panjang maksimal 100 karakter.
* timestamps: Kolom created\_at dan updated\_at yang otomatis diisi oleh Laravel.
* Fungsi up juga mendefinisikan foreign key level\_id yg merujuk pada kolom level\_id di tabel m\_level

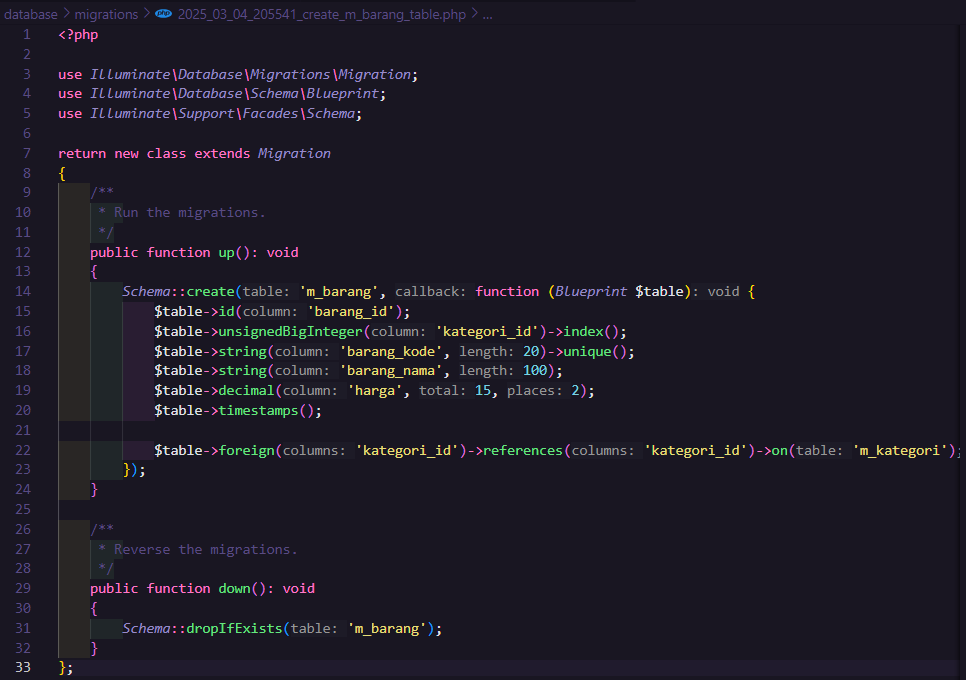
1. Fungsi `down` digunakan untuk mendefinisikan perubahan yang akan diterapkan pada database saat migration dibatalkan (rollback). Dalam contoh ini, fungsi `down` menghapus tabel `m\_user` jika ada.
2. Buat table database dengan migration untuk table-tabel yang memiliki foreign key

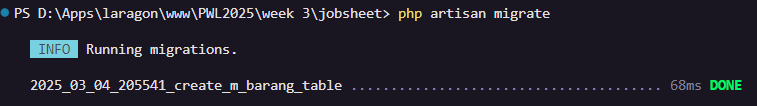


**Jawab:**

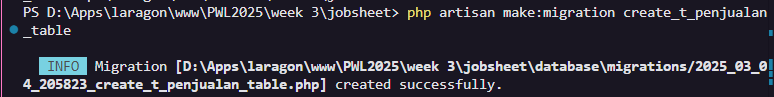
1. M\_barang

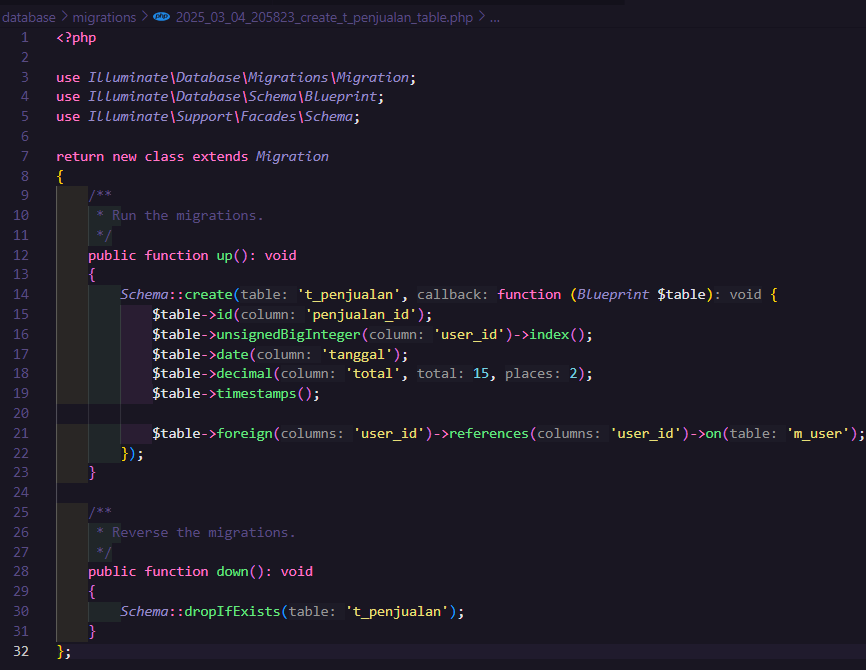


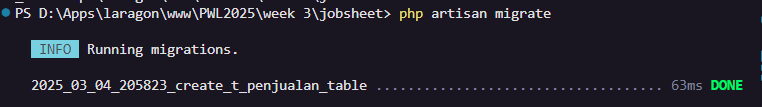




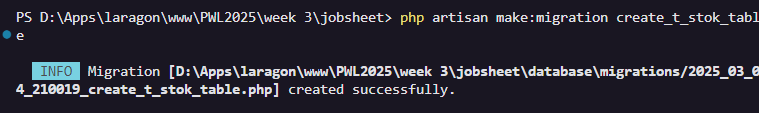
1. T\_penjualan

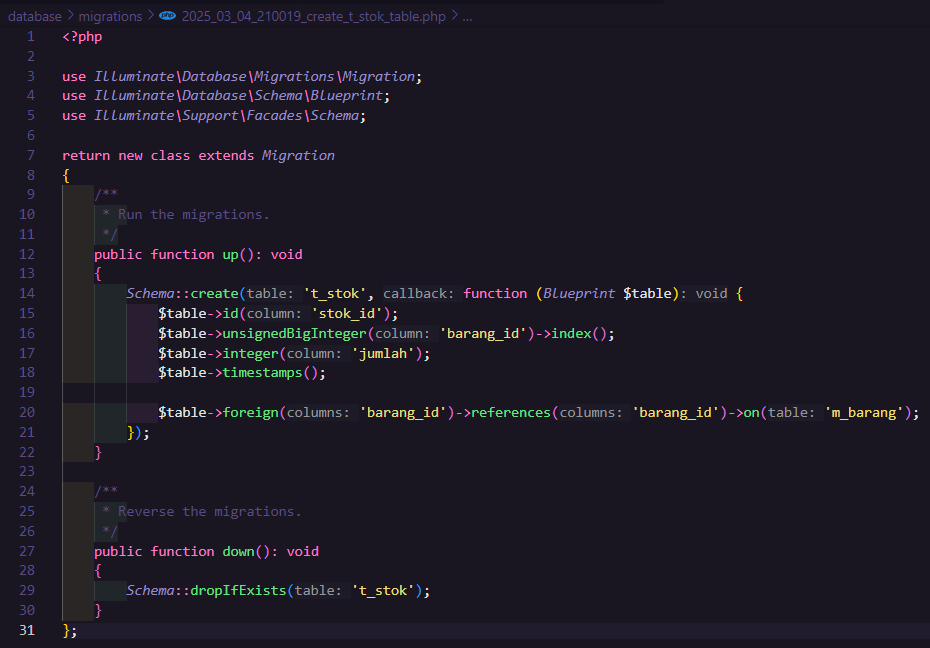


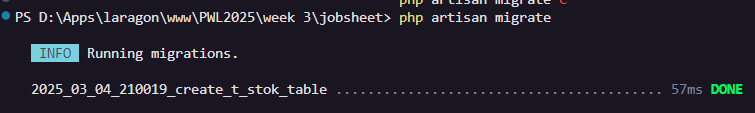




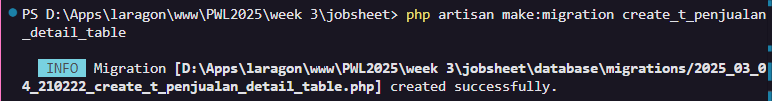
1. T\_stok

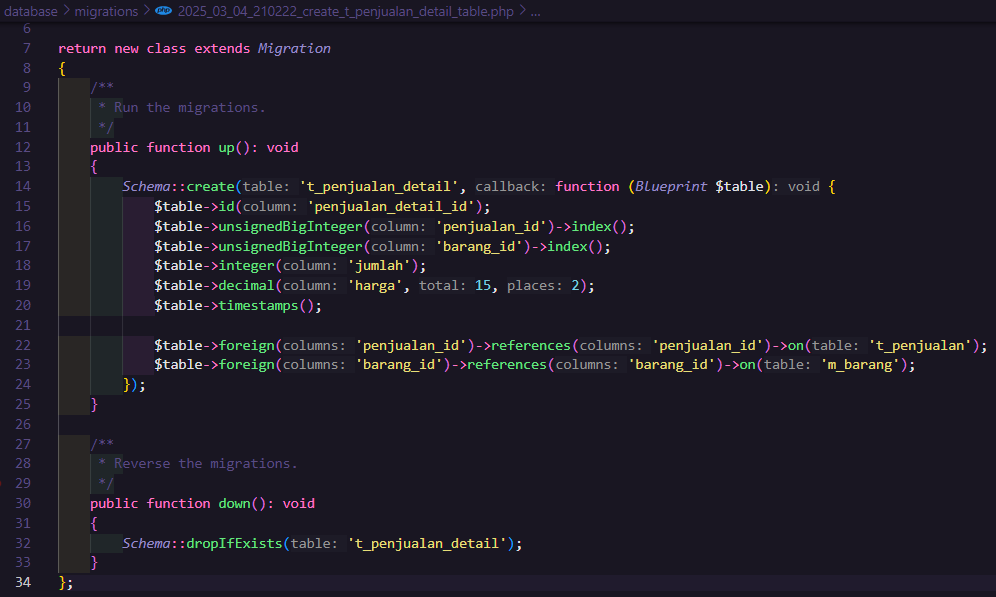


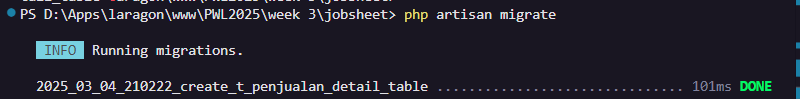




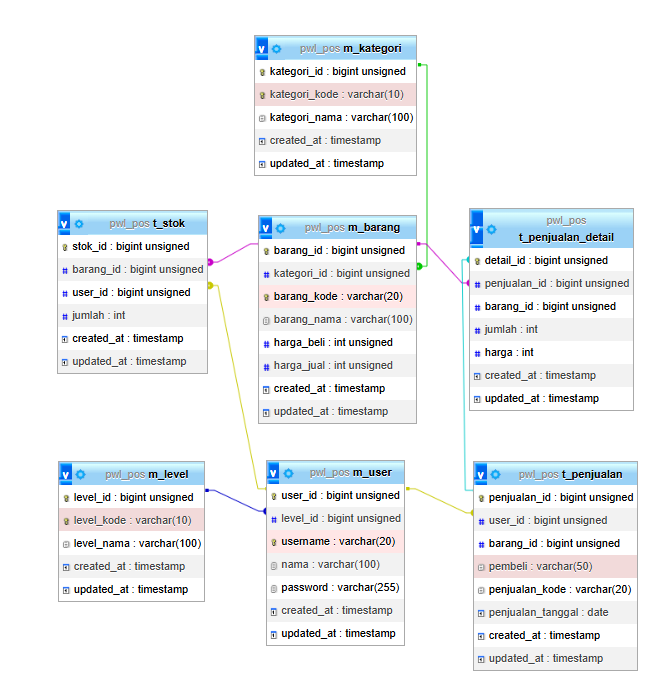
1. T\_penjualan\_detail







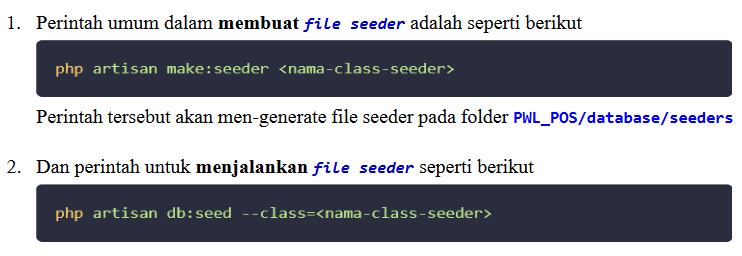
1. Jika semua file migrasi sudah di buat dan dijalankan maka bisa kita lihat tampilan designer pada phpMyAdmin seperti berikut



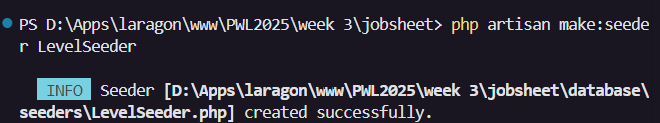
1. Laporkan hasil Praktikum-2.2 ini dan commit perubahan pada git.
2. **SEEDER**

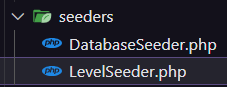
Seeder merupakan sebuah fitur yang memungkinkan kita untuk mengisi database kita

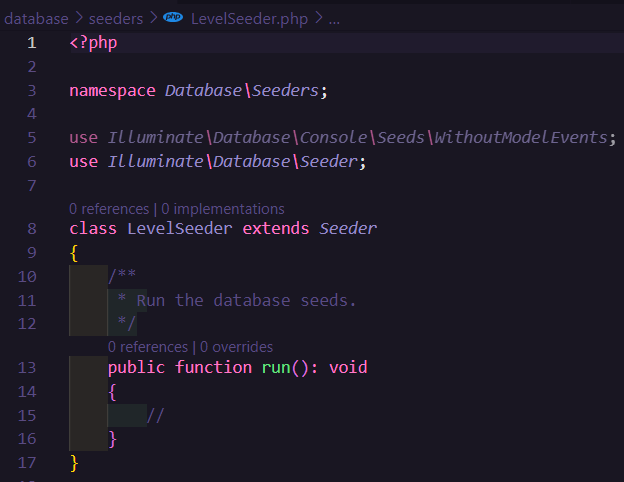
dengan data awal atau data dummy yang telah ditentukan. Seeder memungkinkan kita untuk membuat data awal yang sama untuk setiap penggunaan dalam pembangunan aplikasi. Umumnya, data yang sering dibuat seeder adalah data penggunna karena data tersebut akan digunakan saat aplikasi pertama kali di jalankan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



1. **Praktikum 3 – Membuat File Seeder**
2. Kita akan membuat file seeder untuk table m\_level dengan mengetikkan perintah



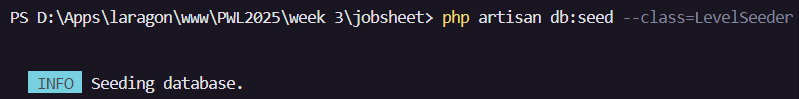




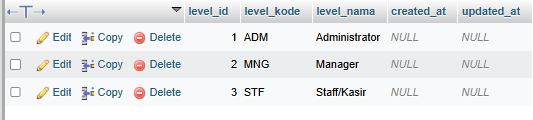
1. Selanjutnya, untuk memasukkan data awal, kita modifikasi file tersebut di dalam function run()



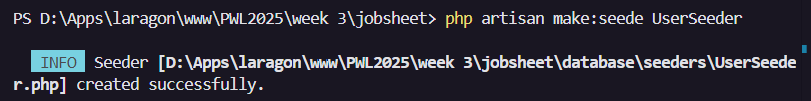
1. Selanjutnya, kita jalankan file seeder untuk table m\_level pada terminal



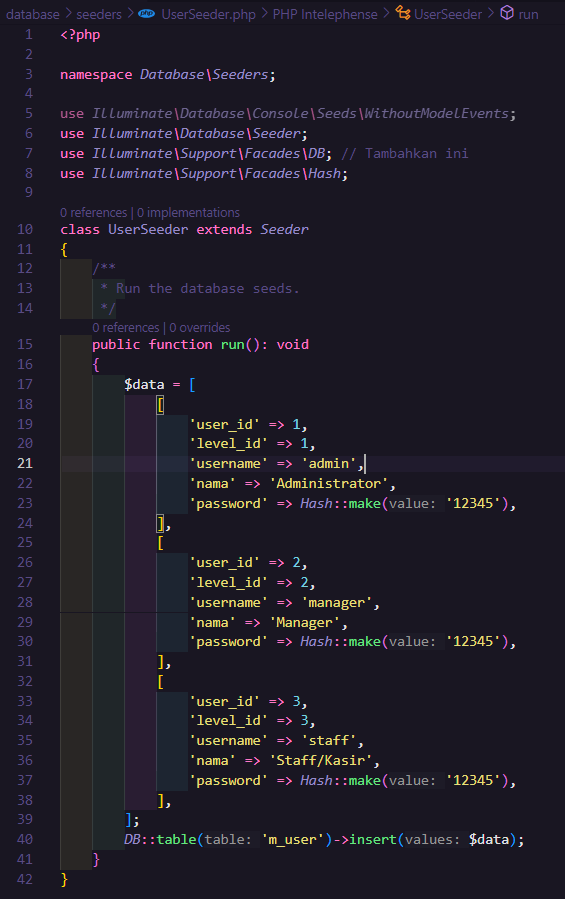
1. Ketika seeder berhasil dijalankan maka akan tampil data pada table m\_level



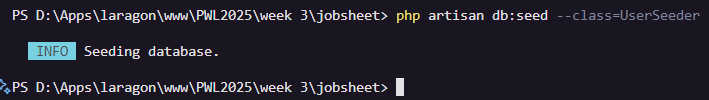
1. Sekarang kita buat file seeder untuk table m\_user yang me-refer ke table m\_level



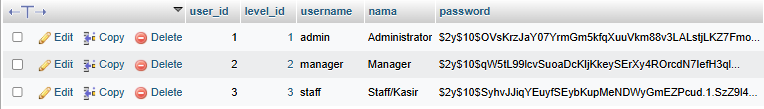
1. Modifikasi file class UserSeeder seperti berikut



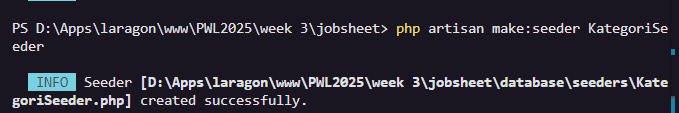
1. Jalankan perintah untuk mengeksekusi class UserSeeder



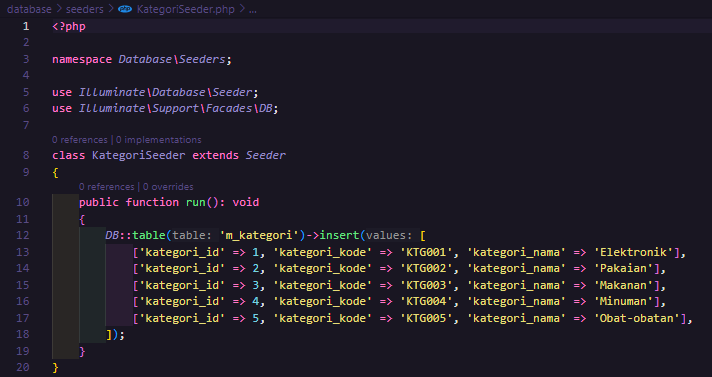
1. Perhatikan hasil seeder pada table m\_user



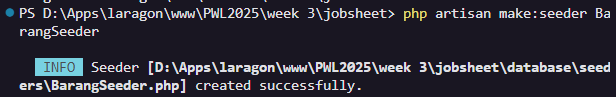
1. Ok, data seeder berhasil di masukkan ke database.
2. Sekarang coba kalian masukkan data seeder untuk table yang lain, dengan ketentuan seperti berikut
3. Seeder untuk kategori (5 kategori barang)



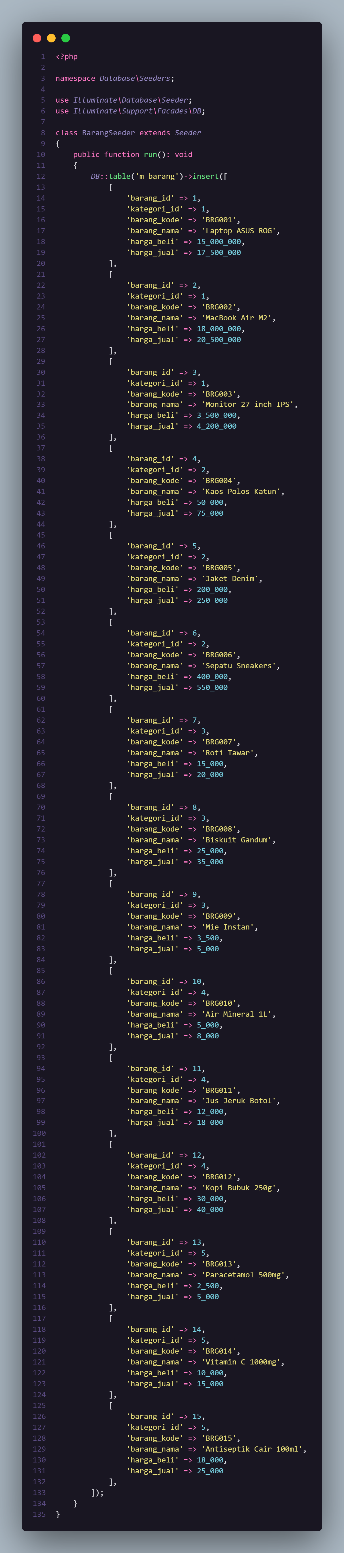
1. Buka database/seeders/KategoriSeeder.php dan tambahkan:



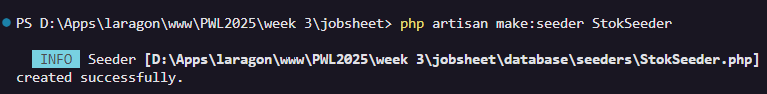
1. Seeder untuk m\_barang (10 Barang Berbeda)



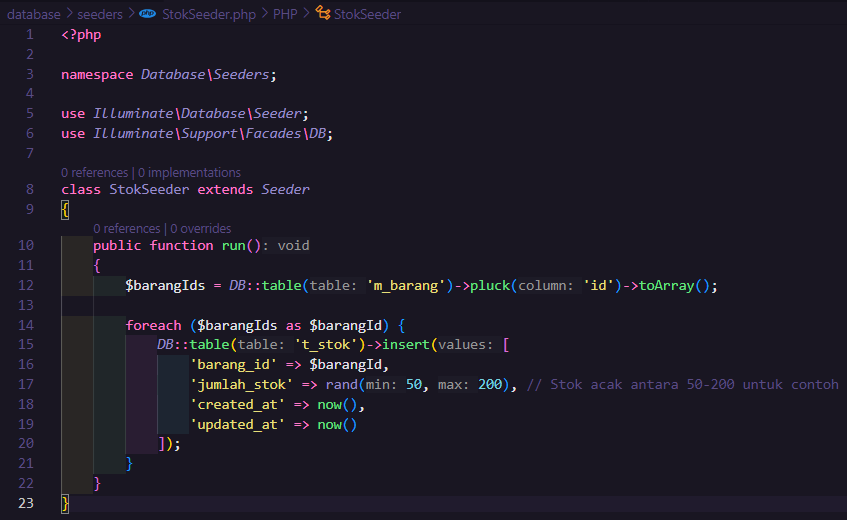
1. Buka database/seeders/BarangSeeder.php dan tambahkan:



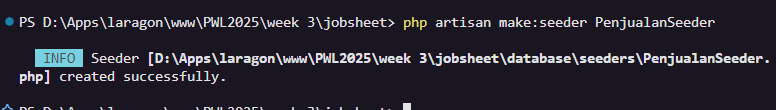
1. Seeder untuk t\_stok (10 Stok untuk 10 Barang)



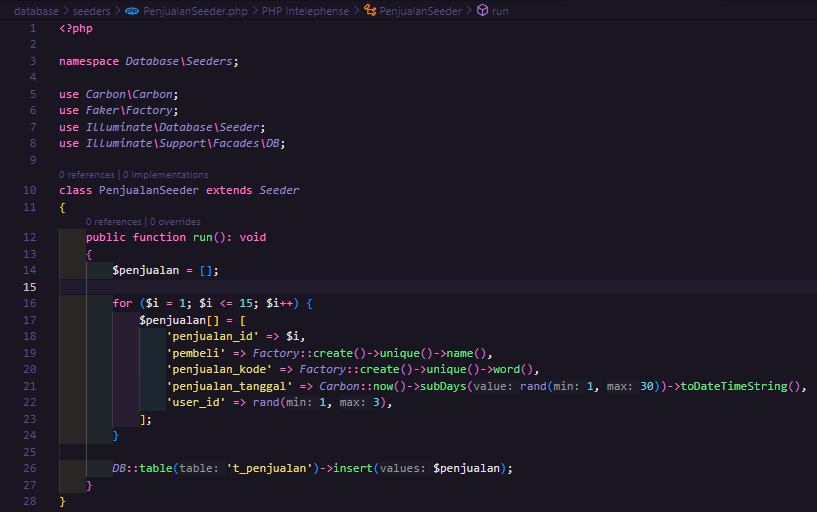
1. Buka database/seeders/StokSeeder.php dan tambahkan:



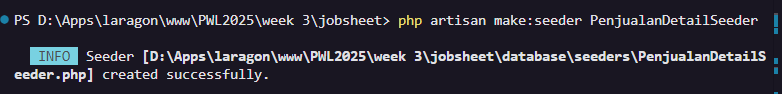
1. Seeder untuk t\_penjualan (10 Transaksi Penjualan)



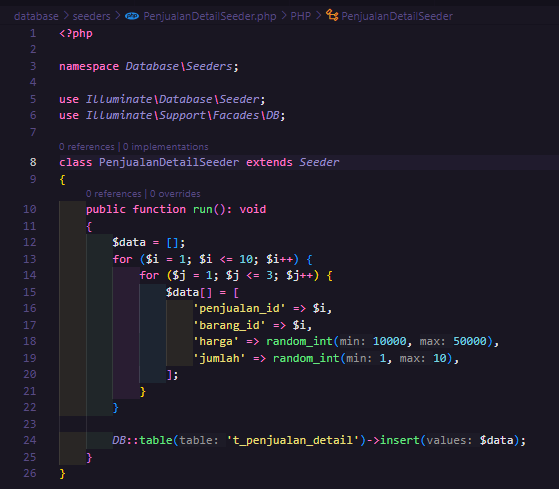
1. Buka database/seeders/PenjualanSeeder.php dan tambahkan:



1. Seeder untuk t\_penjualan\_detail (30 Entri, 3 Barang per Transaksi)



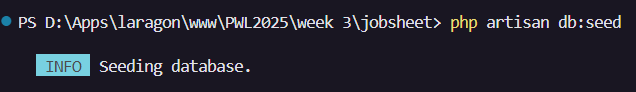
1. Buka database/seeders/PenjualanDetailSeeder.php dan tambahkan:



1. Buka database seeders/LevelSeeder.php dan tambahkan:



1. Menjalankan Seeder

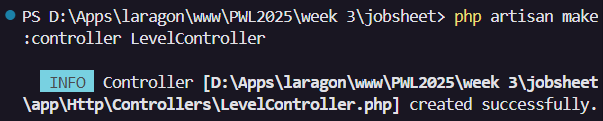


1. **DB FACADES**

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 …”



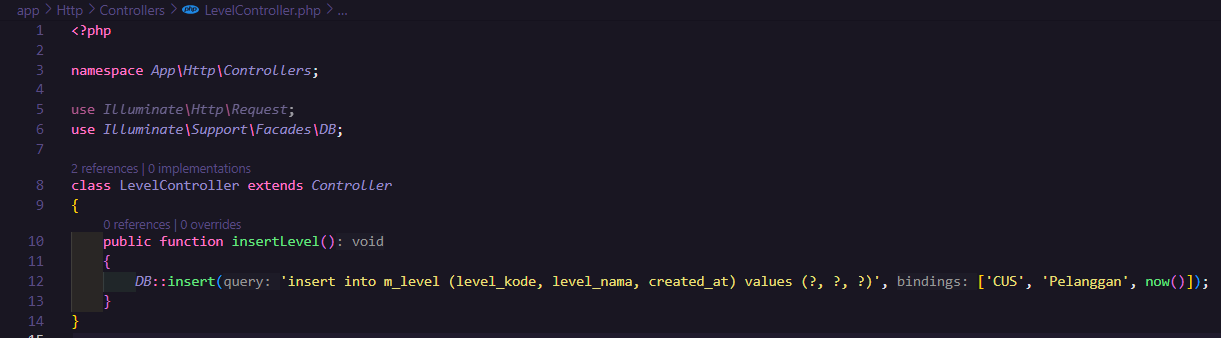
1. **Praktikum 4 – Implementasi DB Façade**
2. Kita buat controller dahulu untuk mengelola data pada table m\_level



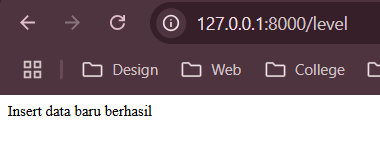
1. Kita modifikasi dulu untuk routing-nya, ada di PWL\_POS/routes/web.php



1. Selanjutnya, kita modifikasi file LevelController untuk menambahkan 1 data ke table m\_level

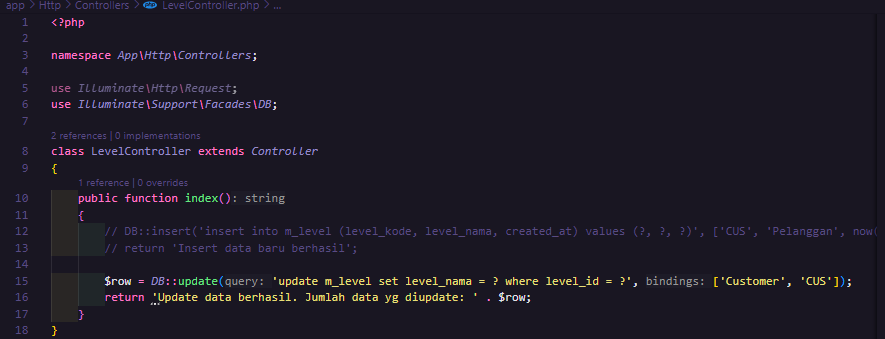


1. Kita coba jalankan di browser dengan url localhost/PWL\_POS/public/level dan amati apa yang terjadi pada table m\_level di database, screenshot perubahan yang ada pada table m\_level

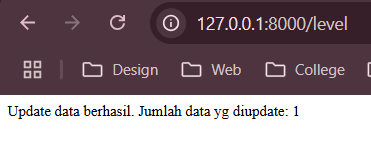


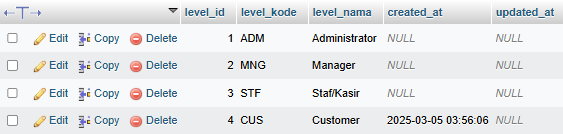


1. Selanjutnya, kita modifikasi lagi file LevelController untuk meng-update data di table m\_level seperti berikut

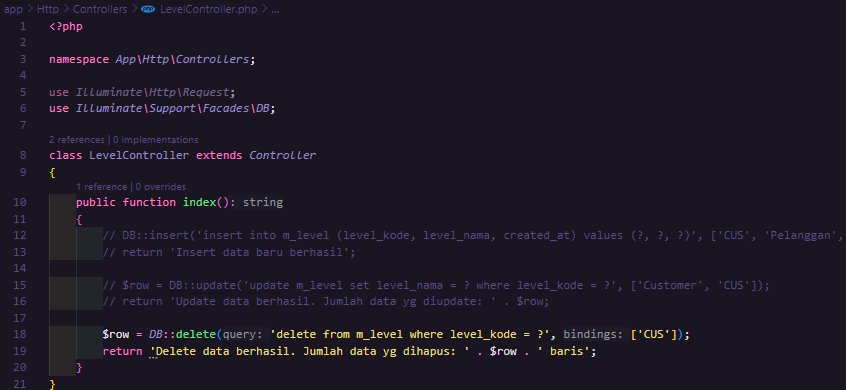


1. Kita coba jalankan di browser dengan url localhost/PWL\_POS/public/level lagi dan amati apa yang terjadi pada table m\_level di database, screenshot perubahan yang ada pada table m\_level

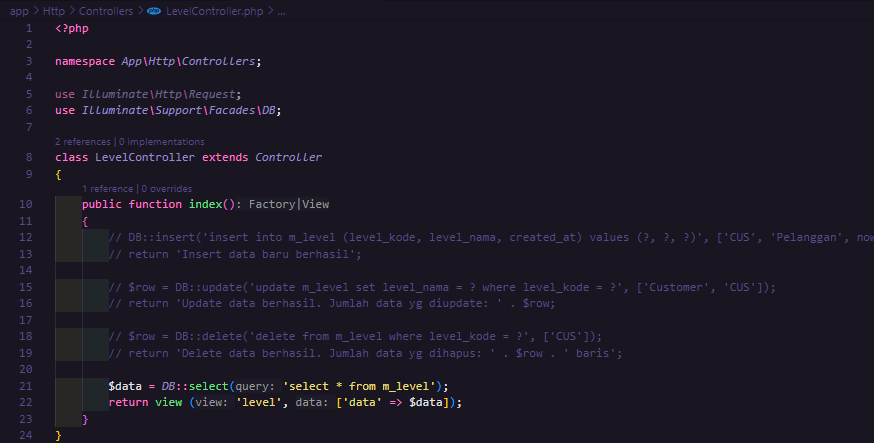




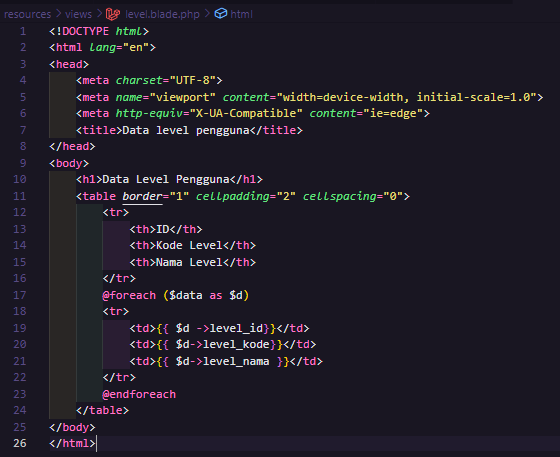
1. Kita coba modifikasi lagi file LevelController untuk melakukan proses hapus data



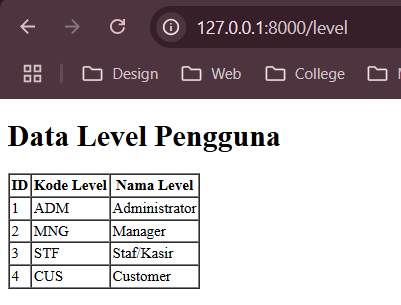
1. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table m\_level. Kita modifikasi file LevelController seperti berikut



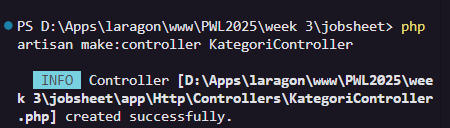
1. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil view(‘level’), maka kita buat file view pada VSCode di PWL\_POS/resources/view/level.blade.php



1. Silahkan dicoba pada browser dan amati apa yang terjadi

  
**Penjelasan:** level.blade digunakan untuk menampilkan data level pengguna dalam bentuk tabel. Data yang ditampilkan berasal dari variabel $data yang dikirim dari controller. Setiap item dalam $data di-looping dan ditampilkan dalam baris tabel.

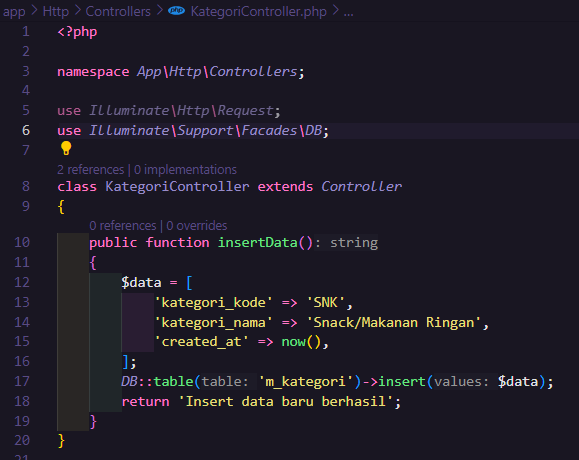
1. Laporkan hasil Praktikum-4 ini dan commit perubahan pada git.
2. **QUERY BUILDER**
3. **Praktikum 5 - Implementasi Query Builder**
4. Kita buat controller dahuku untuk mengelola data pada table m\_kategori



1. Kita modifikasi dulu untuk routing-nya, ada di PWL\_POS/routes/web.php

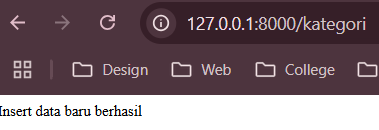


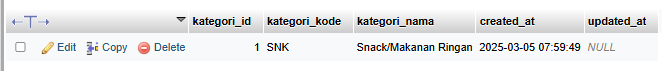
1. Selanjutnya, kita modifikasi file KategoriController untuk menambahkan 1 data ke table m\_kategori



1. 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

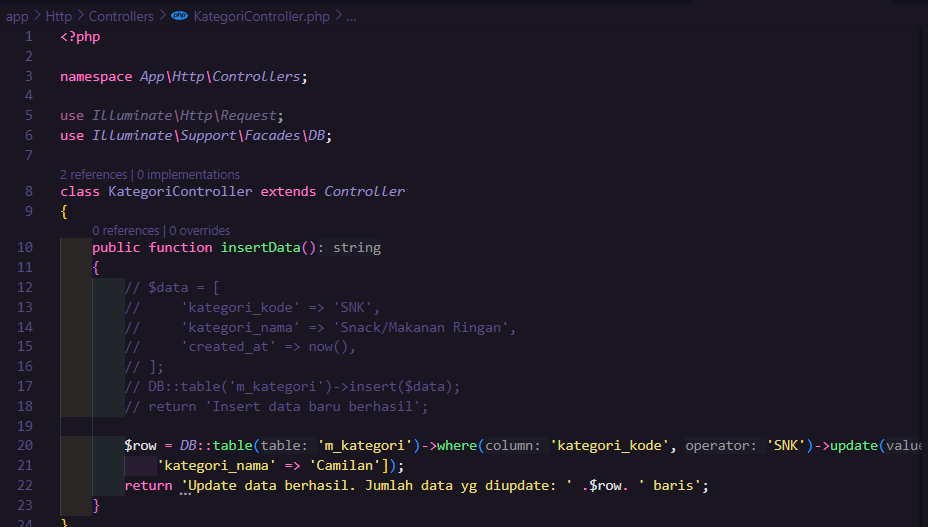
**Jawab:**

****

****

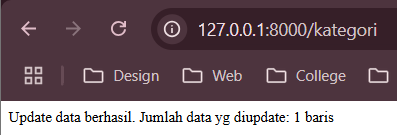
Controller ini digunakan untuk memasukkan data baru ke dalam tabel m\_kategori melalui metode index. Saat metode index dipanggil, data baru akan dimasukkan ke dalam tabel m\_kategori, dan pesan sukses akan dikembalikan.

1. Selanjutnya, kita modifikasi lagi file KategoriController untuk meng-update data di table m\_kategori seperti berikut



1. 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

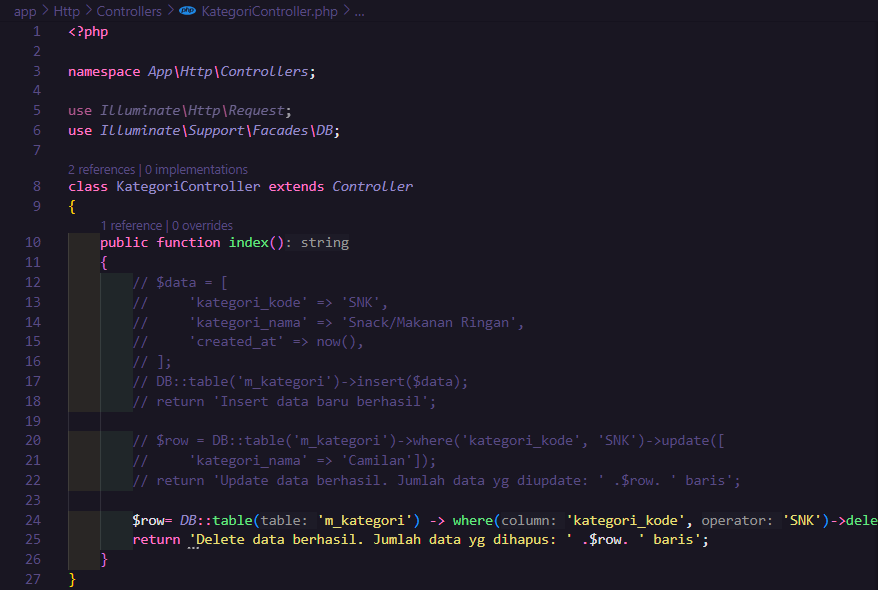
**Jawab:**

****

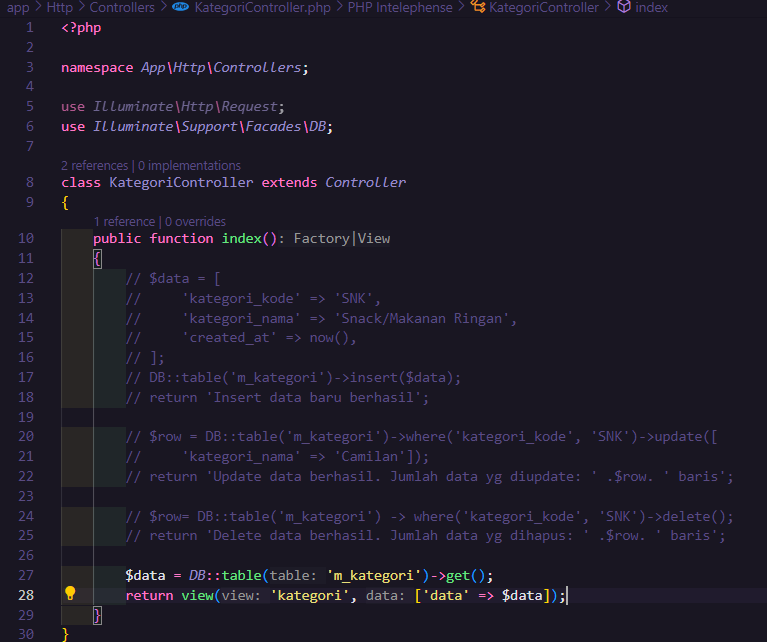
****

Saat metode index dipanggil, kolom kategori\_nama untuk baris yang memiliki kategori\_kode 'SNK' akan diupdate menjadi 'Camilan', dan pesan sukses akan dikembalikan dengan jumlah baris yang diupdate. Sebelumnya, di phpMyAdmin, kolom kategori\_nama untuk kategori\_kode 'SNK' berisi 'Snack/Makanan Ringan', dan setelah menjalankan metode ini, kolom tersebut akan diupdate menjadi 'Camilan'.

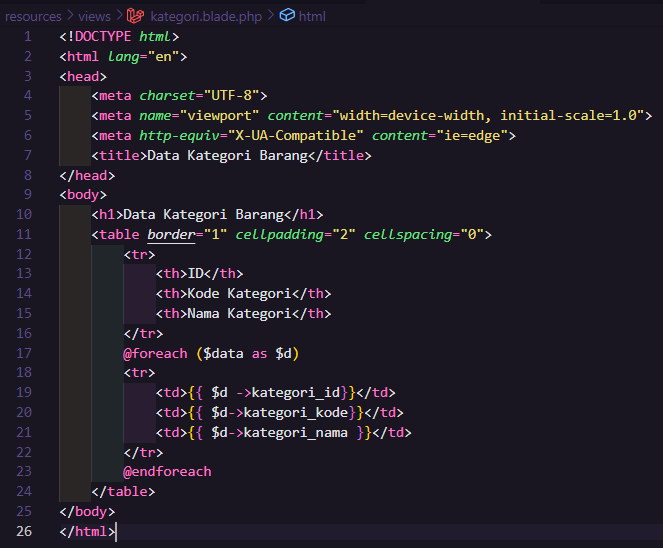
1. Kita coba modifikasi lagi file KategoriController untuk melakukan proses hapus data



1. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table m\_kategori. Kita modifikasi file KategoriController seperti berikut

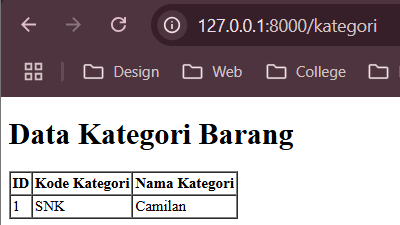


1. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil view(‘kategori’), maka kita buat file view pada VSCode di PWL\_POS/resources/view/kategori.blade.php



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

**Jawab:**



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