



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 6 (enam)
Pertemuan ke- : 1 (satu)
Nama : Dhevina Agustina (2341760065)

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**.

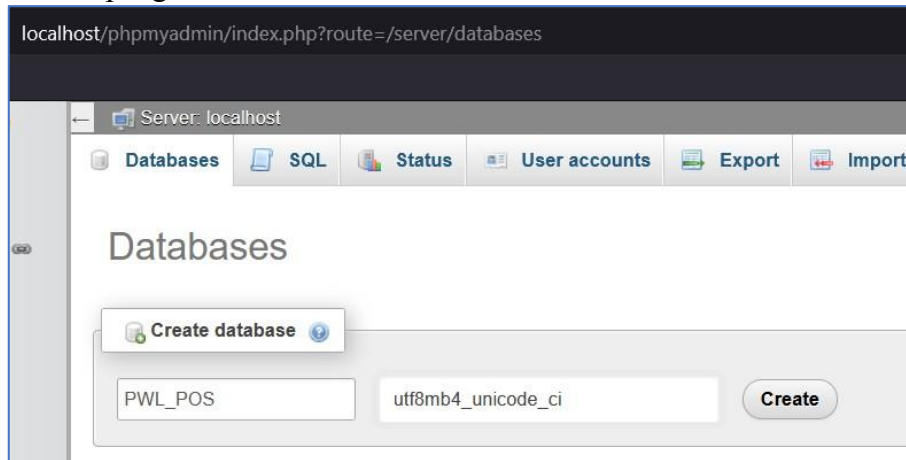
Project PWL_POS akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

A. PENGATURAN DATABASE

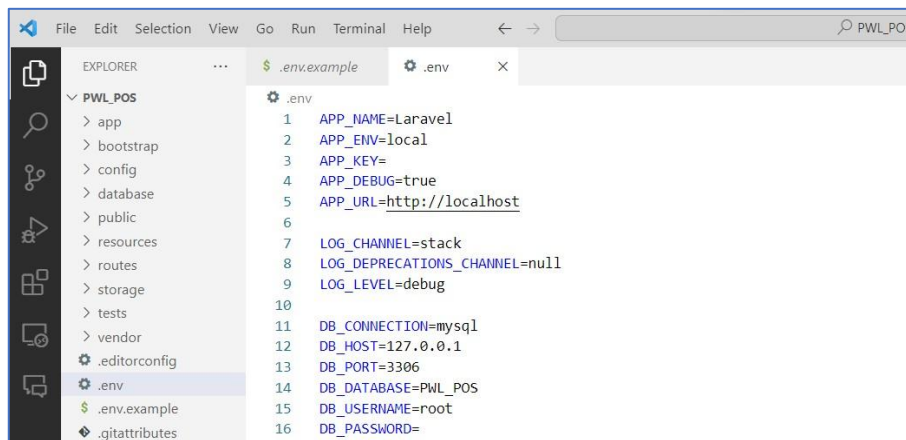
Database atau basis data menjadi komponen penting dalam membangun sistem. Hal ini dikarenakan database menjadi tempat untuk menyimpan data-data transaksi yang ada pada sistem. Koneksi ke database perlu kita atur agar sesuai dengan database yang kita gunakan.



Praktikum ¹ - pengaturan database:



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

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



```
.env
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=
```

```
Minggu 3 > PWL_POS > .env
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:21loUB/xgOMboS2v+EPzxQiVI73CK21haFQaLUaaKvs=
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=
```

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

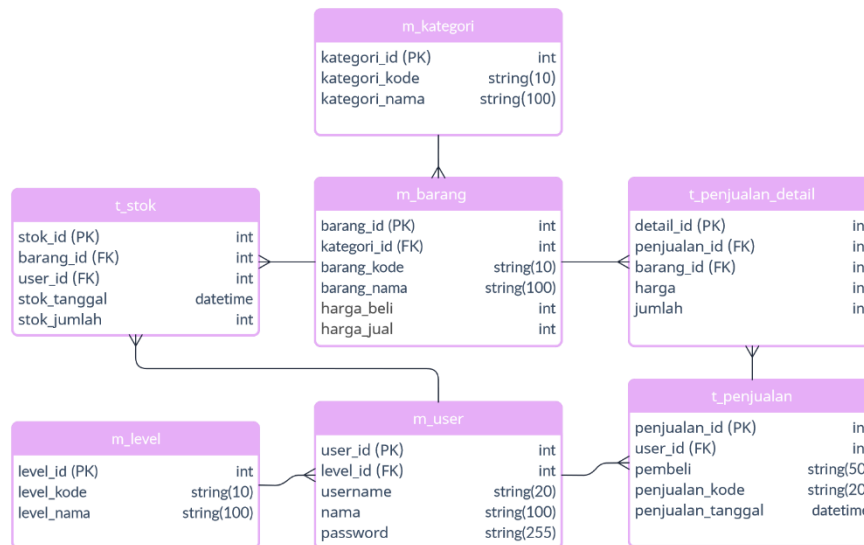
B. MIGRATION

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 dilanjut 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

- a. Menggunakan **artisan** untuk membuat *file migration*

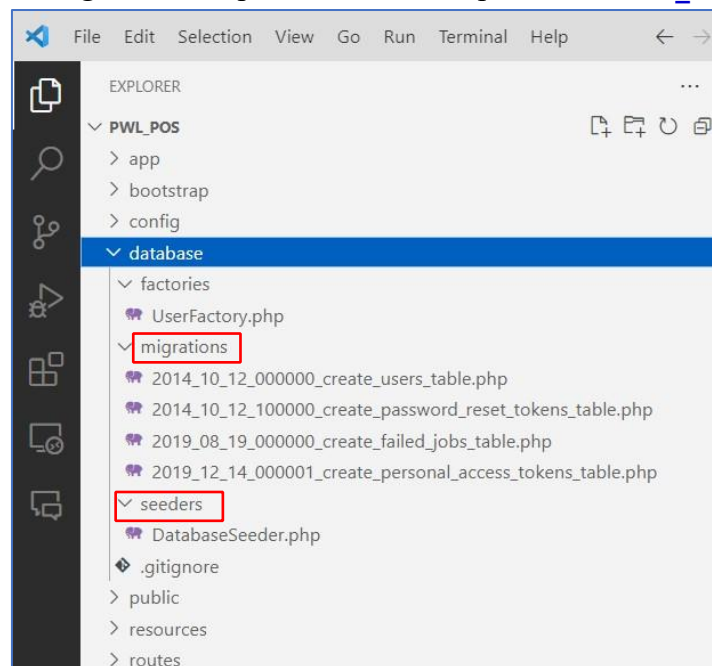
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

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

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

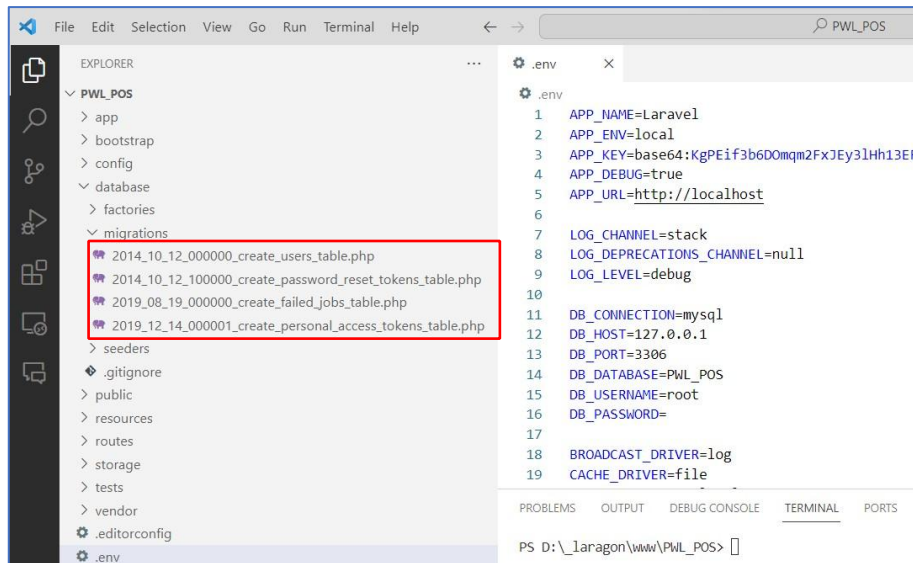
Perintah **-m** di atas adalah *shorthand* untuk opsi membuat file migrasi berdasarkan model yang dibuat.

Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder **PWL_POS/database**



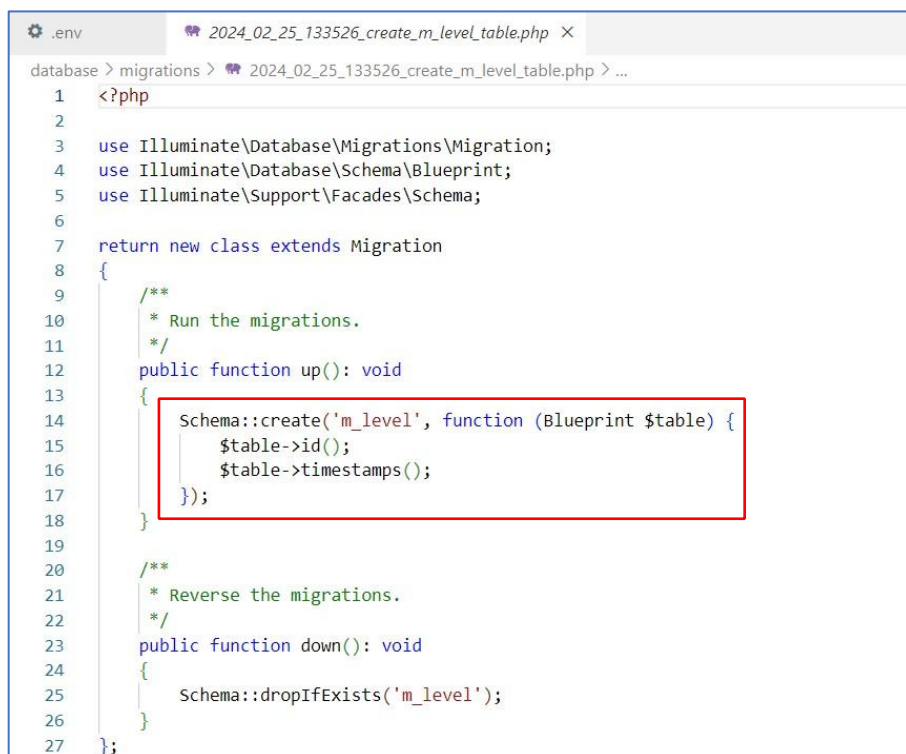
Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

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
```



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 };
```

```
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('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
```

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:\laragon\www\Pemrograman Web-Lanjut_2025\Minggu 3\PWL_POS> php artisan migrate

INFO Preparing database.

Creating migration table 376ms **DONE**

INFO Running migrations.

2014_10_12_000000_create_users_table 259ms **DONE**
2014_10_12_100000_create_password_reset_tokens_table 20ms **DONE**
2019_08_19_000000_create_failed_jobs_table 153ms **DONE**
2019_12_14_000001_create_personal_access_tokens_table 90ms **DONE**
2025_02_27_082123_create_m_level_table 104ms **DONE**

PS D:\laragon\www\Pemrograman Web-Lanjut_2025\Minggu 3\PWL_POS>

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

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	5	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"/> users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
6 tables	Sum	5	InnoDB	utf8mb4_0900_ai_ci	96.0 KiB	0 B

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-2.² ini dan *commit* perubahan pada *git*.

praktikum 2.1 dhevinaagustina

Praktikum 2.³ - Pembuatan file migrasi dengan relasi

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

² . Buka *terminal* VSCode kalian, dan buat file migrasi untuk table **m_user**

³ . 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 };
```

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

m_barang

```
1 POS > database > migrations > 2025_02_28_041101_create_m_barang_table.php > class > up > Closure
2
3 return new class extends Migration
4 {
5     /**
6      * Run the migrations.
7      */
8     public function up(): void
9     {
10         Schema::create('m_barang', function (Blueprint $table) {
11             $table->id('barang_id'); // Primary Key & Auto Increment
12             $table->unsignedBigInteger('kategori_id')->index();
13             $table->string('barang_kode', 10);
14             $table->string('barang_nama', 100);
15             $table->integer('harga_beli');
16             $table->integer('harga_jual');
17             $table->timestamp('created_at');
18             $table->timestamp('updated_at');
19
20             // Foreign Key
21             $table->foreign('kategori_id')->references('kategori_id')->on('m_kategori');
22         });
23     }
24
25     /**
26      * Reverse the migrations.
27      */
28     public function down(): void
29     {
30     }
```

t_penjualan



```
POS > database > migrations > 2025_02_28_041338_create_t_penjualan_table.php > class > up > Closure

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('t_penjualan', function (Blueprint $table) {
            $table->id('penjualan_id');
            $table->unsignedBigInteger('user_id')->index();
            $table->string('pembeli');
            $table->string('penjualan_kode');
            $table->dateTime('penjualan_tanggal');
            $table->timestamp('created_at');
            $table->timestamp('updated_at');

            // Foreign Key
            $table->foreign('user_id')->references('user_id')->on('m_user');
        });

        /**
         * Reverse the migrations.
         */
    }
}
```

```
t_stok
_POS > database > migrations > 2025_02_28_041156_create_t_stok_table.php > class > up > Closure

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('t_stok', function (Blueprint $table) {
            $table->id('stok_id');
            $table->unsignedBigInteger('barang_id')->index();
            $table->unsignedBigInteger('user_id')->index();
            $table->dateTime('stok_tanggal');
            $table->integer('stok_jumlah');
            $table->timestamp('created_at');
            $table->timestamp('updated_at');

            // Foreign Key
            $table->foreign('barang_id')->references('barang_id')->on('m_barang');
            $table->foreign('user_id')->references('user_id')->on('m_user');
        });

        /**
         * Reverse the migrations.
         */
    }
}
```

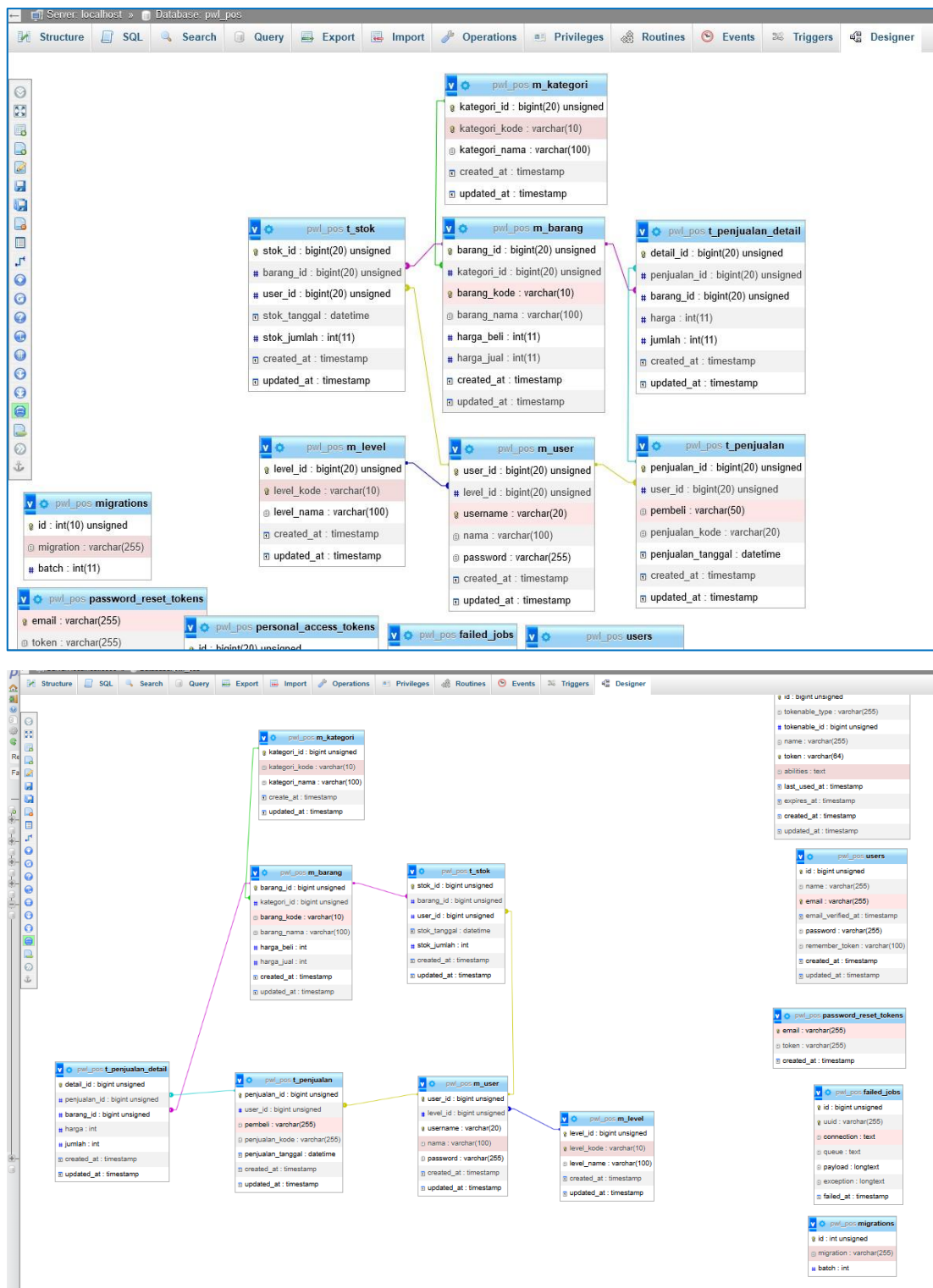
```
t_penjualan_detail

/**
 * Run the migrations.
 */
public function up(): void
{
    Schema::create('t_penjualan_detail', function (Blueprint $table) {
        $table->id('detail_id');
        $table->unsignedBigInteger('penjualan_id')->index();
        $table->unsignedBigInteger('barang_id')->index();
        $table->integer('harga');
        $table->integer('jumlah');
        $table->timestamp('created_at');
        $table->timestamp('updated_at');

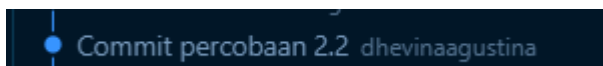
        $table->foreign('penjualan_id')->references('penjualan_id')->on('t_penjualan');
        $table->foreign('barang_id')->references('barang_id')->on('m_barang');
    });

    /**
     * Reverse the migrations.
     */
}
```

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>
```

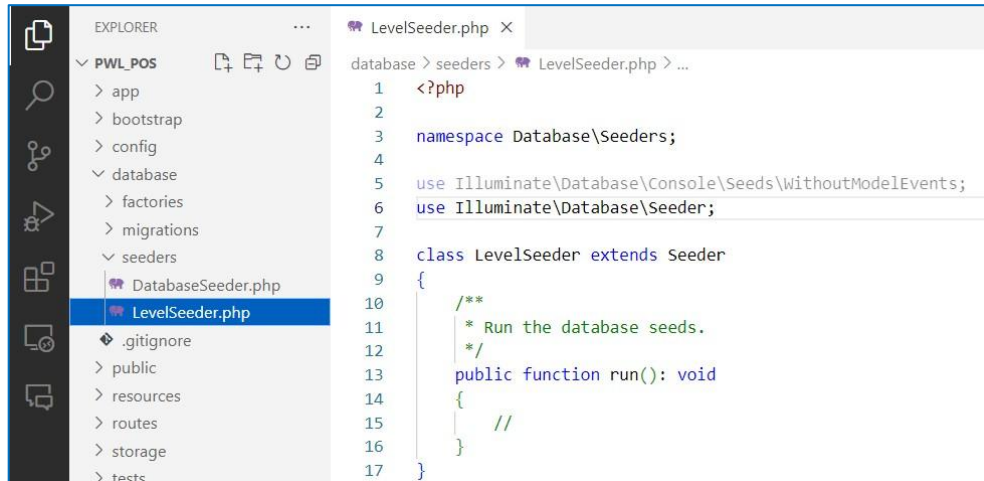
Dalam proses pengembangan suatu aplikasi, seringkali kita membutuhkan data awal tiruan atau *dummy* data untuk memudahkan pengujian dan pengembangan aplikasi kita. Sehingga fitur *seeder* bisa kita pakai dalam membuat sebuah aplikasi web.

Praktikum 3 – Membuat file *seeder*

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

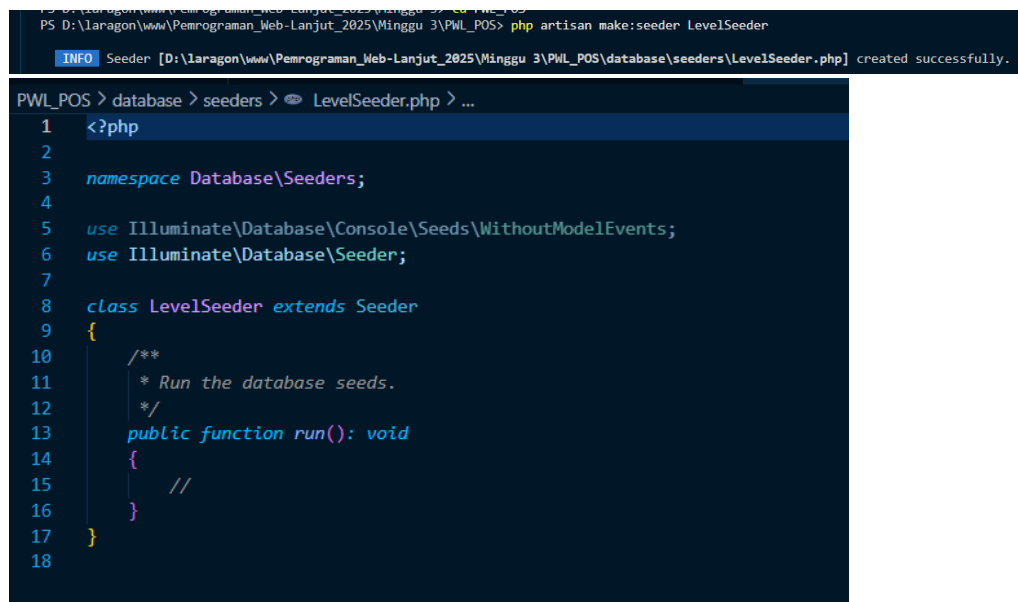


```
php artisan make:seeder LevelSeeder
```



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar displays the file structure of a project named 'PWL_POS'. The 'database' folder is expanded, showing 'seeder' files. 'LevelSeeder.php' is selected. The main editor window shows the content of 'LevelSeeder.php', which is a PHP class extending 'Seeder' from the 'Database\Seeders' namespace. It includes a docblock comment and 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 class LevelSeeder extends Seeder
9 {
10     /**
11      * Run the database seeds.
12      */
13     public function run(): void
14     {
15         //
16     }
17 }
```



The screenshot shows a terminal window at the top with the command 'php artisan make:seeder LevelSeeder' and its successful output. Below the terminal, the VS Code editor shows the same 'LevelSeeder.php' file content as the previous screenshot.

```
PS D:\laragon\www\Pemrograman_Web-Lanjut_2025\Minggu 3\PWL_POS> php artisan make:seeder LevelSeeder
INFO Seeder [D:\laragon\www\Pemrograman_Web-Lanjut_2025\Minggu 3\PWL_POS\database\seeders\LevelSeeder.php] created successfully.

PWL_POS > 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
8 class LevelSeeder extends Seeder
9 {
10     /**
11      * Run the database seeds.
12      */
13     public function run(): void
14     {
15         //
16     }
17 }
18
```




2. Selanjutnya, untuk memasukkan data awal, kita modifikasi file tersebut di dalam function `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;
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 }
```

```
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         $data = [
17             ['level_id' => 1, 'level_kode' => 'ADM', 'level_name' => 'Administrator'],
18             ['level_id' => 2, 'level_kode' => 'MNG', 'level_name' => 'Manager'],
19             ['level_id' => 3, 'level_kode' => 'STF', 'level_name' => 'Staff/Kasir'],
20         ];
21         DB::table('m_level')->insert($data);
22     }
23 }
24
```

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>
```



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
<input type="checkbox"/> Check all With selected: Edit Copy Delete Export								

				level_id	level_kode	level_name	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
```

```
PS D:\laragon\www\Pemrograman_Web-Lanjut_2025\Minggu 3\PWL_POS> php artisan make:seeder UserSeeder
INFO Seeder [D:\laragon\www\Pemrograman_Web-Lanjut_2025\Minggu 3\PWL_POS\database\seeders\UserSeeder.php] created successfully.
PS D:\laragon\www\Pemrograman_Web-Lanjut_2025\Minggu 3\PWL_POS>
```

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 }
```

```
PWL-POS / database / seeders / UserSeeder.php / ...
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
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         $data = [
18             [
19                 'user_id' => 1,
20                 'level_id' => 1,
21                 'username' => 'admin',
22                 'nama' => 'Administrator',
23                 'password' => Hash::make('12345'), //class untuk mengenkripsi/hash password
24             ],
25             [
26                 'user_id' => 2,
27                 'level_id' => 2,
28                 'username' => 'manager',
29                 'nama' => 'Manager',
30                 'password' => Hash::make('12345'),
31             ],
32             [
33                 'user_id' => 3,
34                 'level_id' => 3,
35                 'username' => 'staff',
36                 'nama' => 'Staff/Kasir',
37                 'password' => Hash::make('12345'),
38             ],
39         ];
40         DB::table('m_user')->insert($data);
41     }
42 }
43
```



```
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.4oAKU7FzwS6IXV...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$Ajfns20/FdPTeUgghz31muEhIFaruLxkh5wvZ9NGRpu...
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMY/BHnbJ9W...
<input type="checkbox"/> Check all With selected: <input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete <input type="checkbox"/> Export					

7. Jalankan perintah untuk mengeksekusi class **UserSeeder**

9. Ok, data seeder berhasil di masukkan ke database.

	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\$NmPo4LhC.7vkOrnvKK4t.D2xUOVhPpj0QsX4UKZHEI...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	2	manager	Manager	\$2y\$12\$914aW4CwsOB6tzZi72/...kRNnnJpShKcQWZ6tf5h0...	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	3	staff	Staff/Kasir	\$2y\$12\$W1k1WcmAig9xq6QHmGgOsy5vsN7hqaqoagBj2pNSz...	NULL	NULL

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

- **m_kategori**

```
class KategoriSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            ['kategori_id' => 1, 'kategori_kode' => 'KAT001', 'kategori_nama' => 'Makanan'],
            ['kategori_id' => 2, 'kategori_kode' => 'KAT002', 'kategori_nama' => 'Elektronik'],
            ['kategori_id' => 3, 'kategori_kode' => 'KAT003', 'kategori_nama' => 'Meubel'],
            ['kategori_id' => 4, 'kategori_kode' => 'KAT004', 'kategori_nama' => 'Minuman'],
            ['kategori_id' => 5, 'kategori_kode' => 'KAT005', 'kategori_nama' => 'Perabotan Pecah Belah'],
        ];

        // Insert ke tabel 'm_kategori'
        DB::table('m_kategori')->insert($data);
    }
}
```



		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KAT001	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KAT002	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KAT003	Meubel	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KAT004	Minuman	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KAT005	Perabotan Pecah Belah	NULL	NULL

- m_barang

```
class BarangSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            ['barang_id' => 1, 'barang_kode' => 'B001', 'barang_nama' => 'Laptop', 'harga_beli' => 5000000, 'harga_jual' => 6000000, 'kategori_id' => 1],
            ['barang_id' => 2, 'barang_kode' => 'B002', 'barang_nama' => 'Handphone', 'harga_beli' => 3000000, 'harga_jual' => 3500000, 'kategori_id' => 1],
            ['barang_id' => 3, 'barang_kode' => 'B003', 'barang_nama' => 'Polo', 'harga_beli' => 50000, 'harga_jual' => 75000, 'kategori_id' => 2],
            ['barang_id' => 4, 'barang_kode' => 'B004', 'barang_nama' => 'Pants', 'harga_beli' => 150000, 'harga_jual' => 200000, 'kategori_id' => 2],
            ['barang_id' => 5, 'barang_kode' => 'B005', 'barang_nama' => 'Nasi Bungkus', 'harga_beli' => 25000, 'harga_jual' => 35000, 'kategori_id' => 3],
            ['barang_id' => 6, 'barang_kode' => 'B006', 'barang_nama' => 'Air AQUA', 'harga_beli' => 5000, 'harga_jual' => 10000, 'kategori_id' => 4],
            ['barang_id' => 7, 'barang_kode' => 'B007', 'barang_nama' => 'Kipas Angin', 'harga_beli' => 200000, 'harga_jual' => 300000, 'kategori_id' => 5],
            ['barang_id' => 8, 'barang_kode' => 'B008', 'barang_nama' => 'Rice Cooker', 'harga_beli' => 400000, 'harga_jual' => 500000, 'kategori_id' => 5],
            ['barang_id' => 9, 'barang_kode' => 'B009', 'barang_nama' => 'Speaker Bluetooth', 'harga_beli' => 100000, 'harga_jual' => 150000, 'kategori_id' => 1],
            ['barang_id' => 10, 'barang_kode' => 'B010', 'barang_nama' => 'Teh Botol', 'harga_beli' => 7000, 'harga_jual' => 12000, 'kategori_id' => 4],
        ];
    }
}
```

		barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	B001	Laptop	5000000	6000000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	1	B002	Handphone	3000000	3500000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	2	B003	Polo	50000	75000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	2	B004	Pants	150000	200000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	3	B005	Nasi Bungkus	25000	35000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	4	B006	Air AQUA	5000	10000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	5	B007	Kipas Angin	200000	300000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	5	B008	Rice Cooker	400000	500000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	1	B009	Speaker Bluetooth	100000	150000	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	4	B010	Teh Botol	7000	12000	NULL	NULL

- t_stok

```
public function run(): void
{
    $data = [];
    for ($i = 1; $i <= 10; $i++) {
        $data[] = [
            'barang_id' => $i,
            'stok_tanggal' => now(),
            'stok_jumlah' => rand(10, 50),
            'user_id' => 1,
        ];
    }
    DB::table('t_stok')->insert($data);
}
```

		stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	1	2025-02-28 15:21:56	44	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	2	2	2025-02-28 15:21:56	21	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	3	3	2025-02-28 15:21:56	13	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	4	4	2025-02-28 15:21:56	14	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	5	5	2025-02-28 15:21:56	50	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	6	6	2025-02-28 15:21:56	47	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	7	7	2025-02-28 15:21:56	26	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	8	8	2025-02-28 15:21:56	23	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	9	9	2025-02-28 15:21:56	46	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	10	10	2025-02-28 15:21:56	38	NULL	NULL



- t_penjualan

```
4 public function run(): void
5 {
6
7     $data = [
8         ['penjualan_id' => 1, 'user_id' => 1, 'pembeli' => 'Bobi', 'penjualan_kode' => 'TRX001', 'penjualan_tanggal' => '2023-01-02 10:15:00'],
9         ['penjualan_id' => 2, 'user_id' => 1, 'pembeli' => 'Budi', 'penjualan_kode' => 'TRX002', 'penjualan_tanggal' => '2023-01-05 14:30:00'],
10        ['penjualan_id' => 3, 'user_id' => 1, 'pembeli' => 'Andi', 'penjualan_kode' => 'TRX003', 'penjualan_tanggal' => '2023-01-07 09:45:00'],
11        ['penjualan_id' => 4, 'user_id' => 1, 'pembeli' => 'Sinta', 'penjualan_kode' => 'TRX004', 'penjualan_tanggal' => '2023-01-10 16:00:00'],
12        ['penjualan_id' => 5, 'user_id' => 1, 'pembeli' => 'Dwi', 'penjualan_kode' => 'TRX005', 'penjualan_tanggal' => '2023-01-12 11:20:00'],
13        ['penjualan_id' => 6, 'user_id' => 1, 'pembeli' => 'Fajar', 'penjualan_kode' => 'TRX006', 'penjualan_tanggal' => '2023-01-15 17:10:00'],
14        ['penjualan_id' => 7, 'user_id' => 1, 'pembeli' => 'Vina', 'penjualan_kode' => 'TRX007', 'penjualan_tanggal' => '2023-01-18 13:40:00'],
15        ['penjualan_id' => 8, 'user_id' => 1, 'pembeli' => 'Edo', 'penjualan_kode' => 'TRX008', 'penjualan_tanggal' => '2023-01-20 08:50:00'],
16        ['penjualan_id' => 9, 'user_id' => 1, 'pembeli' => 'Putri', 'penjualan_kode' => 'TRX009', 'penjualan_tanggal' => '2023-01-25 15:05:00'],
17        ['penjualan_id' => 10, 'user_id' => 1, 'pembeli' => 'Eko', 'penjualan_kode' => 'TRX010', 'penjualan_tanggal' => '2023-01-28 12:25:00'],
18    ];
19
20    // Insert data ke tabel 't_penjualan'
21    DB::table('t_penjualan')->insert($data);
22 }
23
24
```

Extra options

		penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
<input type="checkbox"/>	  		1	1 Bobi	TRX001	2023-01-02 10:15:00	NULL	NULL
<input type="checkbox"/>	  		2	1 Budi	TRX002	2023-01-05 14:30:00	NULL	NULL
<input type="checkbox"/>	  		3	1 Andi	TRX003	2023-01-07 09:45:00	NULL	NULL
<input type="checkbox"/>	  		4	1 Sinta	TRX004	2023-01-10 16:00:00	NULL	NULL
<input type="checkbox"/>	  		5	1 Dwi	TRX005	2023-01-12 11:20:00	NULL	NULL
<input type="checkbox"/>	  		6	1 Fajar	TRX006	2023-01-15 17:10:00	NULL	NULL
<input type="checkbox"/>	  		7	1 Vina	TRX007	2023-01-18 13:40:00	NULL	NULL
<input type="checkbox"/>	  		8	1 Edo	TRX008	2023-01-20 08:50:00	NULL	NULL
<input type="checkbox"/>	  		9	1 Putri	TRX009	2023-01-25 15:05:00	NULL	NULL
<input type="checkbox"/>	  		10	1 Eko	TRX010	2023-01-28 12:25:00	NULL	NULL

- t_penjualan_detail

```
9 class PenjualanDetailSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         $penjualan_ids = DB::table('t_penjualan')->pluck('penjualan_id'); // Ambil semua ID penjualan
17         $barang_ids = DB::table('m_barang')->pluck('barang_id'); // Ambil semua ID barang
18
19         $data = [];
20         $detail_id = 1; // Mulai dari 1 untuk detail_id
21
22         foreach ($penjualan_ids as $penjualan_id) {
23             // Ambil 3 barang secara acak untuk setiap transaksi
24             $barang_acak = $barang_ids->shuffle()->take(3);
25
26             foreach ($barang_acak as $barang_id) {
27                 $harga = rand(10000, 50000); // Harga random antara 10.000 - 50.000
28                 $jumlah = rand(1, 5); // Jumlah barang antara 1 - 5
29
30                 $data[] = [
31                     'detail_id' => $detail_id++, // Auto-increment manual
32                     'penjualan_id' => $penjualan_id,
33                     'barang_id' => $barang_id,
34                     'harga' => $harga,
35                     'jumlah' => $jumlah,
36                 ];
37             }
38         }
39
40         // Insert data ke tabel 't_penjualan_detail'
41         DB::table('t_penjualan_detail')->insert($data);
42     }
43 }
44
```




		detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	10	23483	5	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	1	7	16120	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	1	8	23202	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	2	1	20311	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	2	8	31502	5	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	2	2	40994	3	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	7	3	7	23467	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	8	3	1	29275	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	9	3	10	25044	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	10	4	4	43866	5	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	11	4	8	13682	5	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	12	4	1	42891	5	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	13	5	9	41458	4	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	14	5	10	33472	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	15	5	1	42152	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	16	6	6	34169	5	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	17	6	2	37897	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	18	6	1	11927	4	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	19	7	4	44044	4	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	20	7	6	36400	3	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	21	7	2	25236	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	22	8	10	18706	2	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	23	8	1	11089	1	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	24	8	5	44148	5	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	25	9	4	28004	3	NULL	NULL

☐ Check all With selected: Edit Copy Delete Export

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

● Praktikum 3 dhevinaagustina

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

```
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']);
```

mengembalikan (*return*) data hasil *query*. Contoh

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']);
```

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 }
```

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`

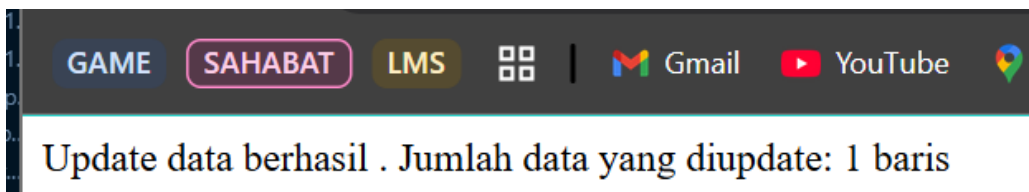
				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	Edit	Copy	Delete	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	Edit	Copy	Delete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

				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
<input type="checkbox"/>	Edit	Copy	Delete	4	CUS	Pelanggan	2025-02-28 15:41:03	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 }
```

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`





	level_id	level_kode	level_name	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	Customer	2025-02-28 15:41:03	NULL

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

```
LevelController.php x 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 }
```

GAME SAHABAT LMS | Gmail YouTube M

Delete data berhasil . Jumlah data yang dihapus: 1 baris

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 }
```



9. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil `view('level')`, maka kita buat file view pada VSCode di

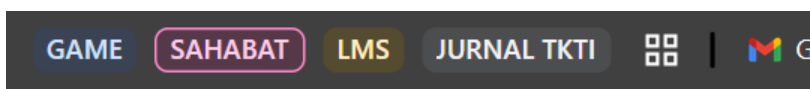
[PWL_POS/resources/view/level.blade.php](#)

```
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>
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi

Jawaban: Menampilkan tabel berisi data level pengguna, dimana setiap baris tabel menampilkan

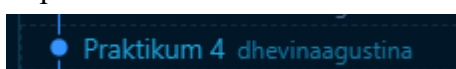
user_id, kode level, nama level.



Data Level Pengguna

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

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





E. QUERY BUILDER

Query builder adalah fitur yang disediakan Laravel untuk melakukan proses CRUD (*create, retrieve/read, update, delete*) pada database. Berbeda dengan *raw query* pada DB Facede 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

- 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');
```

- 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 dahuku 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`

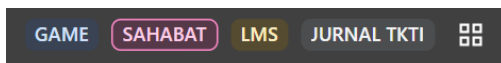
```
LevelController.php  KategoriController.php  level.blade.php  web.php  X
routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\KategoriController;
4  use App\Http\Controllers\LevelController;
5  use Illuminate\Support\Facades\Route;
6
7
8  Route::get('/', function () {
9      return view('welcome');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

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 }
```

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



Insert data baru berhasil

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KAT001	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KAT002	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KAT003	Meubel	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KAT004	Minuman	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KAT005	Perabotan Pecah Belah	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	SNK	Snack/Makanan Ringan	2025-02-28 16:16:31	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 }
```

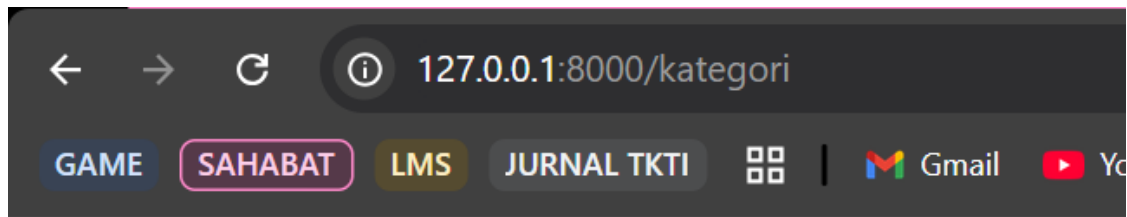
6. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/kategori lagi dan amati apa yang terjadi pada table `m_kategori` di database, *screenshot* perubahan yang ada pada table `m_kategori`

Update data berhasil. Jumlah data yang diupdate: 1 baris

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KAT001	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KAT002	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KAT003	Meubel	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KAT004	Minuman	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KAT005	Perabotan Pecah Belah	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	SNK	Camilan	2025-02-28 16:16:31	NULL

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 }
```



Delete data berhasil. Jumlah data yang diupdate: 1 baris

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KAT001	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KAT002	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KAT003	Meubel	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KAT004	Minuman	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KAT005	Perabotan Pecah Belah	NULL	NULL

Check all With selected: Edit Copy Delete Export

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 }
```

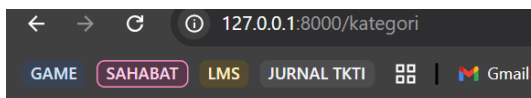
9. Coba kita perhatikan kode yang diberi tanda kotak merah, berhubung kode tersebut memanggil `view('kategori')`, maka kita buat file view pada VSCode di `PWL_POS/resources/view/kategori.blade.php`



```
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>
```

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

Jawab: Menampilkan tabel berisi data m_barang, dimana setiap baris tabel menampilkan ID, kode_kategori, nama_kategori..



Data kategori Barang

ID	Kode Kategori	Nama Kategori
1	KAT001	Makanan
2	KAT002	Elektronik
3	KAT003	Meubel
4	KAT004	Minuman
5	KAT005	Perabotan Pecah Belah

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

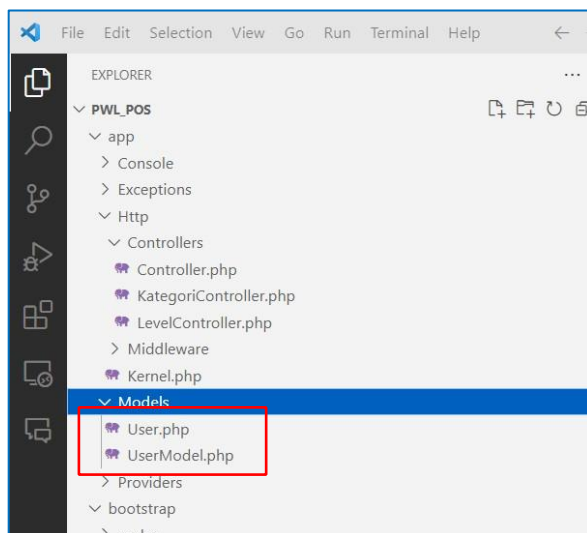
```
php artisan make:model <nama-model-CamelCase>
```

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
```



2. Setelah berhasil generate model, terdapat 2 file pada folder `model` yaitu file `User.php` bawaan dari laravel dan file `UserModel.php` yang telah kita buat. Kali ini kita akan menggunakan file `UserModel.php`



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

```
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
```

```
namespace App\Models;

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

class UserModel extends Model
{
    use HasFactory;

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

4. 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
```

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



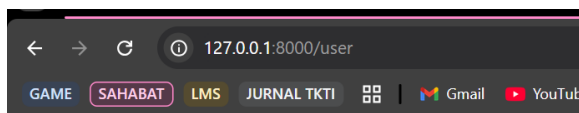
```
app > Http > Controllers > UserController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use App\Models\UserModel;
6  use Illuminate\Http\Request;
7
8  class UserController extends Controller
9  {
10     public function index()
11     {
12         // coba akses model UserModel
13  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

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>
```

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

Jawaban: Menampilkan tabel berisi data `m_user`, dimana setiap baris tabel menampilkan `user_id`, `username`, `nama`, dan `level_id`.



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 }
```

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

Jawaban: terjadi error, karena level_id = 4 tidak ada dalam tabel m_level. Sehingga menyebabkan error, seharusnya data level_id = 4 di tabel m_level harus ada terlebih dulu sebelum memasukkan data ke tabel m_user.

```
Illuminate\Database\QueryException
PHP 8.1.10 10:48:28

SQLSTATE[23000]: Integrity constraint violation: 1452 Cannot add or update a child row: a foreign key constraint fails
('pwl_pos`.`m_user`, CONSTRAINT `m_user_level_id_foreign` FOREIGN KEY (`level_id`) REFERENCES `m_level` (`level_id`))

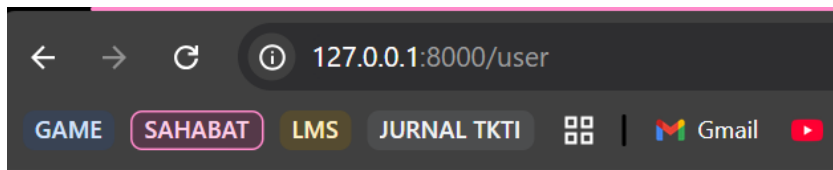
INSERT INTO `m_user` (`username`, `nama`, `password`, `level_id`) VALUES (customer-1, Pelanggan, $2y$12$xmKdxL13uv2wZi2zs21nMeKNC31WBTmW9)
```

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

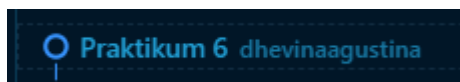
Jawaban: Menampilkan sama seperti sebelumnya, hal ini karena di tabel User username customer-1 tidak ada jadi tidak ada yang diupdate atau dirubah.



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

1. Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari **APP_KEY** pada *file setting .env* Laravel?

Jawaban: APP_KEY di Laravel berfungsi untuk enkripsi data, keamanan session & token, serta validasi CSRF & signed URLs.

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

Jawaban: Dengan menggunakan perintah Artisan berikut untuk menghasilkan APP_KEY: `php artisan key:generate`

3. Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?

Jawaban: Secara default ada 3 file migrasi yang dimiliki oleh Laravel yaitu

- `create_users_table.php`: untuk membuat tabel users, termasuk kolom name, email, password, dan timestamps.
- `create_password_reset_tokens_table.php`: Untuk membuat tabel password_reset_tokens, yang digunakan untuk menyimpan token reset password.



- `create_personal_access_tokens_table.php`: untuk membuat tabel `personal_access_tokens`, yang digunakan oleh Laravel Sanctum untuk autentikasi berbasis token.
4. Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/output dari fungsi tersebut?
- Jawaban: Fungsi `$table->timestamps();` di Laravel digunakan untuk otomatis menambahkan dua kolom ke dalam tabel database
- `created_at` : Menyimpan waktu saat data pertama kali dibuat
 - `updated_at`: Menyimpan waktu saat data terakhir diperbarui.
5. Pada File Migrasi, terdapat fungsi `$table->id();` Tipe data apa yang dihasilkan dari fungsi tersebut?
- Jawaban: Fungsi `$table->id();` pada file migrasi Laravel secara default menghasilkan kolom **id** dengan tipe data **BIGINT (UNSIGNED BIG INTEGER)**. Kolom ini bersifat **auto-increment** dan digunakan sebagai **primary key** dalam tabel.
6. Apa bedanya hasil migrasi pada table `m_level`, antara menggunakan `$table->id();` dengan menggunakan `$table->id('level_id');` ?
- Jawaban: Jika menggunakan `$table->id();`, maka Laravel akan otomatis membuat kolom `id` sebagai primary key dengan tipe data **BIGINT (UNSIGNED BIG INTEGER)** dan **auto-increment**.
- Sedangkan jika menggunakan `$table->id('level_id');`, Laravel akan membuat kolom `level_id` sebagai primary key dengan karakteristik yang sama, hanya saja namanya berubah sesuai yang ditentukan.
7. Pada migration, Fungsi `->unique()` digunakan untuk apa?
- Jawaban: Pada migration, fungsi `->unique()` digunakan untuk memastikan bahwa nilai dalam kolom tersebut tidak boleh duplikat di 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')` ?
- Jawaban: Kolom `level_id` pada tabel `m_level` menggunakan `$table->id('level_id');` karena kolom ini berfungsi sebagai primary key, sehingga otomatis memiliki tipe data **BIGINT (UNSIGNED BIG INTEGER)** dengan **auto-increment**.



Sedangkan, kolom `level_id` pada tabel `m_user` menggunakan `$table->unsignedBigInteger('level_id');` karena berfungsi sebagai foreign key yang merujuk ke `level_id` di tabel `m_level`. Dengan menggunakan `unsignedBigInteger`, tipe datanya disamakan agar cocok dengan tipe primary key di tabel `m_level`, sehingga bisa digunakan untuk relasi antar tabel.

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

Jawaban: Class `Hash` di Laravel digunakan untuk melakukan hashing atau enkripsi satu arah pada data, terutama untuk password agar lebih aman. Fungsi ini memastikan bahwa password tidak disimpan dalam bentuk teks biasa di database, sehingga sulit untuk disalahgunakan.

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

Jawaban: Tanda tanya (?) digunakan sebagai placeholder untuk nilai yang akan di-binding secara otomatis ke dalam query SQL. Tujuannya adalah untuk mencegah SQL Injection dan membuat query lebih aman.

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

Jawaban:

- `protected $table = 'm_user';` digunakan untuk menentukan bahwa model Laravel ini terhubung dengan tabel `m_user` di database, karena secara default Laravel akan mengasumsikan nama tabel mengikuti bentuk jamak dari nama model
- `protected $primaryKey = 'user_id';` digunakan untuk menetapkan primary key tabel sebagai `user_id`, karena secara default Laravel menganggap primary key bernama `id`. Jika primary key memiliki nama berbeda, maka harus ditentukan secara eksplisit.

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

Jawaban: Menurut saya, lebih mudah menggunakan Eloquent ORM karena kita bisa melakukan operasi CRUD dengan lebih sedikit kode, sehingga lebih cepat dan efisien. Selain itu, Eloquent mendukung berbagai relasi antar model seperti satu-ke-satu, satu-ke-banyak, dan banyak-ke-banyak dengan cara yang lebih mudah dipahami. Hal ini membuat pengelolaan data dan logika bisnis menjadi lebih sederhana, karena kita bisa menggunakan model untuk mengatur data dengan lebih baik.



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>

**** Sekian, dan selamat belajar ****