

**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**

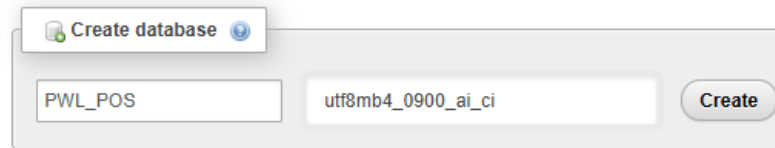
**2024**

## A. PENGATURAN DATABASE

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

### 1. Praktikum 1 – Pengaturan Database:

- a. Buka aplikasi phpMyAdmin, dan buat database baru dengan nama PWL\_POS



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

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan key:generate
INFO Application key set successfully.
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> |
APP_KEY=base64:gnj4qusoQ9QHwTQqiNcJIk3vKd4Fb+0QYsSRb3g1UTQ=
```

- e. 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=
```

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

## B. MIGRATION

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

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

### TIPS MIGRATION

Buatlah file migration untuk table yang tidak memiliki relasi (table yang tidak ada *foreign key*) dulu, dan dilanjutkan dengan membuat file migrasi yang memiliki relasi yang sedikit, dan dilanjutkan ke file migrasi dengan table yang memiliki relasi yang banyak.

### INFO

Secara default Laravel sudah ada table **users** untuk menyimpan data pengguna, tapi pada praktikum ini, kita gunakan table sesuai dari file *Studi Kasus PWL.pdf* yaitu **m\_user**.

Pembuatan file migrasi bisa menggunakan 2 cara, yaitu:

- a. Menggunakan artisan untuk membuat file migration

```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

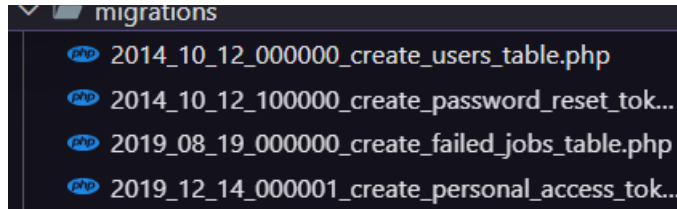
- b. Menggunakan artisan untuk membuat file model + file migration

```
php artisan make:model <nama-model> -m
```

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

## 1. Praktikum 2.1 – Pembuatan File Migrasi Tanpa Relasi

- Buka terminal VSCode kalian, untuk yang di kotak merah adalah default dari laravel



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

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:migration create_m_level_table --create level
```

**INFO** Migration [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\migrations\2025\_03\_04\_154548\_create\_m\_level\_table.php] created successfully.

```
database > migrations > 2025_03_04_154548_create_m_level_table.php > class > down
```

```
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create(table: 'm_level', callback: function (Blueprint $table): void {
15             $table->id();
16             $table->timestamps();
17         });
18     }
19
20     /**
21      * Reverse the migrations.
22      */
23     public function down(): void
24     {
25         Schema::dropIfExists(table: 'level');
26     }
27 };
28
```

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

```

database > migrations > 2025_03_04_154548_create_m_level_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::create(table: 'm_level', callback: function (Blueprint $table): void {
15               $table->id(column: 'level_id');
16               $table->string(column: 'level_kode', length: 10)->unique();
17               $table->string(column: 'level_nama', length: 100);
18               $table->timestamps();
19           });
20       }
21
22       /**
23        * Reverse the migrations.
24        */
25       public function down(): void
26       {
27           Schema::dropIfExists(table: 'level');
28       }
29  };

```

### INFO

Dalam fitur migration Laravel, terdapat berbagai macam function untuk membuat kolom di table database. Silahkan cek disini

<https://laravel.com/docs/10.x/migrations#available-column-types>

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

```

PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan migrate

INFO Preparing database.

Creating migration table ..... 33ms DONE

INFO Running migrations.

2014_10_12_000000_create_users_table ..... 32ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 11ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 31ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 37ms DONE
2025_03_04_104926_create_products_table ..... 11ms DONE
2025_03_04_154548_create_m_level_table ..... 36ms DONE

```

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

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> failed_jobs	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> m_level	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> password_reset_tokens	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> personal_access_tokens	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> products	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
7 tables	Sum	6	InnoDB	utf8mb4_0900_ai_ci	112.0 KiB	0 B

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

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:migration create_m_kategori_table
● te_m_kategori_table

INFO Migration [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\migrations\2025_03_04_155559_create_m_kategori_table.php] created successfully.

database > migrations > 2025_03_04_155559_create_m_kategori_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create(table: 'm_kategori', callback: function (Blueprint $table): void {
15             $table->id(column: 'kategori_id');
16             $table->string(column: 'kategori_kode', length: 10)->unique();
17             $table->string(column: 'kategori_nama', length: 100);
18             $table->timestamps();
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      */
25     public function down(): void
26     {
27         Schema::dropIfExists(table: 'm_kategori');
28     }
29 };
```

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan migrate

INFO Running migrations.

2025_03_04_155559_create_m_kategori_table 29ms DONE
```

Table	Action	Rows	Type	Collation	Size	Overhead
<input type="checkbox"/> failed_jobs		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
<input type="checkbox"/> migrations		7	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
<input type="checkbox"/> m_kategori		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
<input type="checkbox"/> m_level		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
<input type="checkbox"/> password_reset_tokens		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
<input type="checkbox"/> personal_access_tokens		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
<input type="checkbox"/> products		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
<input type="checkbox"/> users		0	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
8 tables	Sum	7	InnoDB	utf8mb4_0900_ai_ci	128.0 Kib	0 B

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

## 2. Praktikum 2.2 – Pembuatan File Migrasi dengan Relasi

- a. Buka terminal VSCode kalian, dan buat file migrasi untuk table m\_user

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:migration create_m_user_table --table=m_user

INFO Migration [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\migrations\2025_03_04_202949_create_m_user_table.php] created successfully.
```

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

```
database > migrations > 2025_03_04_202949_create_m_user_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12      public function up(): void
13      {
14          Schema::create(table: 'm_user', callback: function (Blueprint $table): void {
15              $table->id(column: 'user_id');
16              $table->unsignedBigInteger(column: 'level_id')->index(); //indexing foreign key
17              $table->string(column: 'username', length: 20)->unique(); //unique memastikan data tidak duplikat
18              $table->string(column: 'nama', length: 100);
19              $table->string(column: 'password', length: 100);
20              $table->timestamps();
21
22              // mendefinisika fk level_id yang merujuk ke kolom level_id di tabel m_level
23              $table->foreign(columns: 'level_id')->references(columns: 'level_id')->on(table: 'm_level');
24          });
25      }
26
27      /**
28       * Reverse the migrations.
29       */
30      public function down(): void
31      {
32          Schema::dropIfExists(table: 'm_user');
33      }
34  };
```

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

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan migrate

INFO Running migrations.

2025_03_04_202949_cretae_m_user_table ..... 73ms DONE
```

**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
  - 2) 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.
- d. Buat table database dengan migration untuk table-table yang memiliki foreign key

m_barang
t_penjualan
t_stok
t_penjualan_detail

**Jawab:**



## 1) M\_barang

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:migration create_m_barang_table
● ble

[INFO] Migration [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\Migrations\2025_03_04_205541_create_m_barang_table.php] created successfully.

database > migrations > 2025_03_04_205541_create_m_barang_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create(table: 'm_barang', callback: function (Blueprint $table): void {
15             $table->id(column: 'barang_id');
16             $table->unsignedBigInteger(column: 'kategori_id')->index();
17             $table->string(column: 'barang_kode', length: 20)->unique();
18             $table->string(column: 'barang_nama', length: 100);
19             $table->decimal(column: 'harga', total: 15, places: 2);
20             $table->timestamps();
21
22             $table->foreign(columns: 'kategori_id')->references(columns: 'kategori_id')->on(table: 'm_kategori');
23         });
24     }
25
26     /**
27      * Reverse the migrations.
28      */
29     public function down(): void
30     {
31         Schema::dropIfExists(table: 'm_barang');
32     }
33 };

● PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan migrate

[INFO] Running migrations.

2025_03_04_205541_create_m_barang_table ..... 68ms DONE
```

## 2) T\_penjualan

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:migration create_t_penjualan_table
● _table

[INFO] Migration [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\Migrations\2025_03_04_205823_create_t_penjualan_table.php] created successfully.
```

```
database > migrations > 2025_03_04_205823_create_t_penjualan_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12      public function up(): void
13      {
14          Schema::create(table: 't_penjualan', callback: function (Blueprint $table): void {
15              $table->id(column: 'penjualan_id');
16              $table->unsignedBigInteger(column: 'user_id')->index();
17              $table->date(column: 'tanggal');
18              $table->decimal(column: 'total', total: 15, places: 2);
19              $table->timestamps();
20
21              $table->foreign(columns: 'user_id')->references(columns: 'user_id')->on(table: 'm_user');
22          });
23      }
24
25      /**
26       * Reverse the migrations.
27       */
28      public function down(): void
29      {
30          Schema::dropIfExists(table: 't_penjualan');
31      }
32  });

PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan migrate
INFO Running migrations.
2025_03_04_205823_create_t_penjualan_table ..... 63ms DONE
```

### 3) T\_stok

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:migration create_t_stok_table
e
INFO Migration [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\migrations\2025_03_04_210019_create_t_stok_table.php] created successfully.
```

```
database > migrations > 2025_03_04_210019_create_t_stok_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::create(table: 't_stok', callback: function (Blueprint $table): void {
15               $table->id(column: 'stok_id');
16               $table->unsignedBigInteger(column: 'barang_id')->index();
17               $table->integer(column: 'jumlah');
18               $table->timestamps();
19
20               $table->foreign(columns: 'barang_id')->references(columns: 'barang_id')->on(table: 'm_barang');
21           });
22       }
23
24       /**
25        * Reverse the migrations.
26        */
27       public function down(): void
28       {
29           Schema::dropIfExists(table: 't_stok');
30       }
31  };

PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan migrate
INFO Running migrations.

2025_03_04_210019_create_t_stok_table ..... 57ms DONE
```

#### 4) T\_penjualan\_detail

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:migration create_t_penjualan_detail_table
INFO Migration [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\Migrations\2025_03_04_210222_create_t_penjualan_detail_table.php] created successfully.

database > migrations > 2025_03_04_210222_create_t_penjualan_detail_table.php > ...
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::create(table: 't_penjualan_detail', callback: function (Blueprint $table): void {
15               $table->id(column: 'penjualan_detail_id');
16               $table->unsignedBigInteger(column: 'penjualan_id')->index();
17               $table->unsignedBigInteger(column: 'barang_id')->index();
18               $table->integer(column: 'jumlah');
19               $table->decimal(column: 'harga', total: 15, places: 2);
20               $table->timestamps();
21
22               $table->foreign(columns: 'penjualan_id')->references(columns: 'penjualan_id')->on(table: 't_penjualan');
23               $table->foreign(columns: 'barang_id')->references(columns: 'barang_id')->on(table: 'm_barang');
24           });
25       }
26
27       /**
28        * Reverse the migrations.
29        */
30       public function down(): void
31       {
32           Schema::dropIfExists(table: 't_penjualan_detail');
33       }
34  };

PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan migrate
INFO Running migrations.

2025_03_04_210222_create_t_penjualan_detail_table ..... 101ms DONE
```

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



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

## C. SEEDER

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

1. Perintah umum dalam **membuat file seeder** adalah seperti berikut

```
php artisan make:seeder <nama-class-seeder>
```

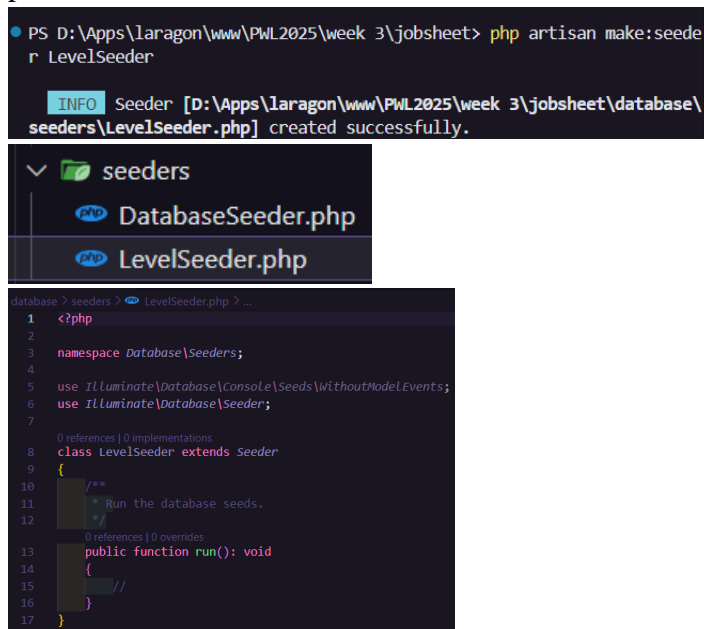
Perintah tersebut akan men-generate file seeder pada folder **PWL\_POS/database/seeder**s

2. Dan perintah untuk **menjalankan file seeder** seperti berikut

```
php artisan db:seed --class=<nama-class-seeder>
```

### 1. Praktikum 3 – Membuat File Seeder

- a. Kita akan membuat file seeder untuk table m\_level dengan mengetikkan perintah



The screenshot shows a terminal window and an IDE. The terminal displays the command `php artisan make:seeder LevelSeeder` and its output: `INFO Seeder [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\seeders\LevelSeeder.php] created successfully.` Below the terminal, the IDE shows the file explorer with the `seeders` folder containing `DatabaseSeeder.php` and `LevelSeeder.php`. The code editor shows the content of `LevelSeeder.php`, which includes the namespace `Database\Seeders`, imports `Illuminate\Database\Console\Seeds\WithoutModelEvents` and `Illuminate\Database\Seeder`, and defines the `LevelSeeder` class extending `Seeder` with a `run()` method.

```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7
8 0 references | 0 implementations
9 class LevelSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     0 references | 0 overrides
15     public function run(): void
16     {
17         //
18     }
```

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

```
database > seeders > LevelSeeder.php > ...
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  0 references | 0 implementations
10 class LevelSeeder extends Seeder
11 {
12     /**
13      * Run the database seeds.
14      */
15     0 references | 0 overrides
16     public function run(): void
17     {
18         $data = [
19             ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
20             ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
21             ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
22         ];
23         DB::table('m_level')->insert(values: $data);
24     }
25 }
```

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

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan db:seed --class=LevelSeeder

INFO Seeding database.
```

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

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit Copy Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit Copy Delete	3	STF	Staff/Kasir	NULL	NULL

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

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:seeder UserSeeder

INFO Seeder [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\seeders\UserSeeder.php] created successfully.
```

f. Modifikasi file class UserSeeder seperti berikut

```
database > seeders > UserSeeder.php > PHP Intelephense > UserSeeder > run
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB; // Tambahkan ini
8 use Illuminate\Support\Facades\Hash;
9
10 class UserSeeder extends Seeder
11 {
12     /**
13      * Run the database seeds.
14      */
15     public function run(): void
16     {
17         $data = [
18             [
19                 'user_id' => 1,
20                 'level_id' => 1,
21                 'username' => 'admin',
22                 'nama' => 'Administrator',
23                 'password' => Hash::make(value: '12345'),
24             ],
25             [
26                 'user_id' => 2,
27                 'level_id' => 2,
28                 'username' => 'manager',
29                 'nama' => 'Manager',
30                 'password' => Hash::make(value: '12345'),
31             ],
32             [
33                 'user_id' => 3,
34                 'level_id' => 3,
35                 'username' => 'staff',
36                 'nama' => 'Staff/Kasir',
37                 'password' => Hash::make(value: '12345'),
38             ],
39         ];
40         DB::table('m_user')->insert(values: $data);
41     }
42 }
```

g. Jalankan perintah untuk mengeksekusi class UserSeeder

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan db:seed --class=UserSeeder

INFO Seeding database.

PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet>
```

h. Perhatikan hasil seeder pada table m\_user

	user_id	level_id	username	nama	password
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$10\$OVsKrzJaY07YrmGm5kfqXuuVkm88v3LALstjLKZ7Fmo...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$10\$W5L99lcvSuaDckIjKkeySErXy4R0rcdN7leftH3ql...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$10\$SyhvJJlqYEuYfSEybKupMeNDWyGmEZPcud.1.Sz29I4...

- i. Ok, data seeder berhasil di masukkan ke database.
- j. Sekarang coba kalian masukkan data seeder untuk table yang lain, dengan ketentuan seperti berikut

1) Seeder untuk kategori (5 kategori barang)

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:seeder KategoriSeeder

INFO Seeder [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\seeders\KategoriSeeder.php] created successfully.
```

2) Buka database/seeders/KategoriSeeder.php dan tambahkan:

```
database / seeders / KategoriSeeder.php / ...
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriSeeder extends Seeder
9  {
10     public function run(): void
11     {
12         DB::table('m_kategori')->insert(values: [
13             ['kategori_id' => 1, 'kategori_kode' => 'KTG001', 'kategori_nama' => 'Elektronik'],
14             ['kategori_id' => 2, 'kategori_kode' => 'KTG002', 'kategori_nama' => 'Pakaian'],
15             ['kategori_id' => 3, 'kategori_kode' => 'KTG003', 'kategori_nama' => 'Makanan'],
16             ['kategori_id' => 4, 'kategori_kode' => 'KTG004', 'kategori_nama' => 'Minuman'],
17             ['kategori_id' => 5, 'kategori_kode' => 'KTG005', 'kategori_nama' => 'Obat-obatan'],
18         ]);
19     }
20 }
```

3) Seeder untuk m\_barang (10 Barang Berbeda)

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:seeder BarangSeeder

INFO Seeder [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\seeders\BarangSeeder.php] created successfully.
```

4) Buka database/seeders/BarangSeeder.php dan tambahkan:



```

1  <?php
2
3  namespace database\users;
4
5  use Illuminate\Database\Eloquent;
6  use Illuminate\Support\Facades\DB;
7
8  class Barang extends Model
9  {
10     public function run(): void
11     {
12         DB::table('barang')->insertIf
13         [
14             'barang_id' => 1,
15             'kategori_id' => 1,
16             'barang_kode' => 'R00001',
17             'barang_nama' => 'Agropro ATIS 400',
18             'harga_beli' => 15.000.000,
19             'harga_jual' => 17.500.000
20         ],
21         [
22             'barang_id' => 2,
23             'kategori_id' => 1,
24             'barang_kode' => 'R00002',
25             'barang_nama' => 'Makina AT 40',
26             'harga_beli' => 18.000.000,
27             'harga_jual' => 20.500.000
28         ],
29         [
30             'barang_id' => 3,
31             'kategori_id' => 1,
32             'barang_kode' => 'R00003',
33             'barang_nama' => 'Makina 22 inch 1hp',
34             'harga_beli' => 3.500.000,
35             'harga_jual' => 4.200.000
36         ],
37         [
38             'barang_id' => 4,
39             'kategori_id' => 2,
40             'barang_kode' => 'R00004',
41             'barang_nama' => 'Kaca Polos Kutan',
42             'harga_beli' => 20.000,
43             'harga_jual' => 25.000
44         ],
45         [
46             'barang_id' => 5,
47             'kategori_id' => 2,
48             'barang_kode' => 'R00005',
49             'barang_nama' => 'Jacket Domba',
50             'harga_beli' => 200.000,
51             'harga_jual' => 250.000
52         ],
53         [
54             'barang_id' => 6,
55             'kategori_id' => 2,
56             'barang_kode' => 'R00006',
57             'barang_nama' => 'Kopora Speakers',
58             'harga_beli' => 400.000,
59             'harga_jual' => 500.000
60         ],
61         [
62             'barang_id' => 7,
63             'kategori_id' => 2,
64             'barang_kode' => 'R00007',
65             'barang_nama' => 'Kopi Tawar',
66             'harga_beli' => 15.000,
67             'harga_jual' => 20.000
68         ],
69         [
70             'barang_id' => 8,
71             'kategori_id' => 2,
72             'barang_kode' => 'R00008',
73             'barang_nama' => 'Piscesir Gendak',
74             'harga_beli' => 55.000,
75             'harga_jual' => 55.000
76         ],
77         [
78             'barang_id' => 9,
79             'kategori_id' => 2,
80             'barang_kode' => 'R00009',
81             'barang_nama' => 'Mie Instan',
82             'harga_beli' => 5.500,
83             'harga_jual' => 6.000
84         ],
85         [
86             'barang_id' => 10,
87             'kategori_id' => 4,
88             'barang_kode' => 'R00010',
89             'barang_nama' => 'Ayu Kernal 1l',
90             'harga_beli' => 5.000,
91             'harga_jual' => 6.000
92         ],
93         [
94             'barang_id' => 11,
95             'kategori_id' => 4,
96             'barang_kode' => 'R00011',
97             'barang_nama' => 'Tas Benua Hitam',
98             'harga_beli' => 12.000,
99             'harga_jual' => 18.000
100        ],
101        [
102            'barang_id' => 12,
103            'kategori_id' => 5,
104            'barang_kode' => 'R00012',
105            'barang_nama' => 'Kopi Robuk 20kg',
106            'harga_beli' => 38.000,
107            'harga_jual' => 48.000
108        ],
109        [
110            'barang_id' => 13,
111            'kategori_id' => 5,
112            'barang_kode' => 'R00013',
113            'barang_nama' => 'Panaswanti 10kg',
114            'harga_beli' => 2.500,
115            'harga_jual' => 3.000
116        ],
117        [
118            'barang_id' => 14,
119            'kategori_id' => 5,
120            'barang_kode' => 'R00014',
121            'barang_nama' => 'Vilawla C 100kg',
122            'harga_beli' => 10.000,
123            'harga_jual' => 15.000
124        ],
125        [
126            'barang_id' => 15,
127            'kategori_id' => 5,
128            'barang_kode' => 'R00015',
129            'barang_nama' => 'Antipatik Cair 100ml',
130            'harga_beli' => 10.000,
131            'harga_jual' => 15.000
132        ],
133    ];
134 }
135 }

```

5) Seeder untuk t\_stok (10 Stok untuk 10 Barang)

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:seeder StokSeeder

INFO Seeder [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\seeders\StokSeeder.php]
created successfully.
```

6) Buka database/seeders/StokSeeder.php dan tambahkan:

```
database / seeders > StokSeeder.php > PHP > StokSeeder
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Seeder;
6 use Illuminate\Support\Facades\DB;
7
8 0 references | 0 implementations
9 class StokSeeder extends Seeder
10 {
11     0 references | 0 overrides
12     public function run(): void
13     {
14         $barangIds = DB::table('m_barang')->pluck('id')->toArray();
15
16         foreach ($barangIds as $barangId) {
17             DB::table('t_stok')->insert(values: [
18                 'barang_id' => $barangId,
19                 'jumlah_stok' => rand(min: 50, max: 200), // Stok acak antara 50-200 untuk contoh
20                 'created_at' => now(),
21                 'updated_at' => now()
22             ]);
23     }
24 }
```

7) Seeder untuk t\_penjualan (10 Transaksi Penjualan)

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:seeder PenjualanSeeder

INFO Seeder [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\seeders\PenjualanSeeder.
php] created successfully.
```

8) Buka database/seeders/PenjualanSeeder.php dan tambahkan:

```

database > seeders > PenjualanSeeder.php > PHP Intelephense > PenjualanSeeder > run
1 <?php
2
3 namespace Database\Seeders;
4
5 use Carbon\Carbon;
6 use Faker\Factory;
7 use Illuminate\Database\Seeder;
8 use Illuminate\Support\Facades\DB;
9
10 0 references | 0 implementations
11 class PenjualanSeeder extends Seeder
12 {
13     0 references | 0 overrides
14     public function run(): void
15     {
16         $penjualan = [];
17         for ($i = 1; $i <= 15; $i++) {
18             $penjualan[] = [
19                 'penjualan_id' => $i,
20                 'pembeli' => Factory::create()->unique()->name(),
21                 'penjualan_kode' => Factory::create()->unique()->word(),
22                 'penjualan_tanggal' => Carbon::now()->subDays(value: rand(min: 1, max: 30))->toDateString(),
23                 'user_id' => rand(min: 1, max: 3),
24             ];
25         }
26         DB::table(table: 't_penjualan')->insert(values: $penjualan);
27     }
28 }

```

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

```

PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:seeder PenjualanDetailSeeder
INFO Seeder [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\database\seeders\PenjualanDetailSeeder.php] created successfully.

```

10) Buka database/seeders/PenjualanDetailSeeder.php dan tambahkan:

```

database > seeders > PenjualanDetailSeeder.php > PHP Intelephense > PenjualanDetailSeeder
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Seeder;
6 use Illuminate\Support\Facades\DB;
7
8 0 references | 0 implementations
9 class PenjualanDetailSeeder extends Seeder
10 {
11     0 references | 0 overrides
12     public function run(): void
13     {
14         $data = [];
15         for ($i = 1; $i <= 10; $i++) {
16             for ($j = 1; $j <= 3; $j++) {
17                 $data[] = [
18                     'penjualan_id' => $i,
19                     'barang_id' => $j,
20                     'harga' => random_int(min: 10000, max: 50000),
21                     'jumlah' => random_int(min: 1, max: 10),
22                 ];
23             }
24         }
25         DB::table(table: 't_penjualan_detail')->insert(values: $data);
26     }
27 }

```

11) Buka database seeders/LevelSeeder.php dan tambahkan:

```

database > seeders > LevelSeeder.php > PHP Intelephense > LevelSeeder
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Seeder;
6  use Illuminate\Support\Facades\DB;
7
8  0 references | 0 implementations
9  class LevelSeeder extends Seeder
10
11     0 references | 0 overrides
12     public function run(): void
13     {
14         DB::table('m_level')->insert([
15             ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
16             ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
17             ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staf/Kasir'],
18         ]);
19     }

```

## 12) Menjalankan Seeder

```

PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan db:seed

```

```

INFO Seeding database.

```

## D. DB FACADES

DB Façade merupakan fitur dari Laravel yang digunakan untuk melakukan query secara langsung dengan mengetikkan perintah 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 ...”

### INFO

Dokumentasi penggunaan DB Façade bisa dicek di laman ini

<https://laravel.com/docs/10.x/database#running-queries>

### 1. Praktikum 4 – Implementasi DB Façade

- a. Kita buat controller dahulu untuk mengelola data pada table m\_level

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make :controller LevelController
```

```
INFO Controller [D:\Apps\laragon\www\PWL2025\week 3\jobsheet \app\Http\Controllers\LevelController.php] created successfully.
```

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

```
routes > web.php > ...
You, 11 seconds ago | 1 author (You)
1 <?php
2
3 use Illuminate\Support\Facades\Route;
4 use App\Http\Controllers\HomeController;
5 use App\Http\Controllers\ProductController;
6 use App\Http\Controllers\ProfileController; You, 12 hours ago • add: setting database
7 use App\Http\Controllers\TransactionController;
8 use App\Http\Controllers\LevelController;
9
10 Route::get(uri: '/', action: [HomeController::class, 'index'])->name(name: 'home');
11
12 Route::prefix(prefix: '/category')->group(callback: function (): void {
13     Route::get(uri: '/food-beverage', action: [ProductController::class, 'foodBeverage']);
14     Route::get(uri: '/beauty-health', action: [ProductController::class, 'beautyHealth']);
15     Route::get(uri: '/home-care', action: [ProductController::class, 'homeCare']);
16     Route::get(uri: '/baby-kid', action: [ProductController::class, 'babyKid']);
17 });
18
19 Route::get(uri: '/user/{id}/name/{name}', action: [ProfileController::class, 'index']);
20
21 Route::get(uri: '/transaction', action: [TransactionController::class, 'index']);
22 Route::get(uri: '/level', action: [LevelController::class, 'index']);
23
```

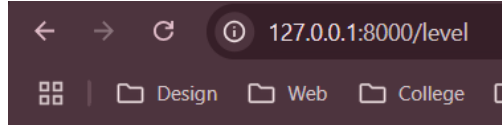
- c. Selanjutnya, kita modifikasi file LevelController untuk menambahkan 1 data ke table m\_level

```



1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function insertlevel(): void
11     {
12         DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', bindings: ['CUS', 'Pelanggan', now()]);
13     }
14 }

```

- d. 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



Insert data baru berhasil

				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	STF	Staf/Kasir	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	CUS	Pelanggan	2025-03-05 03:56:06	NULL

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

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     public function index(): string
11     {
12         // DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13         // return 'Insert data baru berhasil';
14
15         $row = DB::update('update m_level set level_nama = ? where level_id = ?', bindings: ['Customer', 'CUS']);
16         return 'Update data berhasil. Jumlah data yg diupdate: ' . $row;
17     }
18 }

```

- f. 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

		level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	STF	Staf/Kasir	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	CUS	Customer	2025-03-05 03:56:06	NULL

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

```

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     // 1 reference | 0 overrides
11     public function index(): string
12     {
13         // DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', ['CUS', 'Pelanggan', now()]);
14         // return 'Insert data baru berhasil';
15
16         // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
17         // return 'Update data berhasil. Jumlah data yg diupdate: ' . $row;
18
19         $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
20         return 'Delete data berhasil. Jumlah data yg dihapus: ' . $row . ' baris';
21     }
22 }

```

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

```

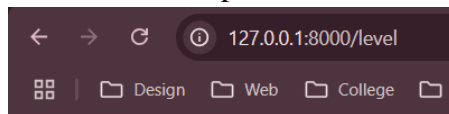
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     // 1 reference | 0 overrides
11     public function index(): Factory\View
12     {
13         // DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', ['CUS', 'Pelanggan', now()]);
14         // return 'Insert data baru berhasil';
15
16         // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
17         // return 'Update data berhasil. Jumlah data yg diupdate: ' . $row;
18
19         // $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
20         // return 'Delete data berhasil. Jumlah data yg dihapus: ' . $row . ' baris';
21
22         $data = DB::select('select * from m_level');
23         return view('level', data: ['data' => $data]);
24     }
25 }

```

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

```
resources > views > level.blade.php > html
1 <!DOCTYPE html>
2 <html Lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7     <title>Data level pengguna</title>
8 </head>
9 <body>
10    <h1>Data Level Pengguna</h1>
11    <table border="1" cellpadding="2" cellspacing="0">
12        <tr>
13            <th>ID</th>
14            <th>Kode Level</th>
15            <th>Nama Level</th>
16        </tr>
17        @foreach ($data as $d)
18            <tr>
19                <td>{{ $d->level_id }}</td>
20                <td>{{ $d->level_kode }}</td>
21                <td>{{ $d->level_nama }}</td>
22            </tr>
23        @endforeach
24    </table>
25 </body>
26 </html>
```

- j. 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	Staf/Kasir
4	CUS	Customer

**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.

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



## E. QUERY BUILDER

### 1. Praktikum 5 - Implementasi Query Builder

- a. Kita buat controller dahulu untuk mengelola data pada table m\_kategori

```
PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php
artisan make:controller KategoriController

[INFO] Controller [D:\Apps\laragon\www\PWL2025\week
3\jobsheet\app\Http\Controllers\KategoriController
.php] created successfully.
```

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

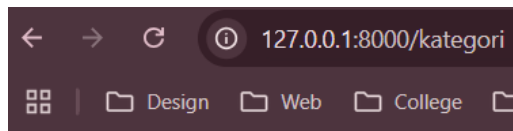
```
Route::get(uri: '/level', action: [LevelController::class, 'index']);
Route::get(uri: '/kategori', action: [KategoriController::class, 'index']);
```

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

```
app > Http > Controllers > KategoriController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController extends Controller
9  {
10     public function insertData(): string
11     {
12         $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now(),
16         ];
17         DB::table(table: 'm_kategori')->insert(values: $data);
18         return 'Insert data baru berhasil';
19     }
20 }
```

- d. 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:**



Insert data baru berhasil

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	SNK	Snack/Makanan Ringan	2025-03-05 07:59:49	NULL

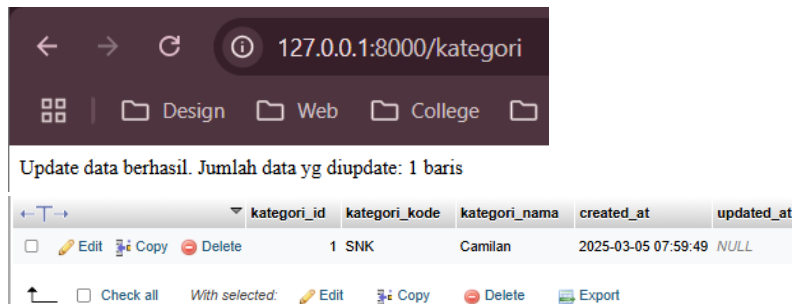
Controller ini digunakan untuk memasukkan data baru ke dalam tabel m\_kategori melalui metode index. Saat metode index dipanggil, data baru akan dimasukkan ke dalam tabel m\_kategori, dan pesan sukses akan dikembalikan.

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

```
app > Http > Controllers > KategoriController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  2 references | 0 implementations
9  class KategoriController extends Controller
10 {
11     0 references | 0 overrides
12     public function insertData(): string
13     {
14         // $data = [
15         //     'kategori_kode' => 'SNK',
16         //     'kategori_nama' => 'Snack/Makanan Ringan',
17         //     'created_at' => now(),
18         // ];
19         // DB::table('m_kategori')->insert($data);
20         // return 'Insert data baru berhasil';
21
22         $row = DB::table('m_kategori')->where(column: 'kategori_kode', operator: 'SNK')->update(value:
23         'kategori_nama' => 'Camilan');
24         return 'Update data berhasil. Jumlah data yg diupdate: ' . $row . ' baris';
25     }
26 }
```

- f. 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:**



Update data berhasil. Jumlah data yg diupdate: 1 baris

kategori_id	kategori_kode	kategori_nama	created_at	updated_at
1	SNK	Camilan	2025-03-05 07:59:49	NULL

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'.

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

```
app > Http > Controllers > KategoriController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController extends Controller
9  {
10     1 reference | 0 overrides
11     public function index(): string
12     {
13         // $data = [
14         //     'kategori_kode' => 'SNK',
15         //     'kategori_nama' => 'Snack/Makanan Ringan',
16         //     'created_at' => now(),
17         // ];
18         // DB::table('m_kategori')->insert($data);
19         // return 'Insert data baru berhasil';
20
21         // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update([
22         //     'kategori_nama' => 'Camilan']);
23         // return 'Update data berhasil. Jumlah data yg diupdate: ' . $row . ' baris';
24
25         $row= DB::table('m_kategori') -> where(column: 'kategori_kode', operator: 'SNK')->delete();
26         return 'Delete data berhasil. Jumlah data yg dihapus: ' . $row . ' baris';
27     }
28 }
```

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

```

app > Http > Controllers > KategoriController.php > PHP Intelephense > KategoriController > index
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class KategoriController extends Controller
9 {
10     1 reference | 0 overrides
11     public function index(): Factory|View
12     {
13         // $data = [
14         //     'kategori_kode' => 'SNK',
15         //     'kategori_nama' => 'Snack/Makanan Ringan',
16         //     'created_at' => now(),
17         // ];
18         // DB::table('m_kategori')->insert($data);
19         // return 'Insert data baru berhasil';
20
21         // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update([
22         //     'kategori_nama' => 'Camilan']);
23         // return 'Update data berhasil. Jumlah data yg diupdate: ' . $row. ' baris';
24
25         // $row = DB::table('m_kategori') -> where('kategori_kode', 'SNK')->delete();
26         // return 'Delete data berhasil. Jumlah data yg dihapus: ' . $row. ' baris';
27
28         $data = DB::table('m_kategori')->get();
29         return view('kategori', data: ['data' => $data]);
30     }
31 }

```

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

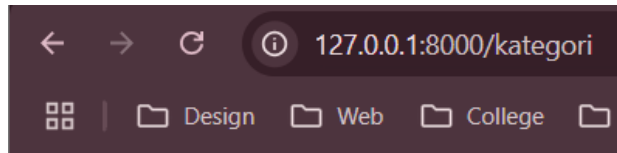
```

resources > views > kategori.blade.php > html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7     <title>Data Kategori Barang</title>
8 </head>
9 <body>
10     <h1>Data Kategori Barang</h1>
11     <table border="1" cellpadding="2" cellspacing="0">
12         <tr>
13             <th>ID</th>
14             <th>Kode Kategori</th>
15             <th>Nama Kategori</th>
16         </tr>
17         @foreach ($data as $d)
18             <tr>
19                 <td>{{ $d->kategori_id }}</td>
20                 <td>{{ $d->kategori_kode }}</td>
21                 <td>{{ $d->kategori_nama }}</td>
22             </tr>
23         @endforeach
24     </table>
25 </body>
26 </html>

```

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

**Jawab:**



## Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	SNK	Camilan

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

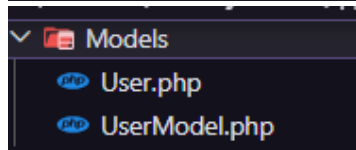
## F. ELOQUENT ORM

### 1. Praktikum 6 – Implementasi Eloquent ORM

- a. Kita buat file model untuk tabel m\_user dengan mengetikkan perintah

```
PS D:\Apps\laragon\www\PhL2025\week 3\jobsheet> php artisan make:model UserModel
1

INFO Model [D:\Apps\laragon\www\PhL2025\week 3\jobsheet\app\Models\UserModel
1.php] created successfully.
```



- b. Setelah berhasil generate model, terdapat 2 file pada folder model yaitu file User.php bawaan dari laravel dan file UserModel.php yang telah kita buat. Kali ini kita akan menggunakan file UserModel.php
- c. Kita buka file UserModel.php dan modifikasi seperti berikut

```
app > Models > UserModel.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  0 references | 0 implementations
9  class UserModel extends Model
10 {
11     use HasFactory;
12
13     0 references
14     protected $table = 'users'; //mendefinisikan nama tabel yg digunakan oleh model i
15
16     0 references
17     protected $primaryKey = 'id'; //mendefinisikan pk dari tabel yg digunakan
18 }
```

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

```

routes > web.php > ...
You, 2 minutes ago | 1 author (You)
1  <?php
2
3  use Illuminate\Support\Facades\Route;
4  use App\Http\Controllers\HomeController;
5  use App\Http\Controllers\ProductController;
6  use App\Http\Controllers\ProfileController;
7  use App\Http\Controllers\TransactionController;
8  use App\Http\Controllers\LevelController;
9  use App\Http\Controllers\KategoriController;
10 use App\Http\Controllers\UserController;
11
12 Route::get(uri: '/', action: [HomeController::class, 'index'])->name(name: 'home');
13
14 Route::prefix(prefix: '/category')->group(callback: function (): void {
15     Route::get(uri: '/food-beverage', action: [ProductController::class, 'foodBeverage']);
16     Route::get(uri: '/beauty-health', action: [ProductController::class, 'beautyHealth']);
17     Route::get(uri: '/home-care', action: [ProductController::class, 'homeCare']);
18     Route::get(uri: '/baby-kid', action: [ProductController::class, 'babyKid']);
19 });
20
21 Route::get(uri: '/user/{id}/name/{name}', action: [ProfileController::class, 'index']);
22
23 Route::get(uri: '/transaction', action: [TransactionController::class, 'index']);
24 Route::get(uri: '/level', action: [LevelController::class, 'index']);
25 Route::get(uri: '/kategori', action: [KategoriController::class, 'index']);
26 Route::get(uri: '/user', action: [UserController::class, 'index']); Undefined type 'App\

```

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

```

PS D:\Apps\laragon\www\PWL2025\week 3\jobsheet> php artisan make:controller Use
rController

INFO Controller [D:\Apps\laragon\www\PWL2025\week 3\jobsheet\app\Http\Contr
ollers\UserController.php] created successfully.

```

```

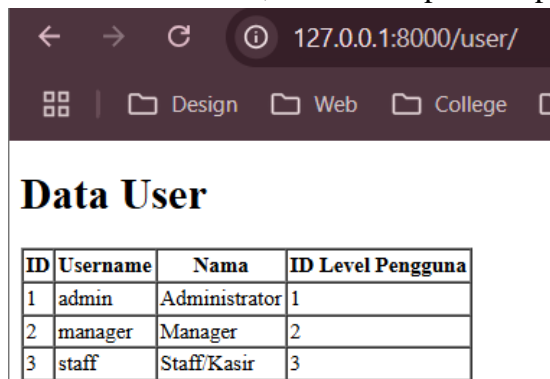
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\UserModel;
7  use Illuminate\Support\Facades\DB;
8
9  2 references | 0 implementations
10 class UserController extends Controller
11 {
12     //coba akses model UserModel
13     1 reference | 0 overrides
14     public function index(): Factory|View
15     {
16         $user = UserModel::all();
17         // Ambil data dari tabel m_user
18         $data = DB::table(table: 'm_user')->get();
19         return view(view: 'user', data: ['data' => $data]);
20     }
21 }

```

f. Kemudian kita buat view user.blade.php

```
resources > views > user.blade.php > html > body > table > tr > td
1 <!DOCTYPE html>
2 <html Lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1.0">
6     <meta http-equiv="X-UA-Compatible" content="ie=edge">
7     <title>Data User</title>
8 </head>
9 <body>
10     <h1>Data User</h1>
11     <table border="1" cellpadding="2" cellspacing="0">
12         <tr>
13             <th>ID</th>
14             <th>Username</th>
15             <th>Nama</th>
16             <th>ID Level Pengguna</th>
17         </tr>
18         @foreach ($data as $d)
19             <tr>
20                 <td>{{ $d->user_id }}</td>
21                 <td>{{ $d->username }}</td>
22                 <td>{{ $d->nama }}</td>
23                 <td>{{ $d->level_id }}</td>
24             </tr>
25         @endforeach
26     </table>
27 </body>
28 </html>
```

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



← → ↻ ⓘ 127.0.0.1:8000/user/

Design Web College

## Data User

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

h. Setelah itu, kita modifikasi lagi file UserController

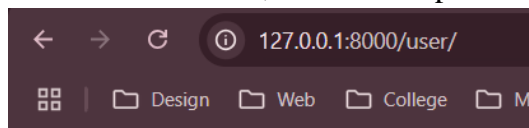


```

app > Http > Controllers > UserController.php > PHP Intelephense > UserController > index
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use App\Models\UserModel;
7  use Illuminate\Support\Facades\DB;
8  use Illuminate\Support\Facades\Hash;
9
10 2 references | 0 implementations
11 class UserController extends Controller
12 {
13     //coba akses model UserModel
14     1 reference | 0 overrides
15     public function index(): Factory|View
16     {
17         // $user = UserModel::all();
18         //         // Ambil data dari tabel m_user
19         // $data = DB::table('m_user')->get();
20
21         // return view('user', ['data' => $data]);
22
23         // tambah data user dengan eloquent model
24         $data = [
25             'username' => 'customer-1',
26             'nama' => 'Customer 1',
27             'password' => Hash::make(value: '12345'),
28             'level_id' => 4,
29         ];
30         UserModel::insert(values: $data); //tambahkan data ke tabel m_user
31
32         // coba akses model user model
33         $user = UserModel::all();
34         return view('user', data: ['data' => $user]);
35     }
36 }

```

- i. 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	Staf/Kasir	3
4	customer-1	Customer 1	4

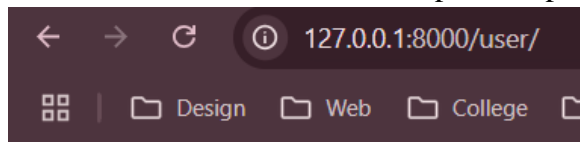
- j. Kita modifikasi lagi file UserController menjadi seperti berikut

```

app > Http > Controllers > UserController.php > PHP Intelephense > UserController
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use App\Models\UserModel;
7 use Illuminate\Support\Facades\DB;
8 use Illuminate\Support\Facades\Hash;
9
10 class UserController extends Controller
11 {
12     public function index(): Factory|View
13     {
14         // tambah data user dengan eloquent model
15         $data = [
16             'nama' => 'Pelanggan Pertama',
17         ];
18         // Akses model UserModel
19         $user = UserModel::all();
20         return view('user', data: ['data' => $user]);
21     }
22 }

```

- g. 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	Staf/Kasir	3
4	customer-1	Customer 1	4

## 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:**

Fungsi APP\_KEY pada setting .env di Laravel berfungsi sebagai kunci enkripsi utama yg digunakan oleh framework untuk mengamankan data sensitif, memastikan data aman tidak dapat dengan mudah diakses atau dimanipulasi org lain

2. Pada Praktikum 1, bagaimana kita men-generate nilai untuk APP\_KEY?

**Jawab:**

Dengan menjalankan perintah php artisan key:generate. Laravel akan secara otomatis menghasilkan kunci acak yg aman. Perintah tersebut hanya boleh dijalankan sekali, jika dijalankan berulang kali maka semua data sebelumnya yg dienkrip akan menjadi tidak valid dan tidak dapat dienkripsi Kembali

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 10 biasanya memiliki 3 file migrasi. File migrasi tsb antara lain:**

- a. 2014\_10\_12\_000000\_create\_users\_table.php : digunakan untuk membuat tabel users di database.
  - b. 2014\_10\_12\_100000\_create\_password\_reset\_tokens\_table.php : digunakan untuk membuat tabel password\_reset\_tokens
  - c. 2019\_08\_19\_000000\_create\_failed\_jobs\_table.php : digunakan untuk membuat tabel failed\_jobs
4. Secara default, file migrasi terdapat kode \$table->timestamps();, apa tujuan/output dari fungsi tersebut?

**Jawab:**

Fungsi ini digunakan untuk menambahkan dua kolom khusus ke dalam tabel database yang dibuat melalui migrasi:

- a. `created_at`: Menyimpan tanggal dan waktu ketika sebuah baris data (record) pertama kali dibuat.
  - b. `updated_at`: Menyimpan tanggal dan waktu ketika baris data tersebut terakhir diperbarui
5. Pada File Migrasi, terdapat fungsi `$table->id()`; Tipe data apa yang dihasilkan dari fungsi tersebut?

**Jawab:**

Fungsi `$table->id()`; menghasilkan kolom dengan tipe data `BIGINT UNSIGNED` yang bersifat auto-increment dan menjadi primary key tabel.

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

**Jawab:**

Perbedaan utama antara `$table->id()`; dan `$table->id('level_id')`; adalah nama kolom yang dihasilkan (`id` vs `level_id`).

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

**Jawab:**

Fungsi `->unique()` pada migrasi Laravel digunakan untuk menambahkan constraint unik pada kolom tertentu, memastikan bahwa tidak ada nilai duplikat di kolom tersebut dalam tabel.

8. Pada Praktikum 2.2 - Tahap 2, kenapa kolom `level_id` pada tabel `m_user` menggunakan `$table->unsignedBigInteger('level_id')`, sedangkan kolom `level_id` pada tabel `m_level` menggunakan `$table->id('level_id')` ?

**Jawab:**

Di tabel `m_level`, `$table->id('level_id')`; digunakan karena `level_id` adalah primary key, yang harus unik, auto-increment, dan bertipe `BIGINT UNSIGNED`, sedangkan Di tabel `m_user`, `$table->unsignedBigInteger('level_id')`; digunakan karena `level_id` adalah foreign key, yang hanya menyimpan nilai yang merujuk ke `level_id` di tabel `m_level`. Kolom ini tidak perlu auto-increment, tetapi harus memiliki tipe data yang sama (`BIGINT UNSIGNED`) untuk mendukung relasi.

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

**Jawab:**

Class tersebut berfungsi untuk melakukan hashing pada string '1234' yang merupakan password pengguna sehingga data tersebut tersimpan dengan aman di database.

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

**Jawab:**

Tanda tanya tersebut berfungsi sebagai placeholder untuk data yang akan di binding terlebih dahulu sebelum diproses oleh database, tujuannya agar terhindar dari sql injection.

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

**Jawab:**

Tujuannya untuk menentukan tabel dan primary key yang akan digunakan saat menjalankan sintaks yang membutuhkan class UserModel.

12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (DB Façade / Query Builder / Eloquent ORM) ? jelaskan

**Jawab:**

Menurut saya lebih mudah jika operasi CRUD menggunakan query builder karena sintaksnya yang lebih ringkas dan penggunaannya yang berbasis objek.