Pemrograman Web Lanjut

Jobsheet 3 - Migration, Seeder, DB Facade, Query Builder, dan Eloquent Orm

Dosen Pengampu : Dimas Wahyu Wibowo, S.T., M.T.

Nama : Athallah Ayudya Paramesti
NIM : 2341760061
Program Studi : D-IV Sistem Informasi Bisnis

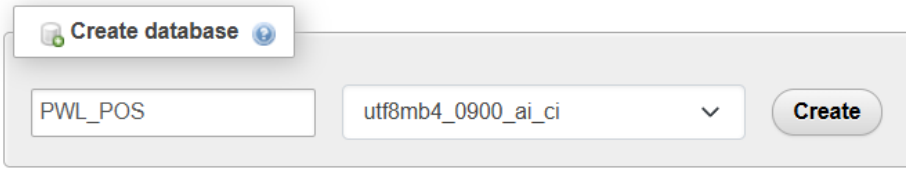
Jurusan Teknologi Informasi
Politeknik Negeri Malang
2025

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.

Praktikum 1 - pengaturan databases:

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.

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:LkSZloF1EO543KdQhwp8R9dah6jxQpO4gx69r0OFtQw=
APP_DEBUG=true
APP_URL=http://localhost
```

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

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

B. Migration

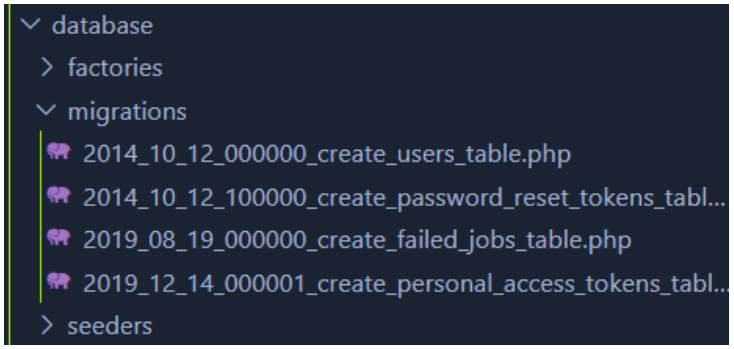
Migration pada Laravel merupakan sebuah fitur yang dapat membantu kita mengelola database secara efisien dengan menggunakan kode program. Migration membantu kita dalam membuat (create), mengubah (edit), dan menghapus (delete) struktur tabel dan kolom pada database yang sudah kita buat dengan cepat dan 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.

Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

1. Ini merupakan default dari laravel



2. Buat file migrasi untuk table m_level dengan perintah berikut:

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

Week_3 > Jobsheet3 > database > migrations > 2025_03_05_162357_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();
        });
    }

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

```

3. modifikasi sesuai desain database yang sudah ada

```php
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();
        });
    }
```

4. jalankan perintah berikut untuk melakukan migrasi

```
λ php artisan migrate
```

```
  INFO  Preparing database.

Creating migration table ..................................... 3,175ms DONE

  INFO  Running migrations.

2014_10_12_000000_create_users_table ......................... 1,486ms DONE
2014_10_12_100000_create_password_reset_tokens_table ......... 165ms DONE
2019_08_19_000000_create_failed_jobs_table ................... 665ms DONE
2019_12_14_000001_create_personal_access_tokens_table ........ 556ms DONE
2025_03_05_162357_create_m_level_table ....................... 287ms DONE
```

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

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	migrations	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	m_level	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	password_reset_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	personal_access_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	users	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
	6 tables	Sum	5	InnoDB	utf8mb4_0900_ai_ci	96.0 KiB	0 B

6. Ok, table sudah dibuat di database
7. Buat table database dengan migration untuk table m_kategori yang sama-sama tidak memiliki foreign key

```
λ php artisan make:migration create_m_kategori_table
```

Week_3 > Jobsheet3 > database > migrations > 2025_03_05_204822_create_m_kategori_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_kategori', function (Blueprint $table) {
            $table->id('kategori_id');
            $table->string('kategori_kode', 10)->unique();
            $table->string('kategori_nama', 100);
            $table->timestamps();
        });
    }

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

```
λ php artisan migrate
```

```
  INFO  Running migrations.

  2025_03_05_204822_create_m_kategori_table ..... 6,739ms DONE
```

Table
failed_jobs
migrations
m_kategori
m_level
password_reset_tokens
personal_access_tokens
users

Praktikum 2.2 - Pembuatan file migrasi dengan relasi

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

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

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

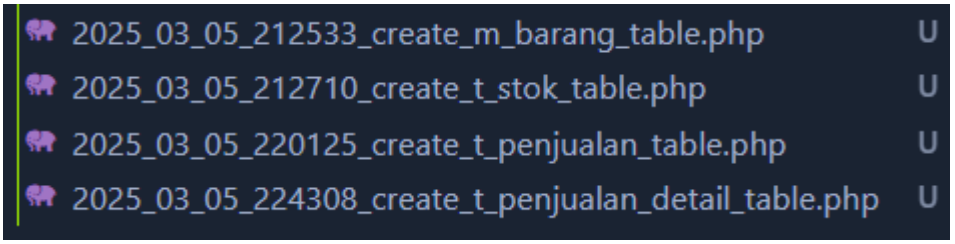
```php
<?php

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::table('m_user', function (Blueprint $table) {
            $table->id('user_id');
            $table->usignedBigInteger('level_id')->index(); //indexing untuk ForeignKey
            $table->string('username', 20)->unique(); //untuk memastikan tidak ada username yang sama
            $table->string('nama', 100);
            $table->string('password');
            $table->timestamps();

            //Mendefinisikan Foreign Key pada kolom leve_id mengacu pada kolom level_id di tabel m_level
            $table->foreign('level_id')->references('level_id')->on('m_level');
        });
    }

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

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

Table
failed_jobs
migrations
m_kategori
m_level
m_user
password_reset_tokens
personal_access_tokens
users

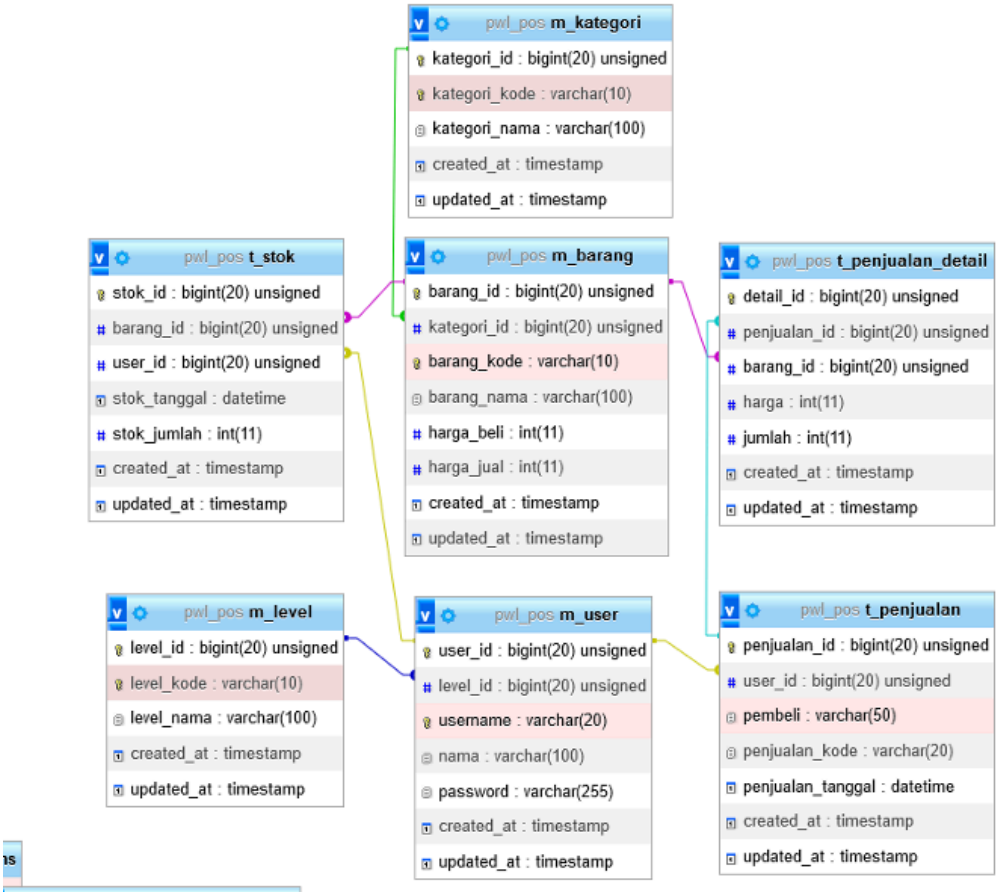
4. Buat tabel database dengan migration untuk tabel-tabel yang memiliki foreign key

m_barang
t_penjualan
t_stok
t_penjualan_detail



isi file sesuai isi tabel, lalu migrasi

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



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 pengguna karena data tersebut akan digunakan saat aplikasi pertama kali dijalankan dan membutuhkan aksi login.

Dalam proses pengembangan suatu aplikasi, seringkali kita membutuhkan data awal tiruan atau dummy data untuk memudahkan pengujian dan pengembangan aplikasi kita. Sehingga fitur seeder bisa kita pakai dalam membuat sebuah aplikasi web.

Praktikum 3 - Membuat file seeder

1. Buat file seeder untuk table m_level dengan mengetikkan perintah

```php
<?php

namespace Database\Seeders;

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

class LevelSeeder extends Seeder{
    public function run(): void{
        //
    }
}
```

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

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

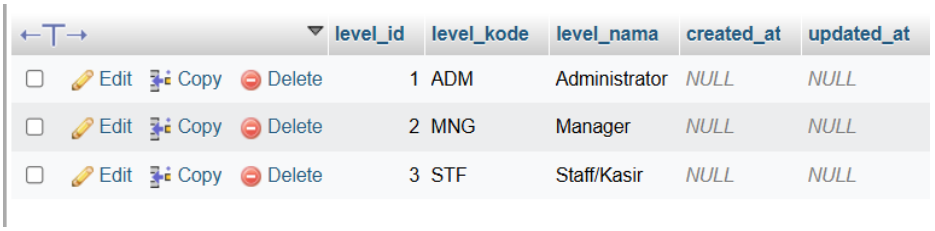
```php
class LevelSeeder extends Seeder{
    public function run(): void{
        $data = [
            ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Admi
            ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Mana
            ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staf
        ];
        DB::table('m_level')->insert($data);
    }
}
```

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

```
λ 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
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL

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

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

```
INFO  Seeder [C:\laragon\www\laravel\PWL_2025\Week_3\Jobsheet3\database\seeders\UserS
successfully.
```

6. Modifikasi file class UserSeeder seperti berikut

```php
public function run(): void
{
    $data =[
        [
            'user_id' => 1,
            'level_id' => 1,
            'username' => 'admin',
            'nama' => 'Administrator',
            'password' => Hash::make('12345'), //untuk enkripsi password
        ],
        [
            'user_id' => 2,
            'level_id' => 2,
            'username' => 'manager',
            'nama' => 'Manager',
            'password' => Hash::make('12345'), //untuk enkripsi password
        ],
        [
            'user_id' => 3,
            'level_id' => 3,
            'username' => 'staff',
            'nama' => 'Staff/Kasir',
            'password' => Hash::make('12345'), //untuk enkripsi password
        ],
    ];
    DB::table('m_user')->insert($data);
}
```

7. Jalankan perintah untuk mengeksekusi class UserSeeder

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

```
INFO  Seeding database.
```

8. Perhatikan hasil seeder pada table m_user

	user_id	level_id	username	nama	password
Edit Copy Delete	1	1	admin	Administrator	$2y$12$diZXcDqSz4CluPPSTy59/.nE3hsMOwxRSrPhxwmErWL...
Edit Copy Delete	2	2	manager	Manager	$2y$12$SH.MKJveHbZkZfYOQtpeMOJQ7dVEtC7eCCJCFvkWhEE.
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$84QbySK//9FZQ8h01bkhzOVrKaKWv2YZO/CU.1ir0fR...

9. Ok, data seeder berhasil dimasukkan 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

```
λ php artisan make:seeder KategoriSeeder
```

```php
DB::table('m_kategori')->insert([
    ['kategori_kode' => 'FOOD', 'kategori_nama' => 'Makanan'],
    ['kategori_kode' => 'DRINK', 'kategori_nama' => 'Minuman'],
    ['kategori_kode' => 'BEAUTY', 'kategori_nama' => 'Kecantikan'],
    ['kategori_kode' => 'HOME', 'kategori_nama' => 'Peralatan Rumah'],
    ['kategori_kode' => 'BABY', 'kategori_nama' => 'Perlengkapan Bayi'],
]);
```

```
λ php artisan db:seed --class=KategoriSeeder
```

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
Edit Copy Delete	1	FOOD	Makanan	NULL	NULL
Edit Copy Delete	2	DRINK	Minuman	NULL	NULL
Edit Copy Delete	3	BEAUTY	Kecantikan	NULL	NULL
Edit Copy Delete	4	HOME	Peralatan Rumah	NULL	NULL
Edit Copy Delete	5	BABY	Perlengkapan Bayi	NULL	NULL

```
λ php artisan make:seeder BarangSeeder
```

```php
DB::table('m_barang')->insert([
    ['kategori_id' => 1, 'barang_kode' => 'BRG001', 'barang_nama' =>
    ['kategori_id' => 1, 'barang_kode' => 'BRG002', 'barang_nama' =>
    ['kategori_id' => 2, 'barang_kode' => 'BRG003', 'barang_nama' =>
    ['kategori_id' => 2, 'barang_kode' => 'BRG004', 'barang_nama' =>
    ['kategori_id' => 3, 'barang_kode' => 'BRG005', 'barang_nama' =>
    ['kategori_id' => 3, 'barang_kode' => 'BRG006', 'barang_nama' =>
    ['kategori_id' => 4, 'barang_kode' => 'BRG007', 'barang_nama' =>
    ['kategori_id' => 4, 'barang_kode' => 'BRG008', 'barang_nama' =>
    ['kategori_id' => 5, 'barang_kode' => 'BRG009', 'barang_nama' =>
    ['kategori_id' => 5, 'barang_kode' => 'BRG010', 'barang_nama' =>
]);
```

```
λ php artisan db:seed --class=BarangSeeder
```

←T→	barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
Edit Copy Delete	1	1	BRG001	Nasi Goreng	10000	15000	NULL	NULL
Edit Copy Delete	2	1	BRG002	Mie Goreng	9000	14000	NULL	NULL
Edit Copy Delete	3	2	BRG003	Es Teh	3000	5000	NULL	NULL
Edit Copy Delete	4	2	BRG004	Jus Jeruk	5000	8000	NULL	NULL
Edit Copy Delete	5	3	BRG005	Lipstik	25000	40000	NULL	NULL
Edit Copy Delete	6	3	BRG006	Face Wash	15000	25000	NULL	NULL
Edit Copy Delete	7	4	BRG007	Sapu Lidi	10000	20000	NULL	NULL
Edit Copy Delete	8	4	BRG008	Pel Lantai	20000	35000	NULL	NULL
Edit Copy Delete	9	5	BRG009	Diapers	50000	75000	NULL	NULL
Edit Copy Delete	10	5	BRG010	Bedak Bayi	10000	15000	NULL	NULL

```
λ php artisan make:seeder StokSeeder
```

```php
public function run(): void
{
    for ($i = 1; $i <= 10; $i++) {
        DB::table('t_stok')->insert([
            'barang_id' => $i,
            'stok_tanggal' => Carbon::now(),
            'stok_jumlah' => rand(10, 100),
        ]);
    }
}
```

```
λ php artisan db:seed --class=StokSeeder
```

←T→	stok_id	barang_id	stok_tanggal	stok_jumlah	created_at	updated_at
Edit Copy Delete	1	1	2025-03-06 01:48:45	82	NULL	NULL
Edit Copy Delete	2	2	2025-03-06 01:48:53	82	NULL	NULL
Edit Copy Delete	3	3	2025-03-06 01:48:54	73	NULL	NULL
Edit Copy Delete	4	4	2025-03-06 01:48:54	73	NULL	NULL
Edit Copy Delete	5	5	2025-03-06 01:48:54	45	NULL	NULL
Edit Copy Delete	6	6	2025-03-06 01:48:54	91	NULL	NULL
Edit Copy Delete	7	7	2025-03-06 01:48:54	23	NULL	NULL
Edit Copy Delete	8	8	2025-03-06 01:48:54	100	NULL	NULL
Edit Copy Delete	9	9	2025-03-06 01:48:54	16	NULL	NULL
Edit Copy Delete	10	10	2025-03-06 01:48:54	52	NULL	NULL

```
λ php artisan make:seeder PenjualanDetailSeeder
```

```php
public function run(): void
{
    for ($i = 1; $i <= 10; $i++) {
        for ($j = 1; $j <= 3; $j++) { // 3 barang untuk setiap transak
            DB::table('t_penjualan_detail')->insert([
                'penjualan_id' => $i,
                'barang_id' => rand(1, 10), // Barang acak
                'jumlah' => rand(1, 5),
            ]);
        }
    }
}
```

D. DB Facade

1. Kita buat controller dahulu untuk mengelola data pada table m_level
2. Kita modifikasi dulu untuk routing-nya, ada di PWL_POS/routes/web.php
3. Selanjutnya, kita modifikasi file LevelController untuk menambahkan 1 data ke table m_level
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
5. Selanjutnya, kita modifikasi lagi file LevelController untuk meng-update data di table m_level seperti berikut
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
8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table m_level. Kita modifikasi file LevelController seperti berikut
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
10. Jalankan di web browser

E. Query Builder

Query builder adalah fitur yang disediakan Laravel untuk melakukan proses CRUD (create, retrieve/read, update, delete) pada database. 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.

Perintah pertama yang dilakukan pada query builder adalah menentukan nama table yang akan dilakukan operasi CRUD. Kemudian baru disusul method yang ingin digunakan sesuai dengan peruntukannya.

Praktikum 5 - Implementasi Query Builder

1. Kita buat controller dahuku untuk mengelola data pada table m_kategori
2. Kita modifikasi dulu untuk routing-nya, ada di PWL_POS/routes/web.php
3. Selanjutnya, kita modifikasi file KategoriController untuk menambahkan 1 data ke table m_kategori
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
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
8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table m_kategori. Kita modifikasi file KategoriController seperti berikut
9. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil view(‘kategori’), maka kita buat file view pada VSCode di PWL_POS/resources/view/kategori.blade.php
10. Jalankan di browser

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.

Untuk bisa melakukan operasi CRUD (create, read/retrieve, update, delete), kita harus membuat sebuah model sesuai dengan target tabel yang ingin digunakan. Jadi, dalam 1 model, merepresentasikan 1 tabel database.

Praktikum 6 - Implementasi Eloquent 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
4. Kita modifikasi route web.php untuk mencoba routing ke controller UserController
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
8. Setelah itu, kita modifikasi lagi file UserController
9. Jalankan di browser, amati dan laporkan apa yang terjadi
10. Kita modifikasi lagi file UserController menjadi seperti berikut
11. Jalankan di browser

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?

   Jawab:

   APP_KEY digunakan untuk **enkripsi data dalam Laravel**, seperti sesi pengguna, token, dan password yang dienkripsi menggunakan hashing.

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

   Jawab:

   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?

   Jawab:

   4 file yaitu, create_users_table.php, create_password_reset_tokens_table.php, create_failed_jobs_table.php, dan create_personal_access_tokens_table.php

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

   Jawab:

   Untuk secara otomatis menambahkan 2 kolom yang berfungsi untuk menyimpan waktu kapan data dibuat dan menyimpan waktu kapan data diperbarui terakhir kali

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

   Jawab:

   menghasilkan kolom primary key dengan tipe data BIGINT (unsigned big integer) dan auto-increment.

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

   Jawab:

   $table->id(); → Secara default, Laravel akan membuat primary key dengan nama id.

   $table->id('level_id'); → Primary key tetap berupa BIGINT (unsigned big integer) dan auto-increment, tetapi namanya diubah menjadi level_id.

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

   Jawab:

   untuk memastikan nilai dalam kolom tidak boleh duplikat

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') ?

   Jawab:

   Pada m_level, $table->id('level_id'); karena kolom ini adalah primary key, sehingga bersifat auto-increment.

   Pada m_user, $table->unsignedBigInteger('level_id'); karena kolom ini adalah foreign key yang merujuk ke level_id pada tabel m_level.

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

   Jawab:

   Class Hash digunakan untuk melakukan hashing pada data sensitif, seperti password.

   Kode Hash::make('1234'); berarti password "1234" akan diubah menjadi hash terenkripsi, sehingga lebih aman saat disimpan di database

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

    Jawab:

    digunakan untuk binding parameter dalam query SQL agar lebih aman dari serangan SQL Injection

11. Pada Praktikum 6 - Tahap 3, apa tujuan penulisan kode protected $table = ‘m_user’; dan protected $primaryKey = ‘user_id’; ?

    Jawab:

    protected $table = 'm_user'; → Memberitahu Eloquent bahwa model ini menggunakan tabel m_user.

    protected $primaryKey = 'user_id'; → Mengatur primary key tabel ini agar menggunakan kolom 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

    Jawab:

Eloquent ORM lebih mudah digunakan untuk CRUD karena lebih singkat dan berbasis OOP.