LAPORAN JOBSHEET 3

MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, dan ELOQUENT ORM

MATA KULIAH PEMROGRAMAN WEB LANJUT

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

Disusun oleh :

Dahniar Davina SIB-2A / 2341760023

JURUSAN TEKNOLOGI INFORMASI

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS

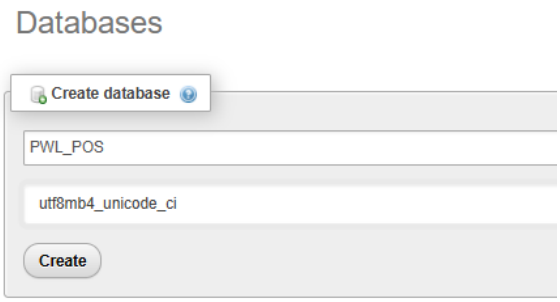
POLITEKNIK NEGERI MALANG

2025

A. PENGATURAN DATABASE

- 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.

```
.env
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:tU7FdQnvkPOzBj3uGEIrP43qvUlw2K266+qS3OT3CE0=
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=
```

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

B. MIGRASI

- 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

```
Windows@DESKTOP-PE7RQP7 MINGW64 /c/laragon/www/Minggu3/Jobsheet3/PWL_POS
$ php artisan make:migration create_m_level_table --create=m_level

  INFO  Migration [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS\database\migr
```

```php
public function up(): void
{
    Schema::create('m_level', function (Blueprint $table) {
        $table->id();
        $table->timestamps();
    });
```

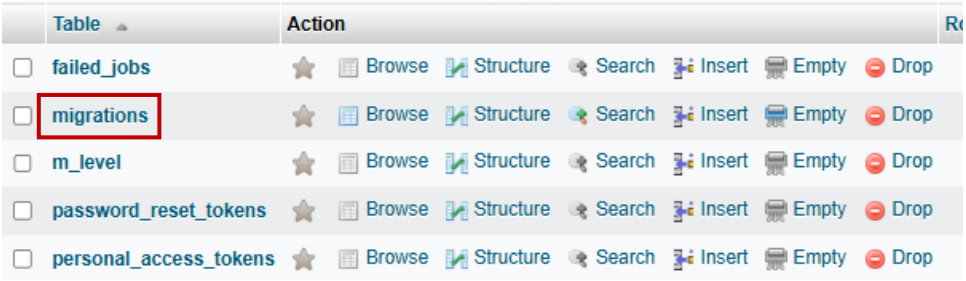
4. Kita perhatikan bagian yang di kotak merah, bagian tersebut yang akan kita modifikasi sesuai desain database yang sudah ada

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

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

```
$ php artisan migrate

  INFO  Running migrations.

  2025_03_05_014742_create_m_level_table ..................... 73ms DONE
```

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

```
Windows@DESKTOP-PE7RQP7 MINGW64 /c/laragon/www/Minggu3/Jobsheet3/PWL_PO$ php artisan make:
migration create_m_kategori_table --create=m_kategori

  INFO  Migration [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS\database\migrations/2025_03_0
5_021407_create_m_kategori_table.php] created successfully.
```

```php
public function up(): void
{
    Schema::create('m_kategori', function (Blueprint $table) {
        $table->id(); //Primary Key, auto-increment
        $table->string('nama_kategori', 100); // Nama kategori dengan panj
        $table->text('deskripsi')->nullable();
        $table->timestamps();
    });
}
```

```
Windows@DESKTOP-PE7RQP7 MINGW64 /c/laragon/www/Minggu3/Jobsheet3/PWL_POS
$ php artisan migrate

  INFO  Running migrations.

  2025_03_05_021407_create_m_kategori_table ..... 49ms DONE
```

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

➢ Pembuatan File Migrasi dengan Relasi

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

```
Windows@DESKTOP-PE7RQP7 MINGW64 /c/laragon/www/Minggu3/Jobsheet3/PWL_POS
$ php artisan make:migration create_m_user_table --table=m_user

  INFO  Migration [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS\database\mig
_05_022151_create_m_user_table.php] created successfully.
```

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

```php
Schema::table('m_user', function (Blueprint $table) {
    $table->id('user_id');
    $table->unsignedBigInteger('level_id')->index(); // Indexing untuk For
    $table->string('username', 20)->unique(); // Unique untuk memastikan t
    $table->string('nama', 100);
    $table->string('password');
    $table->timestamps();

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

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

```
$ php artisan migrate

  INFO  Running migrations.

  2025_03_05_024217_create_m_user_table ..... 129ms DONE
```

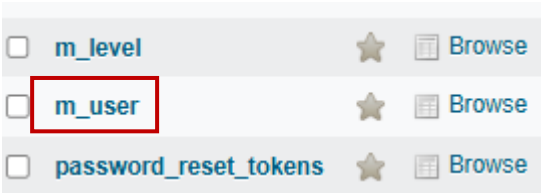


Table m_user muncul pada database

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

a) Buat Migration m_barang

```
$ php artisan make:migration create_m_barang_table

  INFO  Migration [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS\database\migrations/2025_03_05_031218_create_m_barang_table.php] created successfully.
```

```
public function up(): void
{
    Schema::create('m_barang', function (Blueprint $table
        $table->id(); //primary key
        $table->string('nama_barang', 100);
        $table->integer('harga');
        $table->integer('stok')->default(0);
        $table->timestamps();
    });
```

b) Buat Migration t_penjualan

```
$ php artisan make:migration create_t_penjualan_table

  INFO  Migration [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS\database\migrations/2025_03_05_031250_create_t_penjualan_table.php] created successfully.
```

```
public function up(): void
{
    Schema::create('t_penjualan', function (Blueprint $t
        $table->id(); //primary key
        $table->date('tanggal');
        $table->integer('total_harga');
        $table->timestamps();
    });
}
```

c) Buat Migration t_stok

```
$ php artisan make:migration create_t_stok_table

  INFO  Migration [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS\database\migrations/2025_03_05_031301_create_t_stok_table.php] created successfully.
```

```
Schema::create('t_stok', function (Blueprint $table) {
    $table->id(); //primary key
    $table->unsignedBigInteger('barang_id'); // Foreign Key ke m_barang
    $table->integer('jumlah');
    $table->date('tanggal_masuk');
    $table->timestamps();

    // Foreign Key Constraint
    $table->foreign('barang_id')->references('id')->on('m_barang')->onDelete('cascade');
});
```

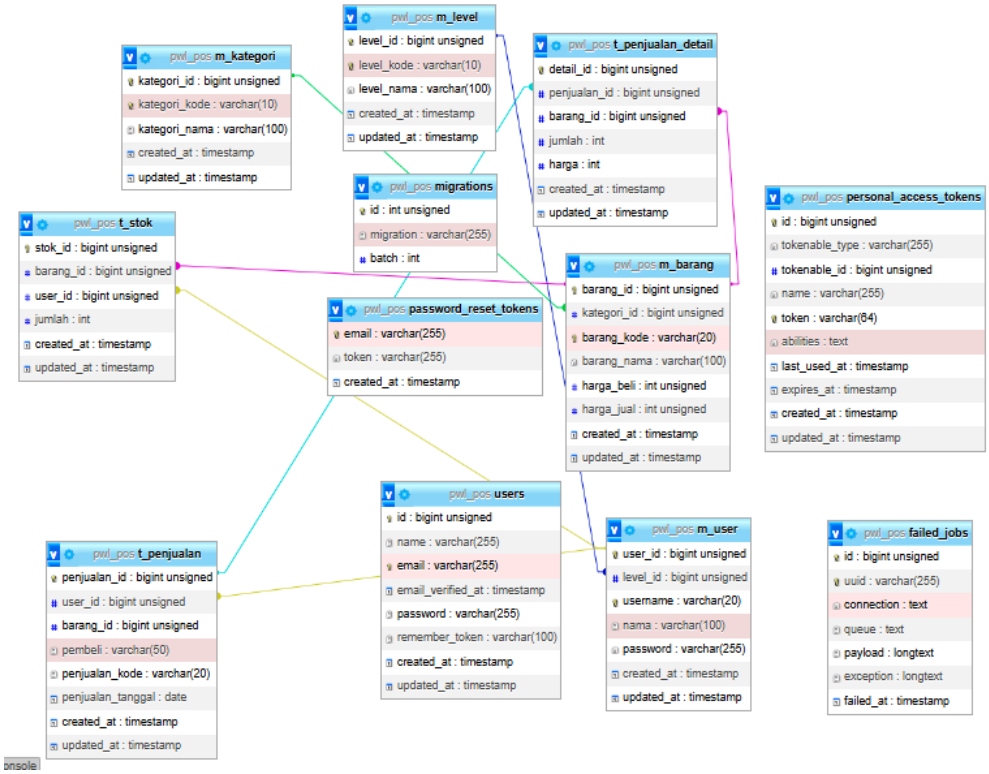
d) Buat Migration t_penjualan_detail

```
$ php artisan make:migration create_t_penjualan_detail_table

  INFO  Migration [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS\database\migrations/2025_03_05_031317_create_t_penjualan_detail_table.php] created successfully.
```

```php
Schema::create('t_penjualan_detail', function (Blueprint $table) {
    $table->id(); //primary key
    $table->unsignedBigInteger('penjualan_id'); // Foreign Key ke t_penjualan
    $table->unsignedBigInteger('barang_id'); // Foreign Key ke m_barang
    $table->integer('jumlah');
    $table->integer('harga_satuan');
    $table->integer('subtotal');
    $table->timestamps();

    // Foreign Key Constraints
    $table->foreign('penjualan_id')->references('id')->on('t_penjualan')->onDelete('cascade');
    $table->foreign('barang_id')->references('id')->on('m_barang')->onDelete('cascade');
});
```

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

```
$ php artisan migrate

  INFO  Running migrations.

  2025_03_05_031218_create_m_barang_table ..... 49ms DONE
  2025_03_05_031250_create_t_penjualan_table ..... 22ms DONE
  2025_03_05_031301_create_t_stok_table ..... 26ms DONE
  2025_03_05_031317_create_t_penjualan_detail_table ..... 57ms DONE
```



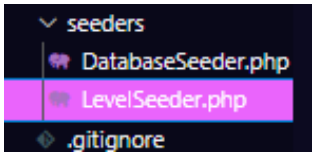
6. Laporkan hasil Praktikum-2.2 ini dan commit perubahan pada git.

C. SEEDER

➢ Membuat File Seeder

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

```
$ php artisan make:seeder LevelSeeder

  INFO  Seeder [C:\laragon\www\Minggu3\Jobsheet3\P
hp] created successfully.
```



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

```php
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

```
Windows@DESKTOP-PE7RQP7 MINGW64 /c/laragon/www/Minggu3/Jobsheet3/PWL_POS
$ 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

```
$ php artisan make:seeder UserSeeder

  INFO  Seeder [C:\laragon\www\Minggu3\Jobsheet3\PWL
eeders\UserSeeder.php] created successfully.
```

6. Modifikasi file class UserSeeder seperti Berikut

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

7. Jalankan perintah untuk mengeksekusi class UserSeeder

```
$ 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$12$sevrEUE6i6f2LN4vEH.6t.8
Edit Copy Delete	2	2	manager	Manager	$2y$12$6zubkcA5pr1bnhnq2on5gu
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$XV6l83m.tP79btXcRX1Ed

9. Ok, data seeder berhasil di masukkan ke database
10. Sekarang coba kalian masukkan data seeder untuk table yang lain, dengan ketentuan seperti Berikut

No	Nama Tabel	Jumlah Data	Keterangan
1	m_kategori	5	5 kategori barang
2	m_barang	10	10 barang yang berbeda
3	t_stok	10	Stok untuk 10 barang
4	t_penjualan	10	10 transaksi penjualan
5	t_penjualan_detail	30	3 barang untuk setiap transaksi penjualan

a) m_kategori

```
$ php artisan make:seeder KategoriSeeder

  INFO  Seeder [C:\laragon\www\Minggu3\Jobsheet3\PWL
se\seeders\KategoriSeeder.php] created successfully.
```

```php
public function run(): void
{
    DB::table('m_kategori')->insert([
        ['kategori_id' => 1, 'kategori_kode' => 'KTG001', 'kategori_nama'
        ['kategori_id' => 2, 'kategori_kode' => 'KTG002', 'kategori_nama'
        ['kategori_id' => 3, 'kategori_kode' => 'KTG003', 'kategori_nama'
        ['kategori_id' => 4, 'kategori_kode' => 'KTG004', 'kategori_nama'
        ['kategori_id' => 5, 'kategori_kode' => 'KTG005', 'kategori_nama'
    ]);
}
```

←T→	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
Edit Copy Delete	1	KTG001	Elektronik	NULL	NULL
Edit Copy Delete	2	KTG002	Pakaian	NULL	NULL
Edit Copy Delete	3	KTG003	Makanan	NULL	NULL
Edit Copy Delete	4	KTG004	Minuman	NULL	NULL
Edit Copy Delete	5	KTG005	Obat-obatan	NULL	NULL

b) m_barang

```
$ php artisan make:seeder BarangSeeder

  INFO  Seeder [C:\laragon\www\Minggu3\Jobsheet3\P
se\seeders\BarangSeeder.php] created successfully.
```

```php
DB::table('m_barang')->insert([
    [
        'barang_id' => 1,
        'kategori_id' => 1,
        'barang_kode' => 'BRG001',
        'barang_nama' => 'Laptop ASUS',
        'harga_beli' => 16000000,
        'harga_jual' => 17500000
    ],
    [
        'barang_id' => 2,
        'kategori_id' => 1,
        'barang_kode' => 'BRG002',
        'barang_nama' => 'MacBook Air M1',
        'harga_beli' => 19000000,
        'harga_jual' => 20500000
    ],
    [
```

←T→	barang_id	kategori_id	barang_kode	barang_nama	harga_beli	har
Edit Copy Delete	1	1	BRG001	Laptop ASUS	16000000	1
Edit Copy Delete	2	1	BRG002	MacBook Air M1	19000000	2
Edit Copy Delete	3	1	BRG003	Monitor 32 inch OLED	3500000	
Edit Copy Delete	4	2	BRG004	Blouse Putih	50000	
Edit Copy Delete	5	2	BRG005	Jaket Denim Abu	200000	
Edit Copy Delete	6	2	BRG006	Sneakers Adidas	600000	
Edit Copy Delete	7	3	BRG007	Roti Sandwich	15000	
Edit Copy Delete	8	3	BRG008	Biskuit Roma Kelapa	15000	

c) t_stok

```
$ php artisan make:seeder StokSeeder

  INFO  Seeder [C:\laragon\www\Minggu3\Jobsheet3
se\seeders\StokSeeder.php] created successfully.
```

```php
use Carbon\Carbon;

class StokSeeder extends Seeder
{
    public function run(): void
    {
        $stok = [];

        for ($i = 1; $i <= 15; $i++) {
            $stok[] = [
                'stok_id' => $i,
                'barang_id' => $i,
                'user_id' => rand(1, 3),
                'stok_tanggal' => Carbon::now()->subDays(rand(1, 30))->toDateT
                'stok_jumlah' => rand(1, 50),
            ];
        }

        DB::table('t_stok')->insert($stok);
    }
}
```

←T→	stok_id	barang_id	user_id	jumlah	created_at	updated_at	s
Edit Copy Delete	1	1	3	0	NULL	NULL	
Edit Copy Delete	2	2	2	0	NULL	NULL	
Edit Copy Delete	3	3	1	0	NULL	NULL	
Edit Copy Delete	4	4	3	0	NULL	NULL	
Edit Copy Delete	5	5	1	0	NULL	NULL	

d) t_penjualan

```
$ php artisan make:seeder PenjualanSeeder

  INFO  Seeder [C:\laragon\www\Minggu3\Jobsheet3\PWL_
se\seeders\PenjualanSeeder.php] created successfully.
```

```php
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Carbon\Carbon;
use Faker\Factory;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class PenjualanSeeder extends Seeder
{
    public function run(): void
    {
        $penjualan = [];

        for ($i = 1; $i <= 15; $i++) {
            $penjualan[] = [
                'penjualan_id' => $i,
                'pembeli' => Factory::create()->unique()->na
                'penjualan_kode' => Factory::create()->uniqu
                'penjualan_tanggal' => Carbon::now()->subDa
                'user_id' => rand(1, 3),
            ];
        }

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

	penjualan_id	user_id	barang_id	pembeli	penjualan_kode	penjual
Edit Copy Delete	1	1	7	Miss Laura Zieme V	architecto	2025-02
Edit Copy Delete	2	3	6	Santiago Hills	saepe	2025-02
Edit Copy Delete	3	3	4	Simone Beier	consectetur	2025-02
Edit Copy Delete	4	1	2	Kevin Dickinson	ut	2025-02
Edit Copy Delete	5	2	1	Kiana McGlynn	et	2025-02

e) t_penjualan_detail

```
$ php artisan make:seeder PenjualanDetailSeeder

  INFO  Seeder [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS\data
se\seeders\PenjualanDetailSeeder.php] created successfully.
```

```php
class PenjualanDetailSeeder extends Seeder
{
    public function run(): void
    {
        $data = [];
        for ($i = 1; $i <= 10; $i++) {
            for ($j = 1; $j <= 3; $j++) {
                $data[] = [
                    'penjualan_id' => $i,
                    'barang_id' => $i,
                    'harga' => random_int(10000, 50000),
                    'jumlah' => random_int(1, 10),
                ];
            }
        }

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

←T→	detail_id	penjualan_id	barang_id	jumlah	harga	created_at	upda
Edit Copy Delete	1	1	1	6	36799	NULL	NULL
Edit Copy Delete	2	1	1	5	13833	NULL	NULL
Edit Copy Delete	3	1	1	5	19605	NULL	NULL
Edit Copy Delete	4	2	2	8	37018	NULL	NULL
Edit Copy Delete	5	2	2	5	28510	NULL	NULL
Edit Copy Delete	6	2	2	1	25132	NULL	NULL

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

D. DB FAÇADE

➢ Implementasi DB Façade

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

```
$ php artisan make:controller LevelController

  INFO  Controller [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS\ap\Http\Controllers\LevelController.php] created successfully.
```

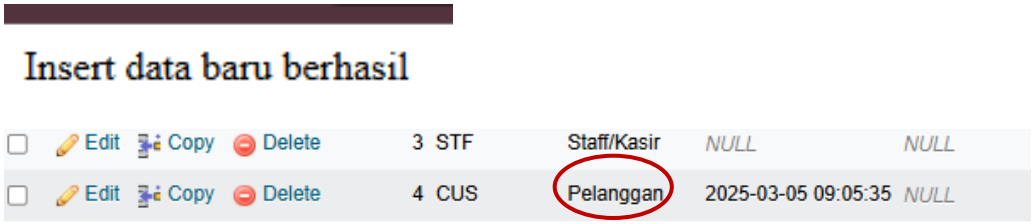
2. Kita modifikasi dulu untuk routing-nya, ada di PWL_POS/routes/web.php

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

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

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

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

```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_n
        return 'Insert data baru berhasil';
    }
}
```

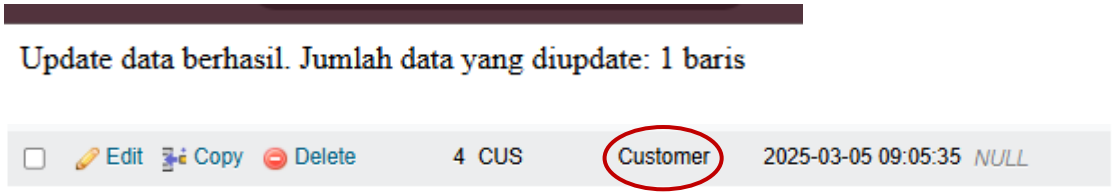
4. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/level dan amati apa yang terjadi pada table m_level di database, screenshot perubahan yang ada pada table m_level



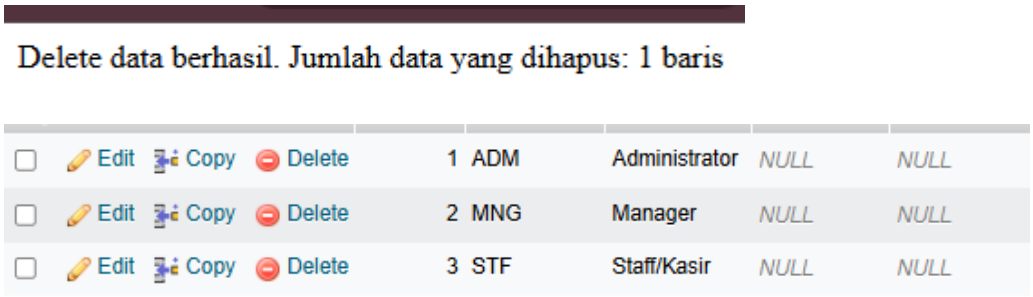
5. Selanjutnya, kita modifikasi lagi file LevelController untuk meng-update data di table m_level seperti Berikut

```
    //return 'Insert data baru berhasil';
    $row = DB::update('update m_level set level_nama = :
    return 'Update data berhasil. Jumlah data yang diupc

}
```

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

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

←T→	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

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

```
<!DOCTYPE html>
<html>
<head>
    <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>
\end{code}
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi

Data Level Pengguna

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

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

E. QUERY BUILDER

➢ Implementasi Query Builder

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

```
$ php artisan make:controller KategoriController

  INFO  Controller [C:\laragon\www\Minggu3\Jobsheet3\PWL
p\Http\Controllers\KategoriController.php] created succes
```

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

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

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

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

class KategoriController extends Controller
{
    public function index()
    {
        $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];

        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil';
    }
}
```

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

Insert data baru berhasil

Edit Copy Delete	5	KTG005	Obat-obatan	NULL	NULL
Edit Copy Delete	6	SNK	Snack/Makanan Ringan	2025-03-05 09:32:11	NULL

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

```
$row = DB::table('m_kategori')->where('kategori_kode', 'SN
return 'Update data berhasil. Jumlah data yang diupdate: '
```

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

Update data berhasil. Jumlah data yang diupdate: 1 baris

7. Kita coba modifikasi lagi file KategoriController untuk melakukan proses 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: 1 baris

Edit Copy Delete	2	KTG002	Pakaian	NULL	NULL
Edit Copy Delete	3	KTG003	Makanan	NULL	NULL
Edit Copy Delete	4	KTG004	Minuman	NULL	NULL
Edit Copy Delete	5	KTG005	Obat-obatan	NULL	NULL

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

```
$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

```
<!DOCTYPE html>
<html>
<head>
    <title>Data Kategori Barang</title>
</head>
<body>
    <h1>Data Kategori Barang</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
```

```blade
<!DOCTYPE html>
<html>
<head>
    <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->kategori_nama }}</td>
        </tr>
        @endforeach
    </table>
</body>
</html>
```

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

Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	KTG001	Elektronik
2	KTG002	Pakaian
3	KTG003	Makanan
4	KTG004	Minuman
5	KTG005	Obat-obatan

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

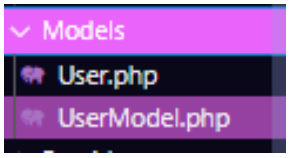
F. ELOQUENT ORM

- Implementasi Eloquent ORM

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

```
$ php artisan make:model UserModel

  INFO  Model [C:\laragon\www\Minggu3\Jobsheet
els\UserModel.php] created successfully.
```

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

```
ass UserModel extends Model

 use HasFactory;

 protected $table = 'm_user'; // Mendefinisika
 protected $primaryKey = 'user_id'; // Mendefi
```

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

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

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 artisan make:controller UserController

  INFO  Controller [C:\laragon\www\Minggu3\Jobsheet3\PWL_POS
p\Http\Controllers\UserController.php] created successfully.
```

```
use App\Models\UserModel;

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

6. Kemudian kita buat view user.blade.php

```blade
<!DOCTYPE html>
<html>
<head>
    <title>Data User</title>
</head>
<body>
    <h1>Data User</h1>
    <table border="1" cellpadding="2" cellspacing="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>
```

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

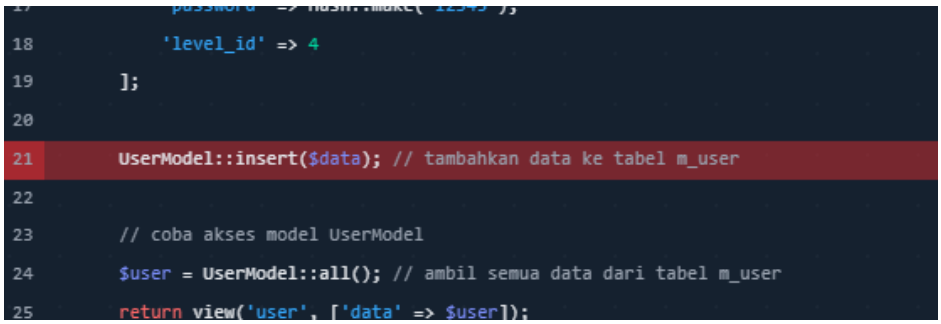
Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3

8. Setelah itu, kita modifikasi lagi file UserController

```php
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
```

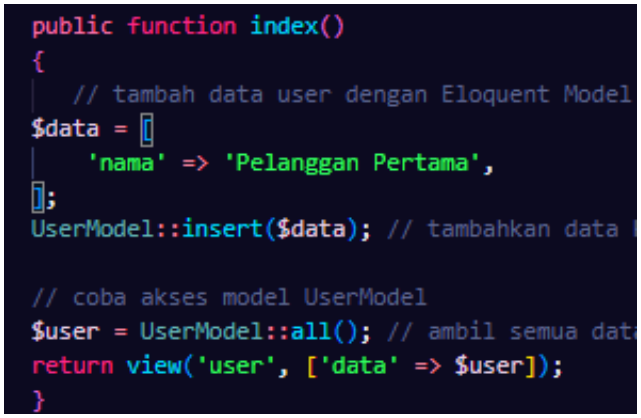
9. Jalankan di browser, amati dan laporkan apa yang terjadi



```
18          'level_id' => 4
19      ];
20
21      UserModel::insert($data); // tambahkan data ke tabel m_user
22
23      // coba akses model UserModel
24      $user = UserModel::all(); // ambil semua data dari tabel m_user
25      return view('user', ['data' => $user]);
```

Error, tidak ada nilai 4 dalam tabel m_level, sehingga tidak bisa menambahkan user dengan level_id = 4. Karena sudah didrop pada langkah praktikum sebelumnya.

10. Kita modifikasi lagi file UserController menjadi seperti Berikut



```
public function index()
{
    // tambah data user dengan Eloquent Model
$data = [
    'nama' => 'Pelanggan Pertama',
];
UserModel::insert($data); // tambahkan data

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

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

Data User

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
6	customer-1	Pelanggan	5

- Id user yang muncul tidak urut (seharusnya 4), karena tipe data id yang saya gunakan auto increment. Sehingga ketika diinputkan kembali dengan id level 4, maka akan error, karena unique code.

12. Jika sudah, laporkan hasil Praktikum-6 ini dan commit perubahan pada git

G. PENUTUP

1. Pada Praktikum 1 - Tahap 5, apakah fungsi dari APP_KEY pada file setting .env Laravel?
    - Untuk membantu membuat token keamanan, seperti CSRF token atau password Hash. Sehingga memastikan bhawa token yang dibuat adalah unik dan aman.
2. Pada Praktikum 1, bagaimana kita men-generate nilai untuk APP_KEY?
    - Menggunakan command php artisan key:generate
3. Pada Praktikum 2.1 - Tahap 1, secara default Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?
    - Secara default Laravel memiliki 4 file migrasi
    - 2014_10_12_000000_create_users_table = File migrasi ini digunakan untuk membuat tabel users, yang menyimpan data pengguna dalam aplikasi Laravel.
    - 2014_10_12_100000_create_password_reset_tokens_table = diguna kan untuk membuat tabel password_reset_tokens, yang menyimpan token reset password jika pengguna lupa password.
    - 2019_08_19_000000_create_failed_jobs_table = digunakan untuk membuat tabel failed_jobs, yang menyimpan informasi tentang job yang gagal diproses dalam Laravel Queue.
    - 2019_12_14_000001_create_personal_access_tokens_table = diguna kan untuk membuat tabel personal_access_tokens, yang menyimpan **token akses pribadi** untuk pengguna yang menggunakan API authentication, seperti **Laravel Sanctum**.
4. Secara default, file migrasi terdapat kode $table->timestamps();, apa tujuan/output dari fungsi tersebut?
    - Saat menambahkan data ke dalam tabel users, kolom **created_at** akan otomatis terisi dengan waktu saat data pertama kali dimasukkan. Jika data diperbarui, **updated_at** akan diperbarui dengan waktu perubahan terakhir.
    - Dapat digunakan untuk tracking kapan data dibuat atau diperbarui.

5. Pada File Migrasi, terdapat fungsi $table->id(); Tipe data apa yang dihasilkan dari fungsi tersebut?
    - BIGINT (Unsigned Big Integer)
6. Apa bedanya hasil migrasi pada table m_level, antara menggunakan $table->id(); dengan menggunakan $table->id('level_id'); ?
    - $table->id(); = jika kita tidak perlu custom nama untuk ID (default : id)
    - $table->id('level_id'); = jika kita ingin custom nama yang lebih deskriptif Atau lebih unik.
7. Pada migration, Fungsi ->unique() digunakan untuk apa?
    - digunakan pada Laravel Migration untuk memastikan bahwa **nilai dalam suatu kolom tidak boleh duplikat**. Artinya, setiap nilai dalam kolom tersebut harus **unik** di dalam tabel.
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') ?
    - Tabel m_level menggunakan $table->id('level_id'); karena level_id adalah Primary Key.
    - Tabel m_user menggunakan $table->unsignedBigInteger('level_id'); karena level_id hanya sebagai Foreign Key yang mengacu ke m_level.level_id.
9. Pada Praktikum 3 - Tahap 6, apa tujuan dari Class Hash? dan apa maksud dari kode program Hash::make('1234');?
    - Tujuan dari Class Hash untuk mengamankan password atau data sensitif lainnya dengan cara meng-hash (mengenkripsi) nilai tersebut**.**
    - Maksud dari kode program Hash::make('1234'); berfungsi untuk mengenkripsi password menggunakan bcrypt**.**
10. Pada Praktikum 4 - Tahap 3/5/7, pada query builder terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?
    - Dalam Laravel Query Builder, tanda tanya (?) digunakan sebagai placeholder untuk parameter binding (teknik untuk menggantikan nilai dalam query SQL) sehingga mencegah SQL Injection & meningkatkan keamanan.

11. Pada Praktikum 6 - Tahap 3, apa tujuan penulisan kode protected $table = ‘m_user’; dan protected $primaryKey = ‘user_id’; ?
    - protected $table = ‘m_user’; = Memberitahu Laravel bahwa model ini menggunakan tabel m_user.
    - protected $primaryKey = ‘user_id’; = Memberitahu Laravel bahwa user_id adalah primary key tabel ini
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (DB Façade / Query Builder / Eloquent ORM) ? jelaskan
    - Menurut saya, jika melakukan operasi CRUD ke database lebih mudah menggunakan Eloquent ORM karena secara konsep seperti OOP, kemudian mendukung fitur Laravel seperti timestamps, dan lebih aman karena encrypt sudah bisa menggunakan hash.
    - Walaupun sedikit lebih lambat dibanding Query Builder dan DB Façade.