



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 1 (satu)

JOBSHEET 03

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

Sebelumnya kita sudah membahas mengenai *Routing*, *Controller*, dan *View* yang ada di Laravel. Sebelum kita masuk pada pembuatan aplikasi berbasis website, alangkah baiknya kita perlu menyiapkan Basis data sebagai tempat menyimpan data-data pada aplikasi kita nanti. Selain itu, umumnya kita perlu menyiapkan juga data awal yang kita gunakan sebelum membuat aplikasi, seperti data user administrator, data pengaturan sistem, dll.

Untuk itu, kita memerlukan teknik untuk merancang/membuat table basis data sebelum membuat aplikasi. Laravel memiliki fitur dalam pengelolaan basis data seperti, migration, seeder, model, dll.

Sebelum kita masuk materi, kita buat dulu project baru yang akan kita gunakan untuk membangun aplikasi sederhana dengan topik *Point of Sales (PoS)*, sesuai dengan **Studi Kasus PWL.pdf**.

Jadi kita bikin project Laravel 10 dengan nama **PWL_POS**.

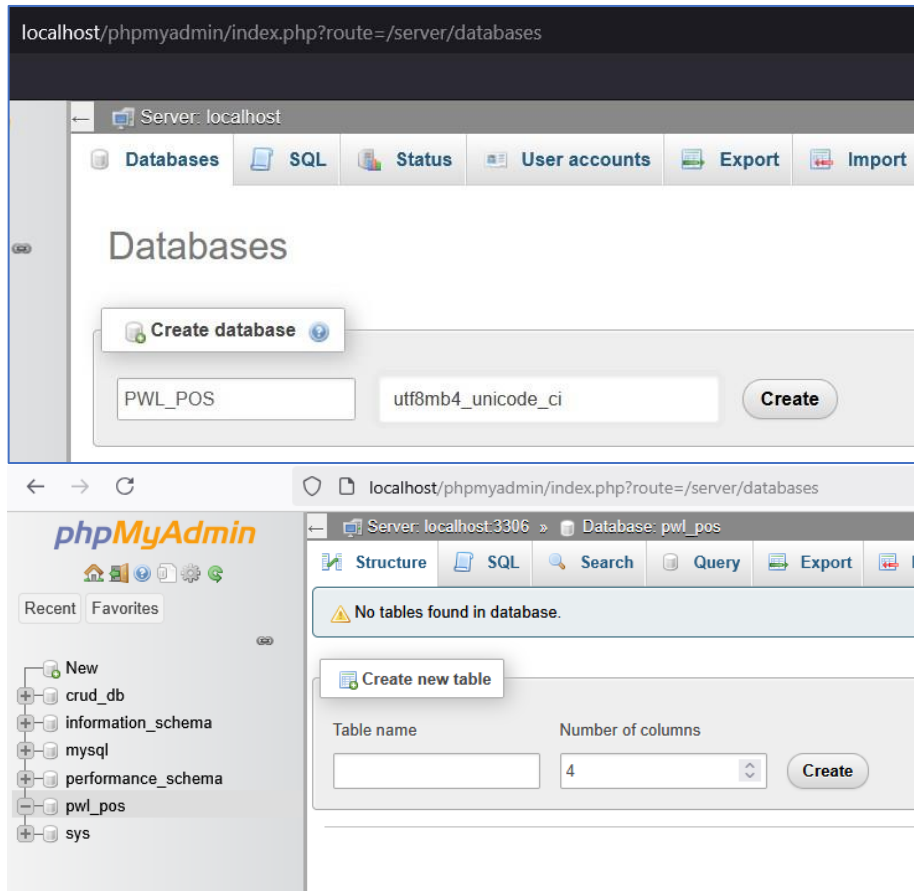
Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. PENGATURAN DATABASE

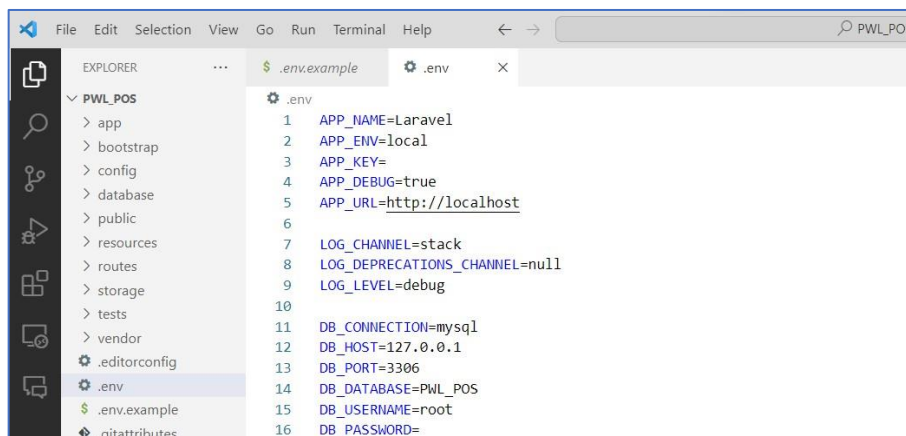
Database atau basis data menjadi komponen penting dalam membangun sistem. Hal ini dikarenakan database menjadi tempat untuk menyimpan data-data transaksi yang ada pada sistem. Koneksi ke database perlu kita atur agar sesuai dengan database yang kita gunakan.

Praktikum 1 - pengaturan 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**.



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



The screenshot shows a VS Code editor window with a file explorer on the left showing the project structure. The main editor displays the `.env` file with the following content:

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:KgPEif3b6D0mqm2FXJEy3lHh13EFasEJDuxXn9Af22Y=
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=PWL_POS
15 DB_USERNAME=root
16 DB_PASSWORD=
```

Below the editor, a terminal window shows the output of the `cat .env` command, displaying the same configuration values as the `.env` file.

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



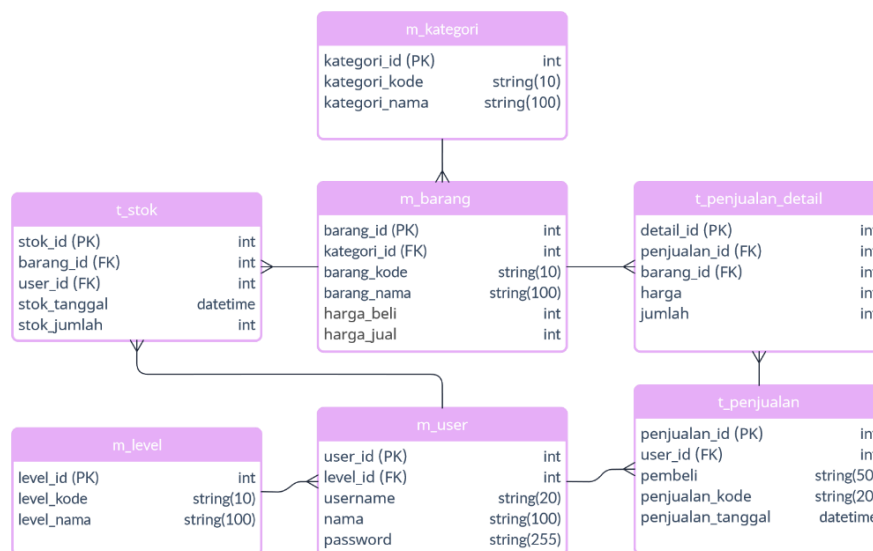
B. MIGRATION

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.

Sesuai dengan topik pembelajaran kita untuk membangun sistem *Point of Sales (PoS)* sederhana, maka kita perlu membuat migration sesuai desain database yang sudah didefinisikan pada file

Studi Kasus PWL.pdf



Dalam membuat file migration di Laravel, yang perlu kita perhatikan adalah struktur table yang ingin kita buat.

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.

Dari tips di atas, kita dapat melakukan cek untuk desain database yang sudah ada dengan mengetahui jumlah *foreign key* yang ada. Dan kita bisa menentukan table mana yang akan kita buat migrasinya terlebih dahulu.



No Urut	Nama Tabel	Jumlah FK
1	m_level	0
2	m_kategori	0
3	m_user	1
4	m_barang	1
5	t_penjualan	1
6	t_stok	2
7	t_penjualan_detail	2

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

- Menggunakan [artisan](#) untuk membuat *file migration*

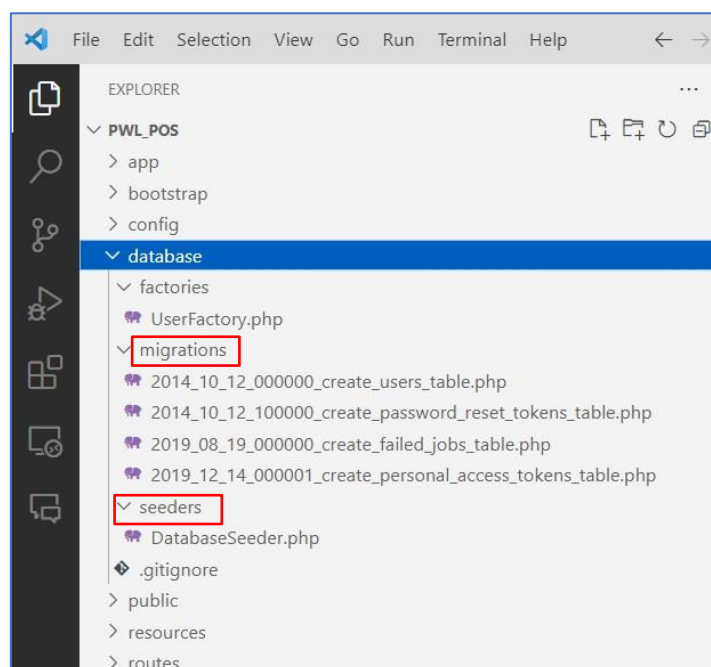
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

- 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.

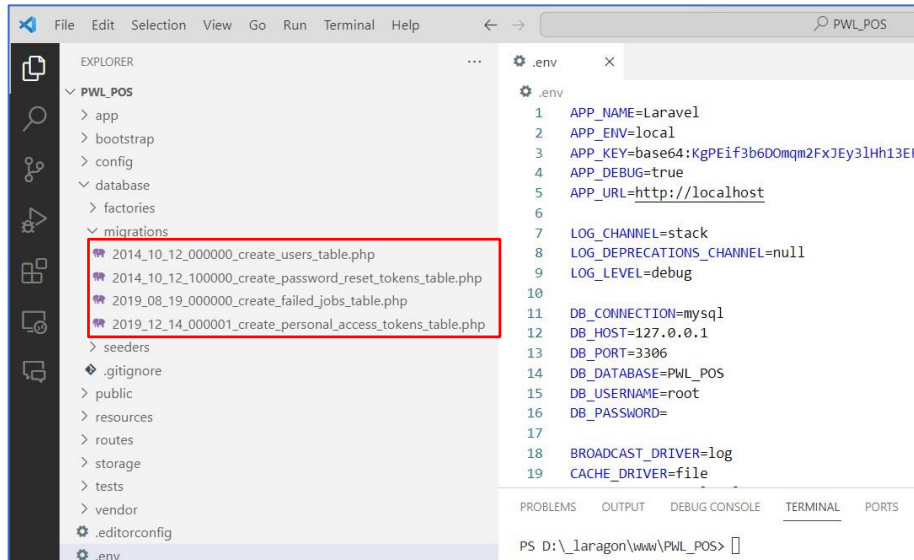
Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder [PWL_POS/database](#)





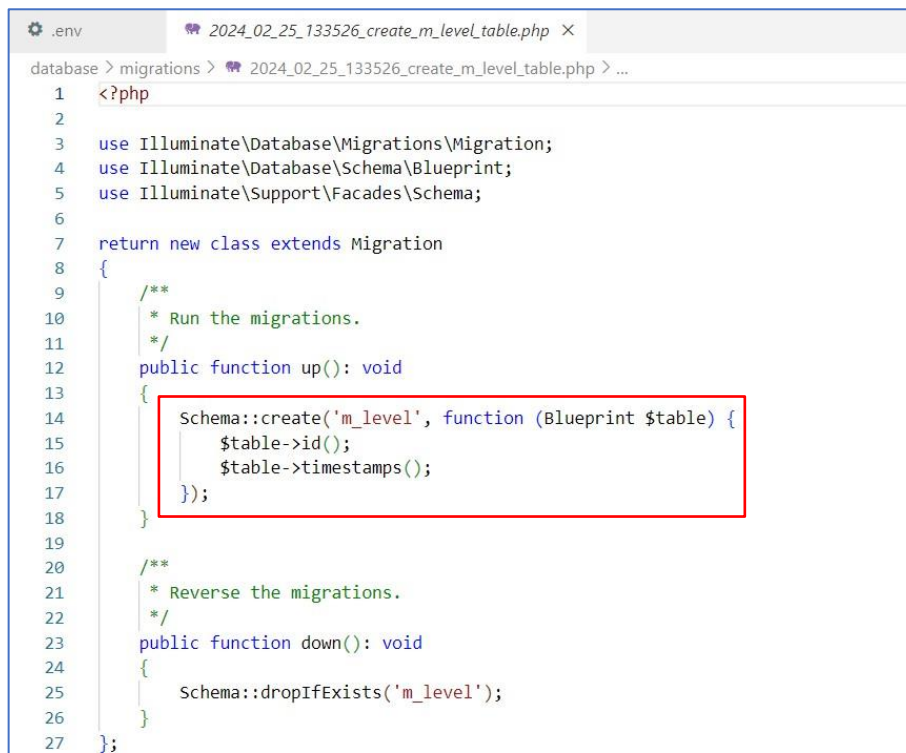
Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

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



2. Kita abaikan dulu yang di kotak merah (jangan di hapus)
3. Kita buat file migrasi untuk table `m_level` dengan perintah

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





```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'm_level', callback: function (Blueprint $table): void {
            $table->id();
            $table->timestamps();
        });
    }

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




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

```
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('m_level', function (Blueprint $table) {
15             $table->id('level_id');
16             $table->string('level_kode', 10)->unique();
17             $table->string('level_nama', 100);
18             $table->timestamps();
19         });
20     }
21
22     /**
23      * Reverse the migrations.
24      */
25     public function down(): void
26     {
27         Schema::dropIfExists('m_level');
28     }
29 };
```

```
return new class extends Migration {
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'm_level', callback: function (Blueprint $table): void {
            $table->id(column: 'level_id');
            $table->string(column: 'level_kode', length: 10)->unique();
            $table->string(column: 'level_nama', length: 100);
            $table->timestamps();
        });
    }

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




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>

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

```
php artisan migrate
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\laragon\www\PWL_POS> php artisan migrate

[INFO] Preparing database.

Creating migration table ..... 12ms DONE

[INFO] Running migrations.

2014_10_12_000000_create_users_table ..... 16ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 6ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 42ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 15ms DONE
2024_02_25_133526_create_m_level_table ..... 13ms DONE

PS D:\laragon\www\PWL_POS>
```

```
PS D:\APLIKASI\polinema\Pweb\laragon\apk\laragon\www\POS> php artisan migrate

[INFO] Preparing database.

Creating migration table ..... 22ms DONE

[INFO] Running migrations.

2014_10_12_000000_create_users_table ..... 33ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 9ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 28ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 37ms DONE
2025_03_02_123416_create_m_level_table ..... 34ms DONE
```



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

Table	Action	Rows
<input type="checkbox"/> failed_jobs	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> migrations	★ Browse Structure Search Insert Empty Drop	5
<input type="checkbox"/> m_level	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> password_reset_tokens	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> personal_access_tokens	★ Browse Structure Search Insert Empty Drop	0
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0

7. Ok, table sudah dibuat di database
8. Buat table *database* dengan *migration* untuk table **m_kategori** yang sama-sama tidak memiliki *foreign key*
9. Laporkan hasil Praktikum-2.1 ini dan *commit* perubahan pada *git*.

Praktikum 2.2 - Pembuatan file migrasi dengan relasi

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

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

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



```
7 return new class extends Migration
8 {
9     /**
10      * Run the migrations.
11      */
12     public function up(): void
13     {
14         Schema::create('m_user', function (Blueprint $table) {
15             $table->id('user_id');
16             $table->unsignedBigInteger('level_id')->index(); // indexing untuk ForeignKey
17             $table->string('username', 20)->unique(); // unique untuk memastikan tidak ada username yang sama
18             $table->string('nama', 100);
19             $table->string('password');
20             $table->timestamps();
21
22             // Mendefinisikan Foreign Key pada kolom level_id mengacu pada kolom level_id di tabel m_level
23             $table->foreign('level_id')->references('level_id')->on('m_level');
24         });
25     }
26
27     /**
28      * Reverse the migrations.
29      */
30     public function down(): void
31     {
32         Schema::dropIfExists('m_user');
33     }
34 };
```

```
<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration {
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'm_user', callback: function (Blueprint $table): void {
            $table->id(column: 'user_id');
            $table->unsignedBigInteger(column: 'level_id')->index(); // indexing untuk ForeignKey
            $table->string(column: 'username', length: 20)->unique(); // unique untuk memastikan tidak ada username yang sama
            $table->string(column: 'nama', length: 100);
            $table->string(column: 'password');
            $table->timestamps();

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

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

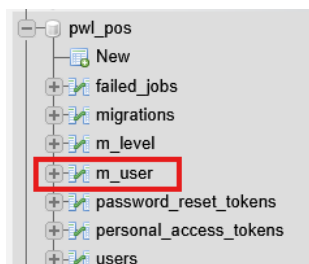


3. Simpan kode program Langkah 2, dan jalankan perintah **php artisan migrate**. Amati apa yang terjadi pada database.

```
PS D:\APLIKASI\polinema\Pweb\laragon\apk\laragon\www\POS> php artisan migrate

INFO Running migrations.

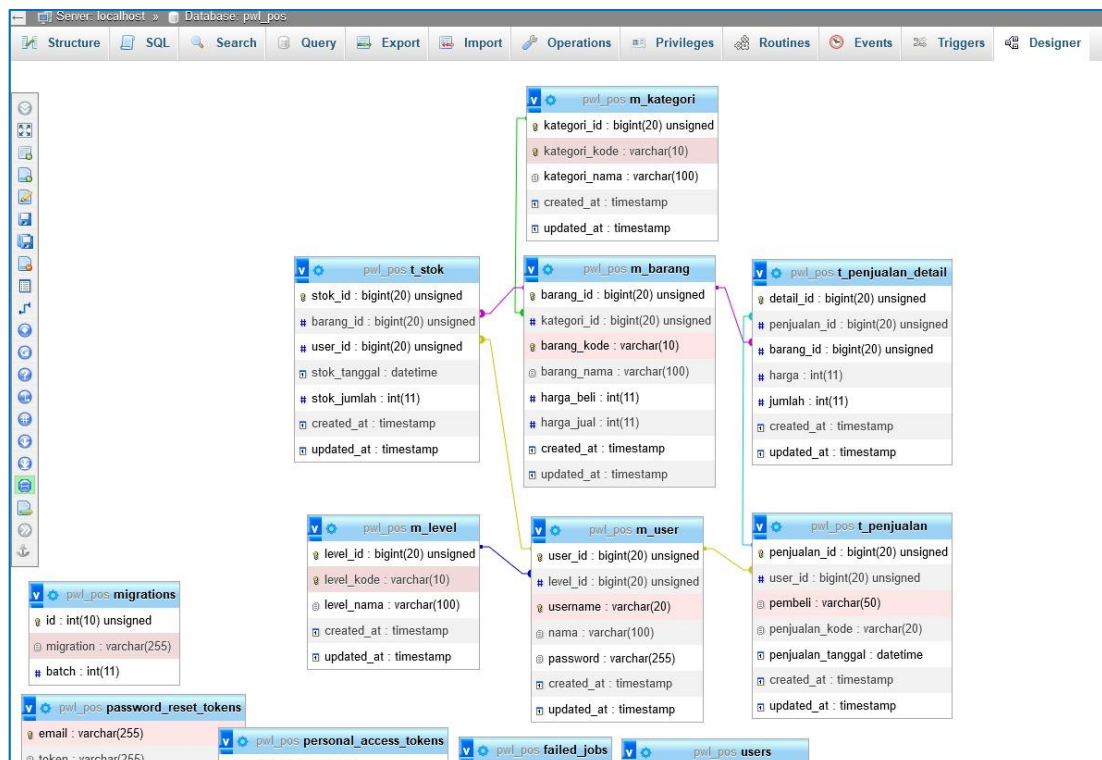
2025_03_02_123959_create_m_user_table ..... 249ms DONE
```

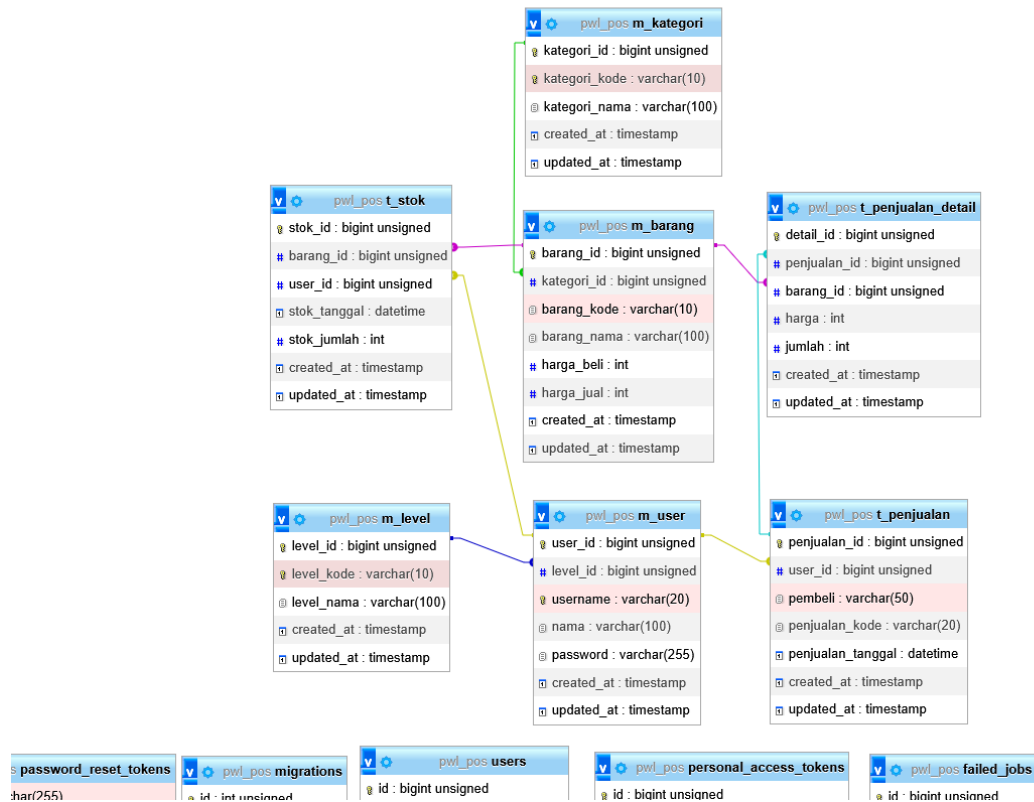


4. Buat table *database* dengan *migration* untuk table-table yang memiliki *foreign key*

m_barang
t_penjualan
t_stok
t_penjualan_detail

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





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



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

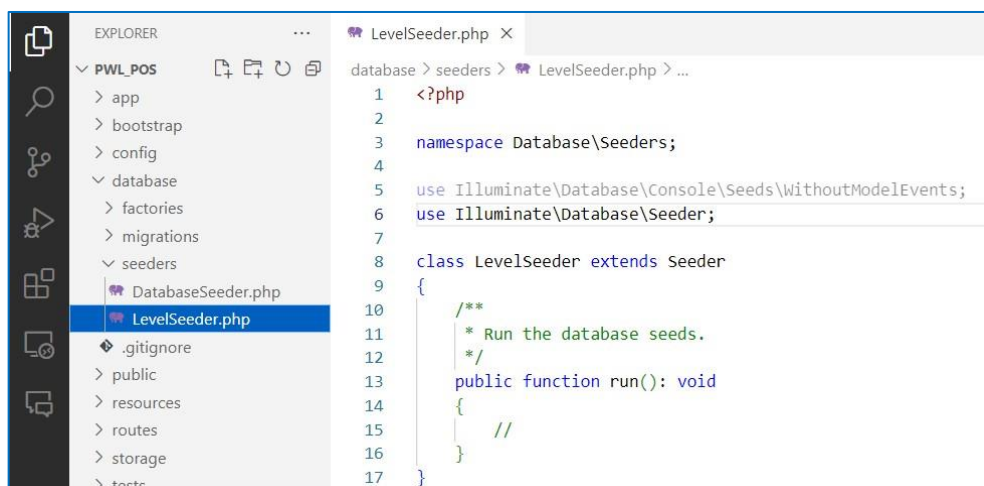
```
php artisan db:seed --class=<nama-class-seeder>
```

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

Praktikum 3 – Membuat file *seeder*

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

```
php artisan make:seeder LevelSeeder
```





```
<?php

namespace Database\Seeders;

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

0 references | 0 implementations
class LevelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    0 references | 0 overrides
    public function run(): void
    {
        //
    }
}
```

2. 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  class LevelSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $data = [
17             ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
18             ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
19             ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
20         ];
21         DB::table('m_level')->insert($data);
22     }
23 }
```




```
<?php

namespace Database\Seeders;

use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

0 references | 0 implementations
class LevelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    0 references | 0 overrides
    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(table: 'm_level')->insert(values: $data);
    }
}
```

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

```
php artisan db:seed --class=LevelSeeder
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\_laragon\www\PWL_POS> php artisan db:seed --class=LevelSeeder
```

```
INFO Seeding database.
```

```
PS D:\_laragon\www\PWL_POS>
```

```
PS D:\APLIKASI\polinema\Pweb\laragon\apk\laragon\www\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
<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
Check all With selected: Edit Copy Delete Export					



<div>←T→</div>				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

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

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



6. Modifikasi file `class UserSeeder` seperti berikut

```
9 class UserSeeder extends Seeder
10 {
11     public function run(): void
12     {
13         $data = [
14             [
15                 'user_id' => 1,
16                 'level_id' => 1,
17                 'username' => 'admin',
18                 'nama' => 'Administrator',
19                 'password' => Hash::make('12345'), // class untuk mengenkripsi/hash password
20             ],
21             [
22                 'user_id' => 2,
23                 'level_id' => 2,
24                 'username' => 'manager',
25                 'nama' => 'Manager',
26                 'password' => Hash::make('12345'),
27             ],
28             [
29                 'user_id' => 3,
30                 'level_id' => 3,
31                 'username' => 'staff',
32                 'nama' => 'Staff/Kasir',
33                 'password' => Hash::make('12345'),
34             ],
35         ];
36         DB::table('m_user')->insert($data);
37     }
38 }
```

```
0 references | 0 implementations
class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    0 references | 0 overrides
    public function run(): void
    {
        $data = [
            [
                'user_id' => 1,
                'level_id' => 1,
                'username' => 'admin',
                'nama' => 'Administrator',
                'password' => Hash::make(value: '12345'), // Hash password
            ],
            [
                'user_id' => 2,
                'level_id' => 2,
                'username' => 'manager',
                'nama' => 'Manager',
                'password' => Hash::make(value: '12345'),
            ],
            [
                'user_id' => 3,
                'level_id' => 3,
                'username' => 'staff',
                'nama' => 'Staff/Kasir',
                'password' => Hash::make(value: '12345'),
            ],
        ];
        DB::table(table: 'm_user')->insert(values: $data);
    }
}
```

7. Jalankan perintah untuk mengeksekusi class `UserSeeder`

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



8. 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\$12\$Tevu4dDO1CUAQpeM6H.Vp.LySwhY.4oAKU7FzwS6fXV...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$Ajfns20/FdPTeUgghz31muEhlFarulxkh5wvZ9NGRpu...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMY/BHnbJ9W...

	user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	admin	Administrator	\$2y\$12\$F68WSVNW5k9aX5q.VfOfieFUVJszqPVP21ac8Ur9TPA...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$Ak4UVNuvOrNg/4CQZ4TC2OK6r.kMMEnI7vj4cyM/8Js...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$vitZb2wkLKU5HHOYMW5ncO4JgLT7GmIANGOBBrnUvJV...	NULL	NULL

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

KATEGORI

```
class KategoriSeeder extends Seeder
{
    0 references|0 overrides
    public function run(): void
    {
        $kategori = [
            ['kategori_kode' => 'ELEC', 'kategori_nama' => 'Elektronik'],
            ['kategori_kode' => 'FASH', 'kategori_nama' => 'Fashion'],
            ['kategori_kode' => 'FOOD', 'kategori_nama' => 'Makanan'],
            ['kategori_kode' => 'HEALTH', 'kategori_nama' => 'Kesehatan'],
            ['kategori_kode' => 'SPORT', 'kategori_nama' => 'Olahraga']
        ];

        DB::table(table: 'm_kategori')->insert(values: $kategori);
    }
}
```

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	ELEC	Elektronik	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	FASH	Fashion	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	FOOD	Makanan	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	HEALTH	Kesehatan	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	SPORT	Olahraga	NULL	NULL

BARANG

```
class BarangSeeder extends Seeder
{
    0 references|0 overrides
    public function run(): void
    {
        $barang = [];
        for ($i = 1; $i <= 10; $i++) {
            $barang[] = [
                'kategori_id' => rand(min: 1, max: 5),
                'barang_kode' => "BRG00" . $i,
                'barang_nama' => "Barang " . $i,
                'harga_beli' => rand(min: 1000, max: 10000),
                'harga_jual' => rand(min: 1100, max: 12000),
                'created_at' => now(),
                'updated_at' => now(),
            ];
        }

        DB::table(table: 'm_barang')->insert(values: $barang);
    }
}
```

	barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	5	BRG001	Barang 1	6113	5744	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	5	BRG002	Barang 2	1713	7821	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	1	BRG003	Barang 3	3037	1978	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	1	BRG004	Barang 4	5055	10402	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	1	BRG005	Barang 5	4151	11745	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	4	BRG006	Barang 6	7285	11182	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	4	BRG007	Barang 7	1432	9508	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	3	BRG008	Barang 8	4567	3231	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	9	1	BRG009	Barang 9	2516	3063	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	10	3	BRG0010	Barang 10	8932	6126	2025-03-02 13:22:43	2025-03-02 13:22:43



STOK

```
class StokSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $stok = [];
        for ($i = 1; $i <= 10; $i++) {
            $stok[] = [
                'barang_id' => $i,
                'user_id' => 1, // Gantilah dengan user_id yang ada
                'stok_tanggal' => now(),
                'stok_jumlah' => rand(min: 10, max: 100),
                'created_at' => now(),
                'updated_at' => now(),
            ];
        }

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

		stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	1	2025-03-02 13:22:43	89	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	2	2	1	2025-03-02 13:22:43	74	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	3	3	1	2025-03-02 13:22:43	95	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	4	4	1	2025-03-02 13:22:43	55	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	5	5	1	2025-03-02 13:22:43	36	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	6	6	1	2025-03-02 13:22:43	33	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	7	7	1	2025-03-02 13:22:43	75	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	8	8	1	2025-03-02 13:22:43	36	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	9	9	1	2025-03-02 13:22:43	87	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	10	10	1	2025-03-02 13:22:43	50	2025-03-02 13:22:43	2025-03-02 13:22:43

PENJUALAN

```
class PenjualanSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $penjualan = [];
        for ($i = 1; $i <= 10; $i++) {
            $penjualan[] = [
                'user_id' => 1, // Sesuaikan dengan user yang ada
                'pembeli' => 'Pelanggan ' . $i,
                'penjualan_kode' => 'PJI00' . $i,
                'penjualan_tanggal' => now(),
                'created_at' => now(),
                'updated_at' => now(),
            ];
        }

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

		penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	Pelanggan 1	PJI001	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	2	1	Pelanggan 2	PJI002	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	3	1	Pelanggan 3	PJI003	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	4	1	Pelanggan 4	PJI004	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	5	1	Pelanggan 5	PJI005	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	6	1	Pelanggan 6	PJI006	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	7	1	Pelanggan 7	PJI007	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	8	1	Pelanggan 8	PJI008	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	9	1	Pelanggan 9	PJI009	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>	Edit Copy Delete	10	1	Pelanggan 10	PJI0010	2025-03-02 13:22:43	2025-03-02 13:22:43	2025-03-02 13:22:43

PENJUALAN DETAIL

```
class PenjualanDetailSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $penjualan_detail = [];
        for ($i = 1; $i <= 10; $i++) {
            for ($j = 1; $j <= 3; $j++) { // 3 barang per transaksi
                $penjualan_detail[] = [
                    'penjualan_id' => $i,
                    'barang_id' => rand(min: 1, max: 10),
                    'harga' => rand(min: 1100, max: 12000),
                    'jumlah' => rand(min: 1, max: 5),
                    'created_at' => now(),
                    'updated_at' => now(),
                ];
            }
        }

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



			detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
<input type="checkbox"/>				1	1	2	1683	5 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				2	1	6	5267	5 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				3	1	3	8955	4 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				4	2	7	3651	3 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				5	2	5	7571	2 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				6	2	10	11843	2 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				7	3	1	3755	5 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				8	3	1	3016	5 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				9	3	1	10351	2 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				10	4	3	5810	4 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				11	4	4	11347	4 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				12	4	8	5425	3 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				13	5	10	1532	5 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				14	5	9	9611	4 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				15	5	2	2298	1 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				16	6	9	9783	2 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				17	6	9	1794	1 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				18	6	7	6090	5 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				19	7	5	7402	1 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				20	7	2	5501	5 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				21	7	9	7171	4 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				22	8	5	10032	1 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				23	8	1	11181	2 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				24	8	4	6644	3 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				25	9	9	5751	3 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				26	9	3	7846	3 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				27	9	6	2426	5 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				28	10	4	9112	1 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				29	10	2	4465	2 2025-03-02 13:22:43	2025-03-02 13:22:43
<input type="checkbox"/>				30	10	7	8452	1 2025-03-02 13:22:43	2025-03-02 13:22:43

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



D. DB FACADE

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 ...*”

Raw query adalah cara paling dasar dan tradisional yang ada di Laravel. Raw query terasa familiar karena biasa kita pakai ketika melakukan query langsung ke database.

INFO

Dokumentasi penggunaan DB Façade bisa dicek di laman ini
<https://laravel.com/docs/10.x/database#running-queries>

Terdapat banyak method yang bisa digunakan pada DB Façade ini. Akan tetapi yang kita pelajari cukup 4 (empat) method yang umum dipakai, yaitu

a. `DB::select()`

Method ini digunakan untuk mengambil data dari database. Method ini **mengembalikan** (*return*) data hasil *query*. Contoh

```
DB::select('select * from m_user'); //Query semua data pada tabel m_user
```

```
DB::select('select * from m_user where level_id = ?', [1]); //Query tabel m_user dengan level_id = 1
```

```
DB::select('select * from m_user where level_id = ? and username = ?', [1, 'admin']);
```

b. `DB::insert()`

Method ini digunakan untuk memasukkan data pada table database. Method ini **tidak memiliki nilai pengembalian** (*no return*). Contoh

```
DB::insert('insert into m_level(level_kode, level_nama) values(?,?)', ['CUS', 'Pelanggan']);
```

c. `DB::update()`

Method ini digunakan saat menjalankan *raw query* untuk meng-update data pada database. Method ini **memiliki nilai pengembalian** (*return*) berupa jumlah baris data yang ter-update. Contoh

```
DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
```




d. `DB::delete()`

Method ini digunakan saat menjalankan *raw query* untuk menghapus data dari table. Method ini **memiliki nilai pengembalian (*return*)** berupa jumlah baris data yang telah dihapus. Contoh

```
DB::delete('delete from m_level where level_kode = ?', ['CUS']);
```

Ok, sekarang mari kita coba praktikkan menggunakan DB Façade pada project kita

Praktikum 4 – Implementasi DB Facade

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

```
php artisan make:controller LevelController
```

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

```
LevelController.php  web.php X
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\LevelController;
4  use Illuminate\Support\Facades\Route;
5
6
7  Route::get('/', function () {
8      return view('welcome');
9  });
10
11 Route::get('/level', [LevelController::class, 'index']);
```

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

Route::get(uri: '/', action: function () { Factory|View {
    return view(view: 'welcome');
});

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

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

```
LevelController.php X  web.php
app > Http > Controllers > LevelController.php > ...
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()
11     {
12         DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13
14         return 'Insert data baru berhasil';
15     }
16 }
```



```
<?php
namespace App\Http\Controllers;

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

2 references | 0 implementations
class LevelController extends Controller
{
    1 reference | 0 overrides
    public function index(): string
    {
        DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', bindings: ['CUS', 'pelanggan', now()]);

        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`

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	CUS	Pelanggan	2024-02-26 08:20:00	NULL

← → ↻ 127.0.0.1:8000/level/store

Data level baru berhasil ditambahkan!

migrations

m_barang

m_kategori

m_level

m_user

password_reset_tokens

personal_access_tokens

←T→

	level_id	level_kode	level_nama	created_at	updated_at
<div><div><div></div></div><div><div>Edit</div><div>Copy</div><div>Delete</div></div></div>	1	ADM	Administrator	NULL	NULL
<div><div><div></div></div><div><div>Edit</div><div>Copy</div><div>Delete</div></div></div>	2	MNG	Manager	NULL	NULL
<div><div><div></div></div><div><div>Edit</div><div>Copy</div><div>Delete</div></div></div>	3	STF	Staff/Kasir	NULL	NULL
<div><div><div></div></div><div><div>Edit</div><div>Copy</div><div>Delete</div></div></div>	4	CUS	Pelanggan	2025-03-02 14:31:55	NULL

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

```
LevelController.php x web.php
app > Http > Controllers > LevelController.php > ...
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()
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_kode = ?', ['Customer', 'CUS']);
16         return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
17     }
18 }
```



```
<?php
namespace App\Http\Controllers;

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

2 references|0 implementations
class LevelController extends Controller
{
    0 references|0 overrides
    public function index(): string
    {
        // Insert data (jika ingin menggunakannya kembali, hapus tanda komentar)
        // DB::insert('INSERT INTO m_level (level_kode, level_nama, created_at) VALUES (?, ?, ?)',
        // ['CUS', 'Pelanggan', now()]);

        // Update data
        $row = DB::update(query: 'UPDATE m_level SET level_nama = ? WHERE level_kode = ?', bindings: ['Customer', 'CUS']);

        return "Update data berhasil. Jumlah data yang diupdate: " . $row . ' baris';
    }
}
```

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

← → ↻ 127.0.0.1:8000/level

Update data berhasil. Jumlah data yang diupdate: 1 baris

migrations	level_id	level_kode	level_nama	created_at	updated_at
m_barang	1	ADM	Administrator	NULL	NULL
m_kategori	2	MNG	Manager	NULL	NULL
m_level	3	STF	Staff/Kasir	NULL	NULL
m_user	4	CUS	Customer	2025-03-02 14:31:55	NULL
password_reset_tokens					
personal_access_tokens					

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

```
LevelController.php × web.php
app > Http > Controllers > LevelController.php > LevelController > index
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()
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_kode = ?', ['Customer', 'CUS']);
16         // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
17
18         $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
19         return "Delete data berhasil. Jumlah data yang dihapus: " . $row . " baris";
20
21     }
}
```

```
// DELETE DATA
$row = DB::delete(query: 'DELETE FROM m_level WHERE level_kode = ?', bindings: ['CUS']);
return "Delete data berhasil. Jumlah data yang dihapus: " . $row . " baris";
```

level_id	level_kode	level_nama	created_at	updated_at
1	ADM	Administrator	NULL	NULL
2	MNG	Manager	NULL	NULL
3	STF	Staff/Kasir	NULL	NULL



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

```
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()
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_kode = ?', ['Customer', 'CUS']);
16         // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
17
18         // $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
19         // return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
20
21         $data = DB::select('select * from m_level');
22         return view('level', ['data' => $data]);
23     }
24 }
```

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

9. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil **view('level')**, maka kita buat file view pada VSCode di **PWL_POS/resources/view/level.blade.php**

```
LevelController.php level.blade.php X web.php
resources > views > level.blade.php > ...
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Data Level Pengguna</title>
5     </head>
6     <body>
7         <h1>Data Level Pengguna</h1>
8         <table border="1" cellpadding="2" cellspacing="0">
9             <tr>
10                 <th>ID</th>
11                 <th>Kode Level</th>
12                 <th>Nama Level</th>
13             </tr>
14             @foreach ($data as $d)
15                 <tr>
16                     <td>{{ $d->level_id }}</td>
17                     <td>{{ $d->level_kode }}</td>
18                     <td>{{ $d->level_nama }}</td>
19                 </tr>
20             @endforeach
21         </table>
22     </body>
23 </html>
```

Data Level Pengguna

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



KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
Jl. Soekarno Hatta No. 9, Jatimulyo, Lowokwaru, Malang 65141
Telp. (0341) 404424 – 404425, Fax (0341) 404420
<http://www.polinema.ac.id>

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



E. QUERY BUILDER

Query builder adalah fitur yang disediakan Laravel untuk melakukan proses CRUD (*create, retrieve/read, update, delete*) pada database. Berbeda dengan *raw query* pada DB Facade yang mengharuskan kita menulis perintah SQL, pada *query builder* perintah SQL ini diakses menggunakan method. Jadi, kita tidak menulis perintah SQL secara langsung, melainkan cukup memanggil method-method yang ada di *query builder*.

Query builder membuat kode kita menjadi rapi dan lebih mudah dibaca. Selain itu *query builder* tidak terikat ke satu jenis database, jadi query builder bisa digunakan untuk mengakses berbagai jenis database seperti MySQL, MariaDB, PostgreSQL, SQL Server, dll. Jika suatu saat ingin beralih dari database MySQL ke PostgreSQL, tidak akan banyak kendala. Namun kelemahan dari *query builder* adalah kita harus mengetahui method-method apa saja yang ada di *query builder*.

INFO

Dokumentasi penggunaan Query Builder pada Laravel bisa dicek di laman ini

<https://laravel.com/docs/10.x/queries>

Ciri khas *query builder* Laravel adalah kita tentukan dahulu target table yang akan kita akses untuk operasi CRUD.

```
DB::table('<nama-tabel>'); // query builder untuk melakukan operasi CRUD pada tabel yang dituju
```

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

- a. Perintah untuk *insert* data dengan method `insert()`

```
DB::table('m_kategori')->insert(['kategori_kode' => 'SMP', 'kategori_nama' => 'Smartphone']);
```

Query yang dihasilkan dari kode di atas adalah

```
insert into m_kategori(kategori_kode, kategori_nama) values('SMP', 'Smartphone');
```

- b. Perintah untuk *update* data dengan method `where()` dan `update()`

```
DB::table('m_kategori')->where('kategori_id', 1)->update(['kategori_nama' => 'Makanan Ringan']);
```

Query yang dihasilkan dari kode di atas adalah

```
update m_kategori set kategori_nama = 'Makanan Ringan' where kategori_id = 1;
```




- c. Perintah untuk *delete* data dengan method `where()` dan `delete()`

```
DB::table('m_kategori')->where('kategori_id', 9) ->delete();
```

Query yang dihasilkan dari kode di atas adalah

```
delete from m_kategori where kategori_id = 9;
```

- d. Perintah untuk ambil data

Method Query Builder	Query yang dihasilkan
<code>DB::table('m_kategori')->get();</code>	<code>select * from m_kategori</code>
<code>DB::table('m_kategori')->where('kategori_id', 1)->get();</code>	<code>select * from m_kategori where kategori_id = 1;</code>
<code>DB::table('m_kategori')->select('kategori_kode')->where('kategori_id', 1)->get();</code>	<code>select kategori_kode from m_kategori where kategori_id = 1;</code>

Praktikum 5 – Implementasi *Query Builder*

1. Kita buat controller dahulu untuk mengelola data pada table `m_kategori`

```
php artisan make:controller KategoriController
```

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

```
LevelController.php  KategoriController.php  level.blade.php  web.php X
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use Illuminate\Support\Facades\Route;
6
7
8  Route::get('/', function () {
9      return view('welcome');
10 });
11
12 Route::get('/level', [LevelController::class, 'index']);
13 Route::get('/kategori', [KategoriController::class, 'index']);
```

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

Route::get(uri: '/', action: function () { Factory|View {
    return view(view: 'welcome');
});

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

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



```
LevelController.php KategoriController.php X level.blade.php web.php
app > Http > Controllers > KategoriController.php > 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     public function index()
11     {
12         $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($data);
18         return 'Insert data baru berhasil';
19     }
20 }
```

```
namespace App\Http\Controllers;

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

2 references | 0 implementations
class KategoriController extends Controller
{
    1 reference | 0 overrides
    public function index(): string
    {
        $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`

← → ↺ 127.0.0.1:8000/kategori

Insert data baru berhasil

← → ↺

				kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	ELEC	Elektronik	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	FASH	Fashion	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	FOOD	Makanan	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	HEALTH	Kesehatan	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	SPORT	Olahraga	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	6	SNK	Snack/Makanan Ringan	2025-03-02 14:56:16	NULL



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

```
app > Http > Controllers > KategoriController.php > 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     public function index()
11     {
12         /* $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($data);
18         return 'Insert data baru berhasil'; */
19
20         $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
21         return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
22     }
23 }
```

```
<?php
namespace App\Http\Controllers;

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

2 references | 0 implementations
class KategoriController extends Controller
{
    1 reference | 0 overrides
    public function index(): string
    {
        // Insert data baru ke dalam tabel m_kategori
        /*
        $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now(),
        ];

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

        // Update kategori_nama berdasarkan kategori_kode
        $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
        return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
    }
}
```

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

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	ELEC	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	FASH	Fashion	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	FOOD	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	HEALTH	Kesehatan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	SPORT	Olahraga	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	SNK	Camilan	2025-03-02 14:56:16	NULL

Update data berhasil. Jumlah data yang diupdate: 1 baris

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



```
10 public function index()
11 {
12     /* $data = [
13         'kategori_kode' => 'SNK',
14         'kategori_nama' => 'Snack/Makanan Ringan',
15         'created_at' => now()
16     ];
17     DB::table('m_kategori')->insert($data);
18     return 'Insert data baru berhasil'; */
19
20     // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
21     // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
22
23     $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
24     return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
25 }
```

```
// Hapus data berdasarkan kategori_kode
$row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
```

← → ↻ 127.0.0.1:8000/kategori

Delete data berhasil. Jumlah data yang dihapus: 1 baris

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

```
10 public function index()
11 {
12     /* $data = [
13         'kategori_kode' => 'SNK',
14         'kategori_nama' => 'Snack/Makanan Ringan',
15         'created_at' => now()
16     ];
17     DB::table('m_kategori')->insert($data);
18     return 'Insert data baru berhasil'; */
19
20     // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
21     // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
22
23     // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
24     // return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
25
26     $data = DB::table('m_kategori')->get();
27     return view('kategori', ['data' => $data]);
28 }
```

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**



```
resources > views > kategori.blade.php > html > body > table > tr > td
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Data Kategori Barang</title>
5   </head>
6   <body>
7     <h1>Data Kategori Barang</h1>
8     <table border="1" cellpadding="2" cellspacing="0">
9       <tr>
10        <th>ID</th>
11        <th>Kode Kategori</th>
12        <th>Nama Kategori</th>
13      </tr>
14      @foreach ($data as $d)
15        <tr>
16          <td>{{ $d->kategori_id }}</td>
17          <td>{{ $d->kategori_kode }}</td>
18          <td>{{ $d->kategori_nama }}</td>
19        </tr>
20      @endforeach
21    </table>
22  </body>
23 </html>
```

Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	ELEC	Elektronik
2	FASH	Fashion
3	FOOD	Makanan
4	HEALTH	Kesehatan
5	SPORT	Olahraga

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

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



F. ELOQUENT ORM

Eloquent ORM adalah fitur bawaan dari laravel. Eloquent ORM adalah cara pengaksesan database dimana setiap baris tabel dianggap sebagai sebuah object. Kata ORM sendiri merupakan singkatan dari **Object-relational mapping**, yakni suatu teknik programming untuk mengkonversi data ke dalam bentuk object.

INFO

Eloquent ORM memerlukan Model untuk proses konversi data pada tabel menjadi object. Object inilah yang nantinya akan kita akses dari dalam controller. Oleh karena itu **membuat Model pada Laravel berarti menggunakan Eloquent ORM**. Silahkan cek disini

<https://laravel.com/docs/10.x/eloquent>

Perintah untuk membuat model adalah sebagai berikut

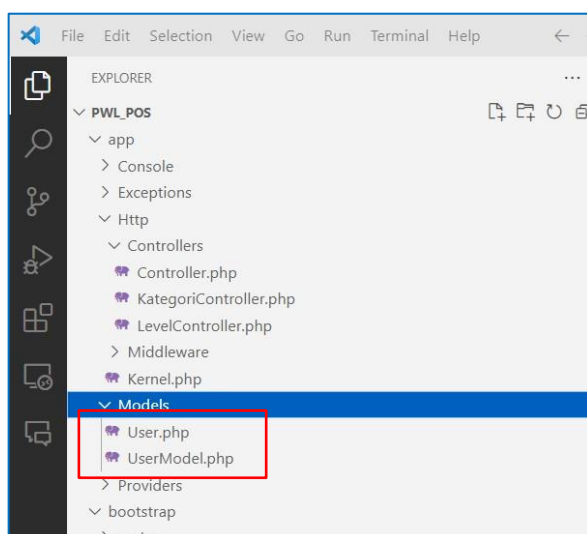
```
php artisan make:model <nama-model-CamelCase>
```

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

Praktikum 6 – Implementasi Eloquent ORM

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

```
php artisan make:model UserModel
```





- 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`
- Kita buka file `UserModel.php` dan modifikasi seperti berikut

```
app > Models > UserModel.php > UserModel
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class UserModel extends Model
9  {
10     use HasFactory;
11
12     protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan oleh model ini
13     protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
14 }
15
```

```
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

0 references|0 implementations
class UserModel extends Model
{
    use HasFactory;

    0 references
    protected $table = 'm_user'; // Mendefinisikan nama tabel yang digunakan
    0 references
    protected $primaryKey = 'user_id'; // Mendefinisikan primary key dari tabel yang digunakan
}
```

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

```
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use App\Http\Controllers\UserController;
6  use Illuminate\Support\Facades\Route;
7
8
9  Route::get('/', function () {
10     return view('welcome');
11 });
12
13 Route::get('/level', [LevelController::class, 'index']);
14 Route::get('/kategori', [KategoriController::class, 'index']);
15 Route::get('/user', [UserController::class, 'index']);
16
Route::get(uri: '/user', action: [UserController::class, 'index']);
```

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



```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7
8  class UserController extends Controller
9  {
10     public function index()
11     {
12         // coba akses model UserModel
13         $user = UserModel::all(); // ambil semua data dari tabel m_user
14         return view('user', ['data' => $user]);
15     }
16 }
```

```
2 references | 0 implementations
class UserController extends Controller
{
    1 reference | 0 overrides
    public function index(): Factory|View
    {
        // Ambil semua data dari tabel m_user
        $user = UserModel::all();

        // Kirim data ke view user.blade.php
        return view(view: 'user', data: ['data' => $user]);
    }
}
```

6. Kemudian kita buat view `user.blade.php`



```
resources > views > user.blade.php > ...
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Data User</title>
5   </head>
6   <body>
7     <h1>Data User</h1>
8     <table border="1" cellpadding="2" cellspacing="0">
9       <tr>
10        <th>ID</th>
11        <th>Username</th>
12        <th>Nama</th>
13        <th>ID Level Pengguna</th>
14      </tr>
15      @foreach ($data as $d)
16        <tr>
17          <td>{{ $d->user_id }}</td>
18          <td>{{ $d->username }}</td>
19          <td>{{ $d->nama }}</td>
20          <td>{{ $d->level_id }}</td>
21        </tr>
22      @endforeach
23    </table>
24  </body>
25 </html>
```

```
<!DOCTYPE html>
<html>

<head>
  <title>Data User</title>
</head>

<body>
  <h1>Data User</h1>

  @if (count(value: $data) > 0)
    <table border="1" cellpadding="2" cellspacing="0">
      <thead>
        <tr>
          <th>ID</th>
          <th>Username</th>
          <th>Nama</th>
          <th>ID Level Pengguna</th>
        </tr>
      </thead>
      <tbody>
        @foreach ($data as $d)
          <tr>
            <td>{{ $d->user_id }}</td>
            <td>{{ $d->username }}</td>
            <td>{{ $d->nama }}</td>
            <td>{{ $d->level_id }}</td>
          </tr>
        @endforeach
      </tbody>
    </table>
  @else
    <p>Tidak ada data pengguna.</p>
  @endif
</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`



```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7  use Illuminate\Support\Facades\Hash;
8
9  class UserController extends Controller
10 {
11     public function index()
12     {
13         // tambah data user dengan Eloquent Model
14         $data = [
15             'username' => 'customer-1',
16             'nama' => 'Pelanggan',
17             'password' => Hash::make('12345'),
18             'level_id' => 4
19         ];
20         UserModel::insert($data); // tambahkan data ke tabel m_user
21
22         // coba akses model UserModel
23         $user = UserModel::all(); // ambil semua data dari tabel m_user
24         return view('user', ['data' => $user]);
25     }
26 }
```

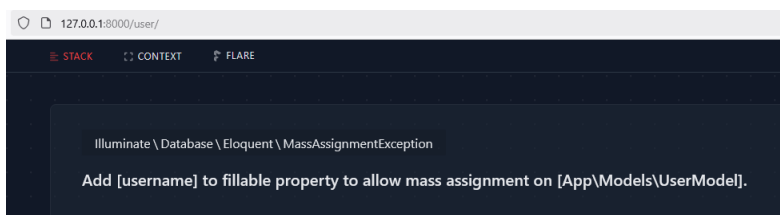
```
2 references|0 implementations
class UserController extends Controller
{
    1 reference|0 overrides
    public function index(): Factory|View
    {
        // Tambah data user dengan Eloquent Model
        $data = [
            'username' => 'customer-1',
            'nama' => 'Pelanggan',
            'password' => Hash::make(value: '12345'), // Menambahkan password dengan hash
            'level_id' => 4
        ];

        // Masukkan data ke dalam database
        UserModel::create(attributes: $data);

        // Ambil semua data dari tabel m_user
        $user = UserModel::all();

        // Kirim data ke tampilan
        return view(view: 'user', data: ['data' => $user]);
    }
}
```

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

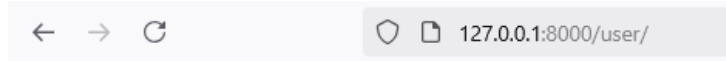


10. Kita modifikasi lagi file `UserController` menjadi seperti berikut

```
9  class UserController extends Controller
10 {
11     public function index()
12     {
13         // tambah data user dengan Eloquent Model
14         $data = [
15             'nama' => 'Pelanggan Pertama',
16         ];
17         UserModel::where('username', 'customer-1')->update($data); // update data user
18
19         // coba akses model UserModel
20         $user = UserModel::all(); // ambil semua data dari tabel m_user
21         return view('user', ['data' => $user]);
22     }
23 }
```



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

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

G. Penutup

Jawablah pertanyaan berikut sesuai pemahaman materi di atas

- Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari `APP_KEY` pada *file setting .env* Laravel?
 - `APP_KEY` digunakan sebagai kunci enkripsi dalam Laravel. Fungsinya adalah:
Mengamankan data yang dienkripsi, seperti sesi dan password.
- Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk `APP_KEY`?
 - `php artisan key:generate`
- Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?
 - `create_users_table.php` : Untuk tabel pengguna (users)
 - `create_password_resets_table.php` : Untuk menyimpan token reset password
 - `create_failed_jobs_table.php` : Untuk mencatat pekerjaan (jobs) yang gagal
- Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/output dari fungsi tersebut?
 - `created_at` : Menyimpan waktu saat data pertama kali dibuat
 - `updated_at` : Menyimpan waktu saat data terakhir diperbarui
- Pada File Migrasi, terdapat fungsi `$table->id();` Tipe data apa yang dihasilkan dari fungsi tersebut?
 - Nama : `id`
 - Tipe data : `BIGINT UNSIGNED`
 - Auto-increment (bertambah otomatis)
 - Primary key



6. Apa bedanya hasil migrasi pada table `m_level`, antara menggunakan `$table->id();` dengan menggunakan `$table->id('level_id');` ?
 - `Table id();` → Membuat kolom id sebagai BIGINT UNSIGNED dan Primary Key
 - `Table id('level_id');` → Membuat kolom level_id dengan nama khusus.
7. Pada migration, Fungsi `->unique()` digunakan untuk apa?
 - memastikan tidak ada duplikasi data dalam kolom tertentu.
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')` ?
 - Di `m_level`, `level_id` adalah primary key
 - Di `m_user`, `level_id` adalah foreign Key
9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');` ?
 - Hash digunakan untuk mengenkripsi data agar lebih aman.
 - `Hash::make('1234');` → Menghasilkan hash dari password "1234",
10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?
 - digunakan sebagai parameter binding untuk menghindari SQL Injection.
11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode `protected $table = 'm_user';` dan `protected $primaryKey = 'user_id';` ?
 - `protected $table = 'm_user';` : Menentukan bahwa model ini berhubungan dengan tabel `m_user`.
 - `protected $primaryKey = 'user_id';` : Mengubah primary key default (id) menjadi `user_id`.
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Façade / Query Builder / Eloquent ORM*) ? jelaskan
 - Mengurangi boilerplate code (kode berulang-ulang) untuk interaksi database. Menyederhanakan query yang kompleks. Menjaga konsistensi dan keamanan dalam interaksi database.

*** Sekian, dan selamat belajar ***