LAPORAN PRAKTIKUM

Pemrograman Web Lanjut

Jobsheet - 3 : MIGRATION, SEEDER, DB FACADE, QUERY BUILDER, dan ELOQUENT ORM

Nama : Ghoffar Abdul Ja'far

NIM : 2341720035

Kelas : 2F

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2023/2024

Praktikum

Praktikum 1 - Pengaturan database:

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

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

```
Minggu_3 > PWL_POS > .env
1   APP_NAME=Laravel
2   APP_ENV=local
3   APP_KEY=
4   APP_DEBUG=true
5   APP_URL=http://localhost
6
7   LOG_CHANNEL=stack
8   LOG_DEPRECATIONS_CHANNEL=null
9   LOG_LEVEL=debug
10
11  DB_CONNECTION=mysql
12  DB_HOST=127.0.0.1
13  DB_PORT=3310
14  DB_DATABASE=PWL_POS
15  DB_USERNAME=root
16  DB_PASSWORD=
```

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

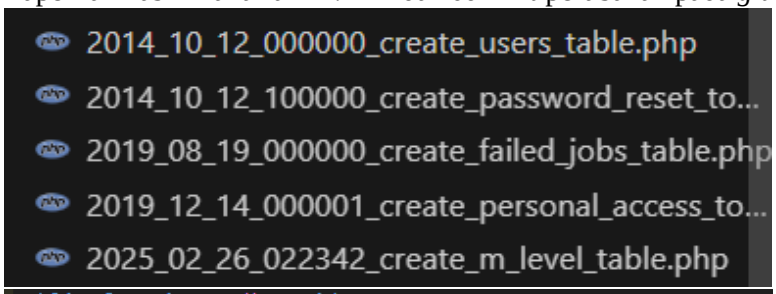
```
Minggu_3 > PWL_POS > .env
1   APP_NAME=Laravel
2   APP_ENV=local
3   APP_KEY=base64:7GOva8c06eAYl0KXok5qu8XW0eux2XcoujDLTQlR63I=
4   APP_DEBUG=true
5   APP_URL=http://localhost
6
7   LOG_CHANNEL=stack
8   LOG_DEPRECATIONS_CHANNEL=null
9   LOG_LEVEL=debug
10
11  DB_CONNECTION=mysql
12  DB_HOST=127.0.0.1
13  DB_PORT=3310
14  DB_DATABASE=PWL_POS
15  DB_USERNAME=root
16  DB_PASSWORD=
```

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

Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

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

2. Kita abaikan dulu yang di kotak merah (jangan di hapus)
3. Kita buat file migrasi untuk table m_level dengan perintah
4. Kita perhatikan bagian yang di kotak merah, bagian tersebut yang akan kita modifikasi sesuai desain database yang sudah ada
5. Simpan kode pada tahapan 4 tersebut, kemudian jalankan perintah ini pada terminal VSCode untuk melakukan migrasi
6. Kemudian kita cek di phpMyAdmin apakah table sudah ter-generate atau belum
7. Ok, table sudah dibuat di database
8. Buat table *database* dengan *migration* untuk table **m_kategori** yang sama-sama tidak memiliki *foreign key*
9. Laporkan hasil Praktikum-2.1 ini dan *commit* perubahan pada *git*




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

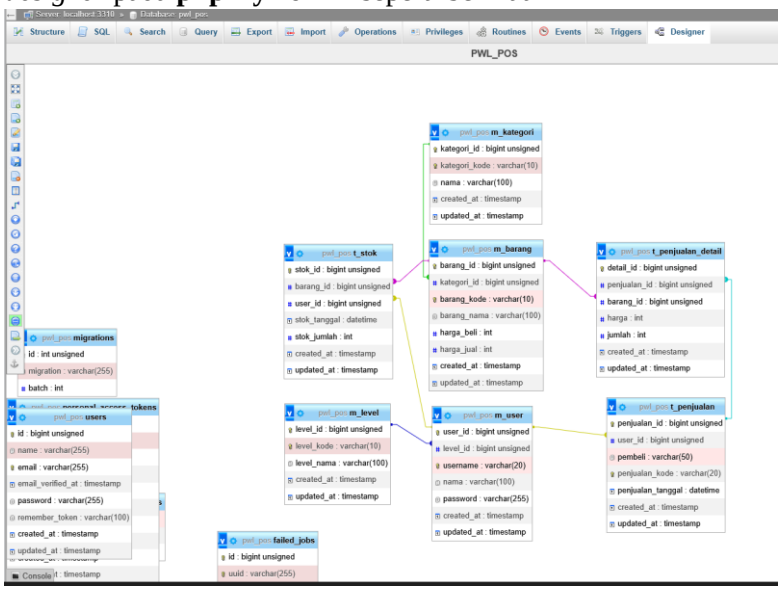
Praktikum 2.2 - Pembuatan file migrasi dengan relasi

1. Buka *terminal* VSCode kalian, dan buat file migrasi untuk table **m_user**
2. Buka file migrasi untuk table **m_user**, dan modifikasi seperti berikut

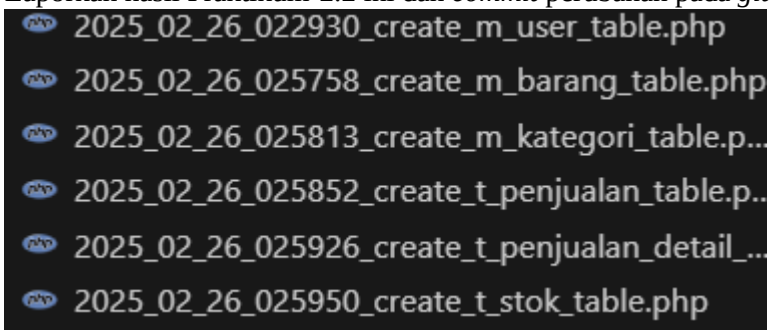
```php
public function up(): void
{
    Schema::create('m_user', function (Blueprint $table) {
        $table->id('user_id');
        $table->unsignedBigInteger('level_id')->index();
        $table->string('username', 20)->unique();
        $table->string('nama', 100);
        $table->string('password');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

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

3. Simpan kode program Langkah 2, dan jalankan perintah **php artisan migrate**. Amati apa yang terjadi pada database.
4. Buat table *database* dengan *migration* untuk table-tabel yang memiliki *foreign key*

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

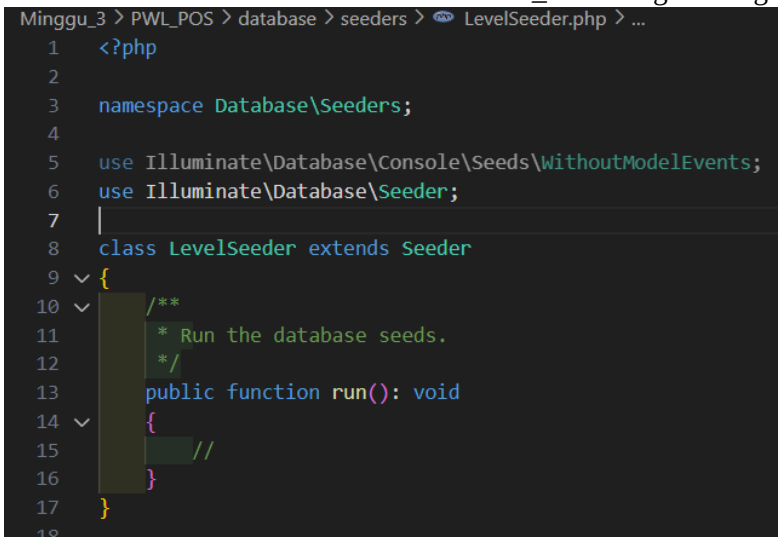


6. Laporkan hasil Praktikum-2.2 ini dan *commit* perubahan pada *git*.



Praktikum 3 - Membuat file *seeder*

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



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

```
Minggu_3 > PWL_POS > database > seeders > LevelSeeder.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(): void
    {
        $data = [
            ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
            ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
            ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
        ];
        DB::table('m_level')->insert($data);
    }
}

```

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

```
PS E:\!Kuliah\Pemrograman_Web_Lanjut\Minggu_3\PWL_POS> php artisan db:seed --class=LevelSeeder

  INFO  Seeding database.

PS E:\!Kuliah\Pemrograman_Web_Lanjut\Minggu_3\PWL_POS>
```

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

	level_id	level_kode	level_nama	created_at	updated_at
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

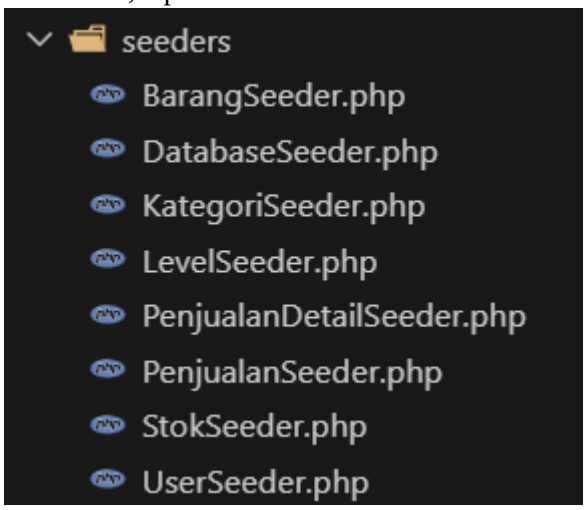
5. Sekarang kita buat file *seeder* untuk table m_user yang me-*refer* ke table m_level
6. Modifikasi file class **UserSeeder** seperti berikut

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

```

7. Jalankan perintah untuk mengeksekusi class UserSeeder

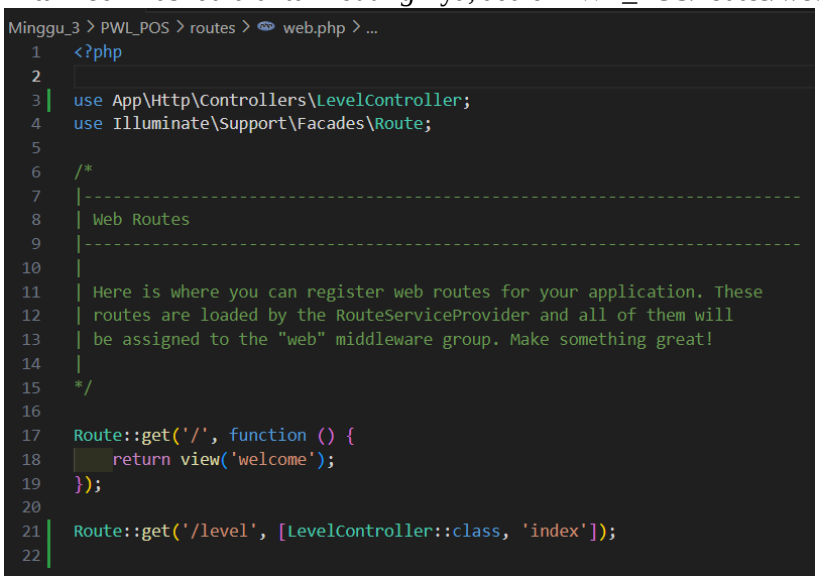
	user_id	level_id	username	nama	password	created_at	updated_at
Edit Copy Delete	1	1	admin	Administrator	$2y$12$4xKnZTLKcYJE6phNcDdXIuls0jDpw9bNYQBDm0CjBaE...	NULL	NULL
Edit Copy Delete	2	2	manager	Manager	$2y$12$TnUbxka42qyukx68K7ftXuyMv8joRbJ5d92gZZqw4h5...	NULL	NULL
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$aii971qyNfYsZc49wrOPnOB.GoGgZeGc8VGW1IIjlxO...	NULL	NULL

8. Perhatikan hasil seeder pada table m_user
9. Ok, data seeder berhasil di masukkan ke database.
10. Sekarang coba kalian masukkan data *seeder* untuk table yang lain, dengan ketentuan seperti berikut
11. Jika sudah, laporkan hasil Praktikum-3 ini dan *commit* perubahan pada *git*

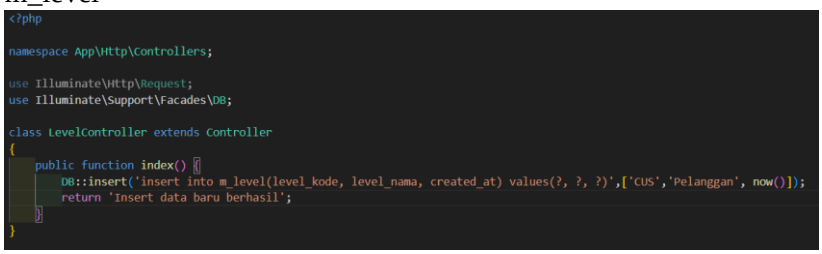


Praktikum 4 - Implementasi DB Facade

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

	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
Edit Copy Delete	4	CUS	Pelanggan	2025-03-04 15:37:59	NULL

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

```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';

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

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

```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';

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

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

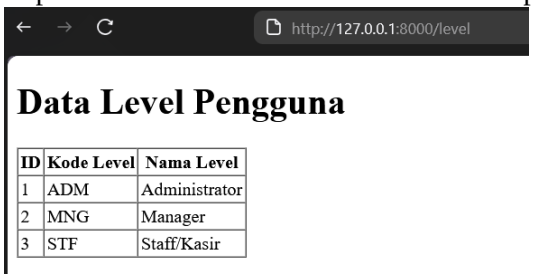
8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table m_level. Kita modifikasi file LevelController seperti berikut

```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';

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

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

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

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

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Data Level Pengguna</title>
</head>
<body>
    <h1>Data Level Pengguna</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <th>ID</th>
            <th>Kode Level</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>
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi
11. Laporkan hasil Praktikum-4 ini dan *commit* perubahan pada *git*.



Praktikum 5 - Implementasi *Query Builder*

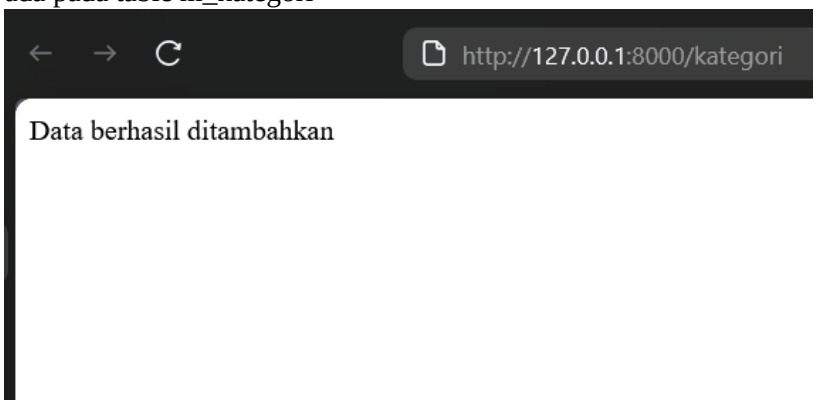
1. Kita buat controller dahuku untuk mengelola data pada table m_kategori
2. Kita modifikasi dulu untuk routing-nya, ada di PWL_POS/routes/web.php

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

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

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

```php
<?php

namespace App\Http\Controllers;

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

class KategoriController extends Controller
{
    public function index()
    {
        $data = [
            'kategori_kode' => 'K06',
            'kategori_nama' => 'Buku',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Data berhasil ditambahkan';
    }
}
```

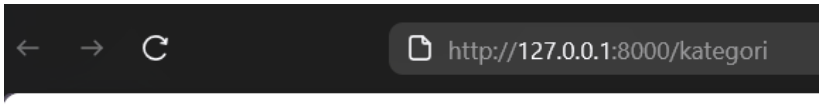
4. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/kategori dan amati apa yang terjadi pada table m_kategori di database, *screenshot* perubahan yang ada pada table m_kategori



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

```php
class KategoriController extends Controller
{
    public function index()
    {
        // $data = [
        //     'kategori_kode' => 'K06',
        //     'nama' => 'Buku',
        //     'created_at' => now()
        // ];
        // DB::table('m_kategori')->insert($data);
        // return 'Data berhasil ditambahkan';

        $row = DB::table('m_kategori')->where('kategori_kode', 'K06')->update(['nama' => 'Buku Tulis']);
        return 'Data berhasil diupdate. Jumlah data yang diupdate: ' . $row . ' baris';
    }
}
```

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



Data berhasil diupdate. Jumlah data yang diupdate: 1 baris

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

```php
class KategoriController extends Controller
{
    public function index()
    {
        // $data = [
        //     'kategori_kode' => 'K06',
        //     'nama' => 'Buku',
        //     'created_at' => now()
        // ];
        // DB::table('m_kategori')->insert($data);
        // return 'Data berhasil ditambahkan';

        // $row = DB::table('m_kategori')->where('kategori_kode', 'K06')->update(['nama' => 'Buku Tulis']);
        // return 'Data berhasil diupdate. Jumlah data yang diupdate: ' . $row . ' baris';

        $row = DB::table('m_kategori')->where('kategori_kode', 'K06')->delete();
        return 'Data berhasil dihapus. Jumlah data yang dihapus: ' . $row . ' baris';
    }
}
```

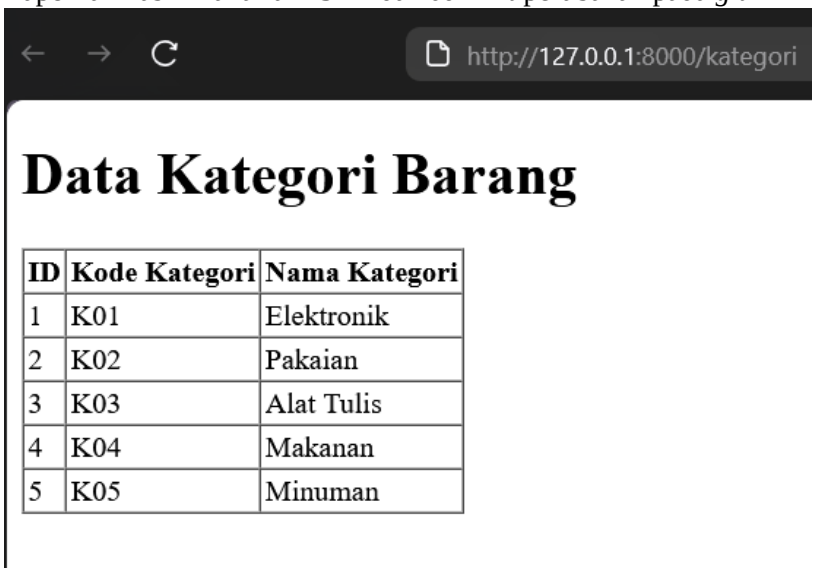
8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table m_kategori. Kita modifikasi file KategoriController seperti berikut

```php
class KategoriController extends Controller
{
    public function index()
    {
        // $data = [
        //     'kategori_kode' => 'K06',
        //     'nama' => 'Buku',
        //     'created_at' => now()
        // ];
        // DB::table('m_kategori')->insert($data);
        // return 'Data berhasil ditambahkan';

        // $row = DB::table('m_kategori')->where('kategori_kode', 'K06')
        // return 'Data berhasil diupdate. Jumlah data yang diupdate: '

        // $row = DB::table('m_kategori')->where('kategori_kode', 'K06')
        // return 'Data berhasil dihapus. Jumlah data yang dihapus: ' .

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

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

```blade
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Data Kategori Barang</title>
</head>
<body>
    <h1>Data Kategori Barang</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <th>ID</th>
            <th>Kode Kategori</th>
            <th>Nama Kategori</th>
        </tr>
        @foreach ($data as $d)
        <tr>
            <td>{{ $d->kategori_id }}</td>
            <td>{{ $d->kategori_kode }}</td>
            <td>{{ $d->nama }}</td>
        </tr>
        @endforeach
</body>
</html>
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi.
11. Laporkan hasil Praktikum-5 ini dan *commit* perubahan pada *git*



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

```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 oleh model ini
}
```

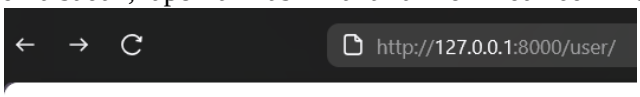
4. Kita modifikasi route web.php untuk mencoba routing ke controller UserController

```php
<?php

use App\Http\Controllers\LevelController;
use App\Http\Controllers\KategoriController;
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']);
```

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

```php
<?php

namespace App\Http\Controllers;

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

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]);
    }
}
```

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



Data User

ID	Username	Nama	ID level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
7	customer 1	Pelanggan Pertama	5

Pertanyaan

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 mengenkripsi data dalam laravel, seperti session dan password
2. Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk APP_KEY?
   Jawab: APP_KEY dapat di generate dengan 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: Secara default, Laravel memiliki tiga file migrasi:
   - 2014_10_12_000000_create_users_table.php: Membuat tabel users.
   - 2014_10_12_100000_create_password_reset_tokens_table.php: Menyimpan token reset password.
   - 2019_08_19_000000_create_failed_jobs_table.php: Mencatat pekerjaan (jobs) yang gagal.
   - 2019_12_14_000001_create_personal_access_tokens_table.php: menyimpan token askses pribadi yang berkaitan dengan laravel Sanctum
4. Secara *default*, file migrasi terdapat kode $table->timestamps();, apa tujuan/*output* dari fungsi tersebut?
   Jawab: untuk menambahkan kolom created at dan updated at secara otomatis
5. Pada File Migrasi, terdapat fungsi $table->id(); Tipe data apa yang dihasilkan dari fungsi tersebut?
   Jawab: Menghasilkan kolom primary key bertipe BIGINT dengan auto-increment
6. Apa bedanya hasil migrasi pada table m_level, antara menggunakan $table->id(); dengan menggunakan $table->id('level_id'); ?
   Jawab: $table->id(); menggunakan nama default id sebagai primary key, sedangkan $table- >id('level_id'); menggunakan nama level_id sebagai primary key
7. Pada migration, Fungsi ->unique() digunakan untuk apa?
   Jawab: digunakan untuk memastikan data tidak memiliki 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: Karena level_id di tabel m_user adalah foreign key yang merujuk ke level_id di tabel m_level, yang menggunakan $table->id('level_id'); yang bertipe BIGINT.
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 enkripsi password. Dan kode tersebut dimaksudkan untuk mengubah string 1234 menjadi hash terenkripsi.
10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?
    Jawab: tanda (?) digunakan sebagai placeholder parameter dalam query sql
11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode protected $table = 'm_user'; dan protected $primaryKey = 'user_id'; ?
    Jawab: penulisan kode protected $table = 'm_user'; digunakan untuk menentukan bahwa model berhubungan dengan tabel m_user. Dan protected $primaryKey = 'user_id'; digunakan untuk menentukan primary key dari tabel ini adalah user_id.
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Façade / Query Builder / Eloquent ORM*) ? jelaskan!

Jawab: penulisan kode protected $table = ‘m_user’; digunakan untuk menentukan bahwa model berhubungan dengan tabel m_user. Dan protected $primaryKey = ‘user_id’; digunakan untuk menentukan primary key dari tabel ini adalah user_id.