



Mata Kuliah : Pemrograman Web Lanjut (PWL)
Program Studi : D4 – Teknik Informatika / D4 – Sistem Informasi Bisnis
Semester : 4 (empat) / 5 (lima)
Pertemuan ke- : 3 (tiga)
Nama : Muhammad Afiq Firdaus
Kelas / NIM : SIB2B / 2341760189

JOBSHEET 03

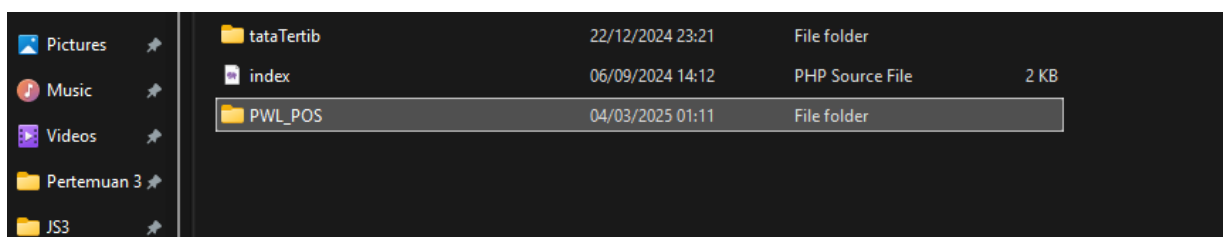
MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, dan ELOQUENT ORM

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



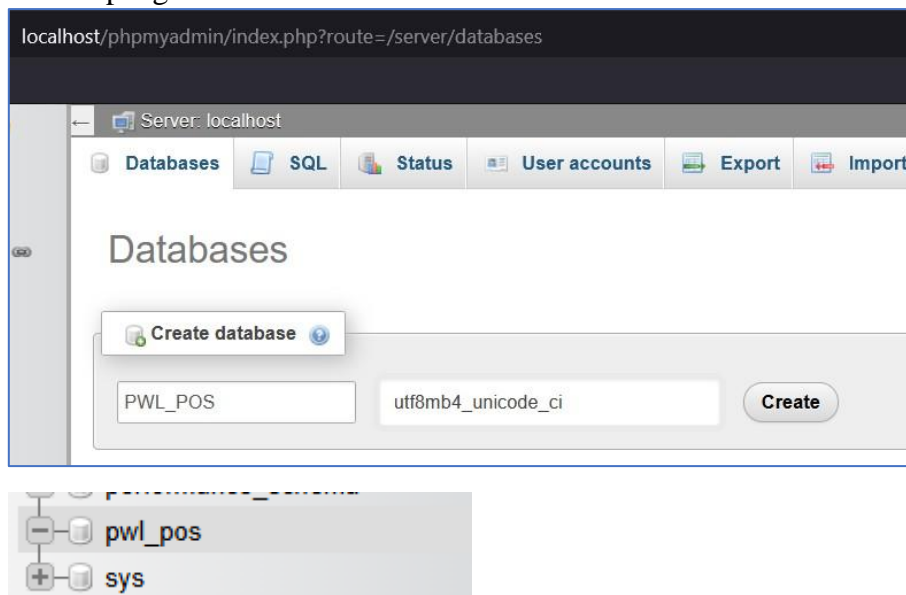
Membuat Project baru



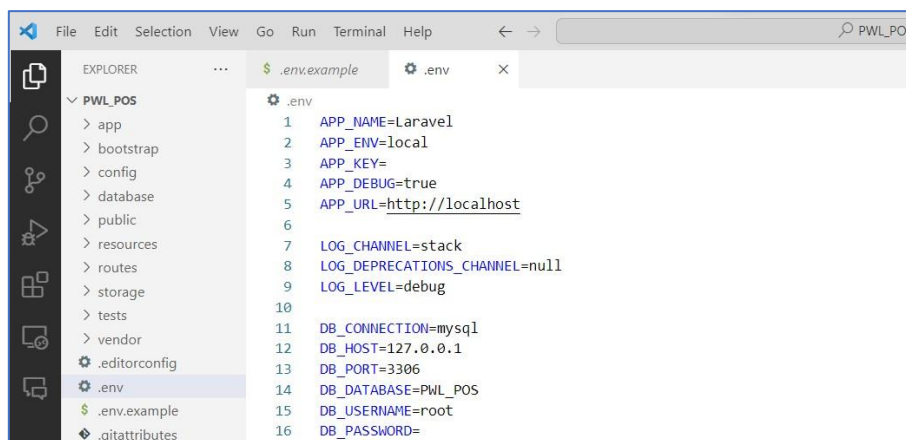
A. PENGATURAN DATABASE

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.

Praktikum¹ - 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**.



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



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

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:KgPEif3b6D0mqm2Fx3Ey3lHh13EFasEJDuxXn9Af22Y=
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=
```

Hasil :

```
1 APP_NAME=Laravel
2 APP_ENV=local
3 APP_KEY=base64:MTuR7LFE5tUyJ5F8I3sc3vF9VrL1Teg8Pdkw8u8Y6IQ=
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=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
22 SESSION_DRIVER=file
23 SESSION_LIFETIME=120
24
```

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

B. MIGRATION

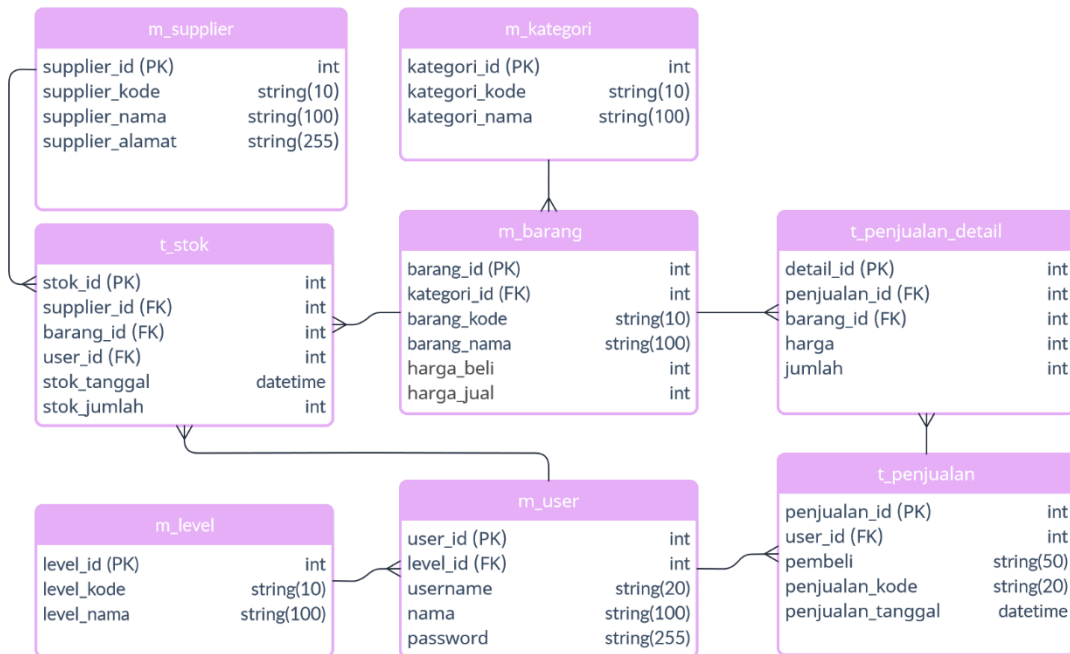
Migration pada Laravel merupakan sebuah fitur yang dapat membantu kita mengelola database secara efisien dengan menggunakan kode program. Migration membantu kita dalam membuat (*create*), mengubah (*edit*), dan menghapus (*delete*) struktur tabel dan kolom pada database yang sudah kita buat dengan cepat 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_supplier	0
4	m_user	1
5	m_barang	2



6	t_penjualan	1
7	t_stok	2
8	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*

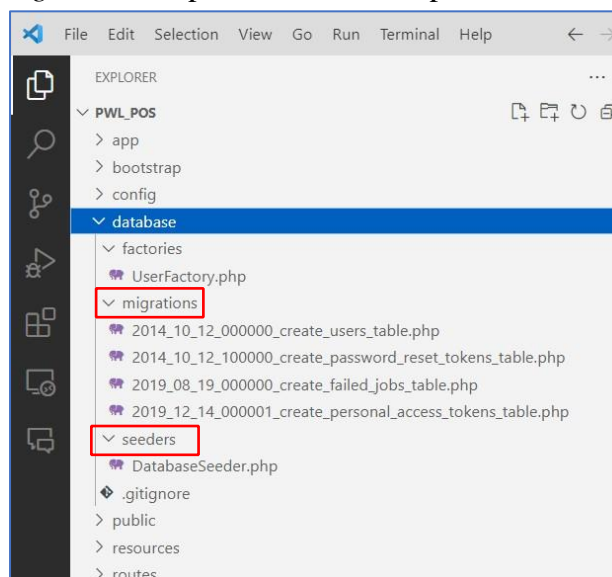
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

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

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

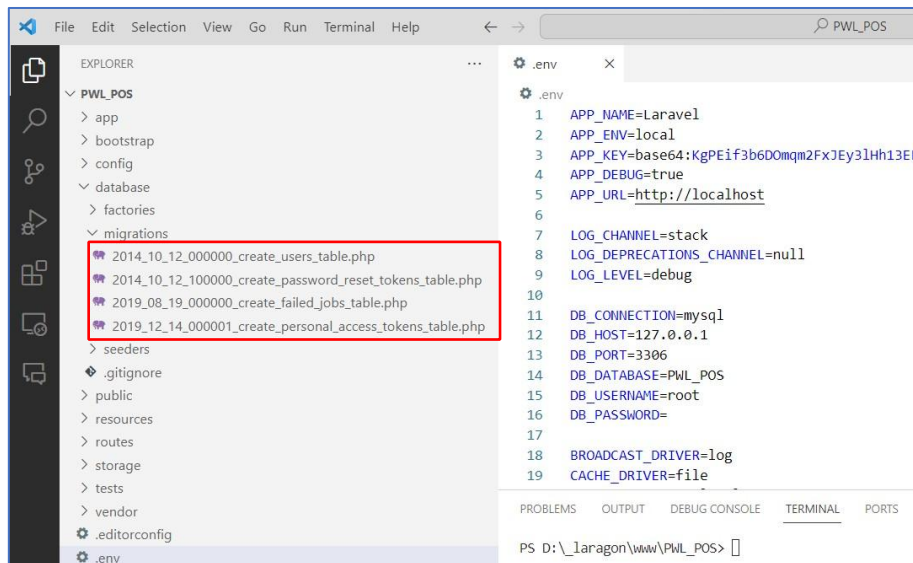
Perintah **-m** di atas adalah *shorthand* untuk opsi membuat file migrasi berdasarkan model yang dibuat.

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



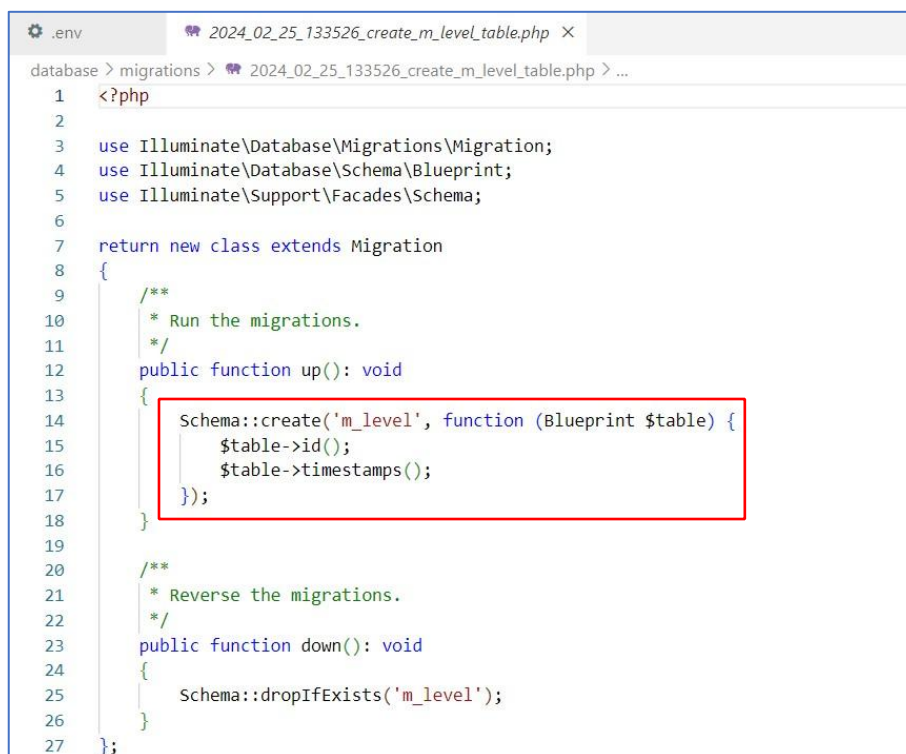
Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

- 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
```





Membuat file migrations melalui terminal

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER AZURE

fafig@AfiaFirdaus MINGW64 /c:/laragon/www/PWL_POS (main)
$ php artisan make:migration create_m_level_table --create=m_level

Warning: PHP Startup: Unable to load dynamic library 'php_sqlsrv_81_nts_x64.dll' (tried: C:/laragon/bin/php/php-8.1.10-Win32-vs16-x64/ext/php_sqlsrv_81_nts_x64.dll, C:/laragon/bin/php/php-8.1.10-Win32-vs16-x64/ext/php_sqlsrv_81_nts_x64.dll.dll (The specified module could not be found)) in Unknown on line 0

[INFO] Migration [C:/laragon/www/PWL_POS/database/migrations/2025_03_03_184458_create_m_level_table.php] created successfully.

fafig@AfiaFirdaus MINGW64 /c:/laragon/www/PWL_POS (main)
$
```

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 };
```

Modifikasi bagian function up()

```
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 /**
```



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>
```

Menyimpan hasil code

```
[INFO] Preparing database.

Creating migration table ..... 25ms DONE

[INFO] Running migrations.

2014_10_12_000000_create_users_table ..... 73ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 19ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 73ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 85ms DONE
2025_03_03_184458_create_m_level_table ..... 57ms DONE
```

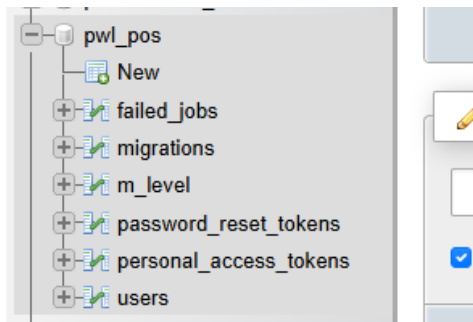
6. Kemudian kita cek di phpMyAdmin apakah table sudah ter-generate atau belum



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

Data base sudah terbuat



8. Buat table *database* dengan *migration* untuk table **m_kategori** dan **m_supplier** yang sama-sama tidak memiliki *foreign key*

Membuat migrasi table m_kategori

```
$ php artisan make:migration create_m_kategori_table --create=m_kategori

Warning: PHP Startup: Unable to load dynamic library 'php_sqlsrv_81_nts_x64.dll' (tried: C:\laragon\bin\php\php-8.1.10-Win32-vs16-x64\ext\php_sqlsrv_81_nts_x64.dll (The specified module could not be found), C:\laragon\bin\php\php-8.1.10-Win32-vs16-x64\ext\php_php_sqlsrv_81_nts_x64.dll.dll (The specified module could not be found)) in Unknown on line 0

INFO Migration [C:\laragon\www\PWL_POS\database\migrations\2025_03_03_185929_create_m_kategori_table.php] created successfully.
```

M_supplier

```
$ php artisan make:migration create_m_supplier_table --create=m_supplier

Warning: PHP Startup: Unable to load dynamic library 'php_sqlsrv_81_nts_x64.dll' (tried: C:\laragon\bin\php\php-8.1.10-Win32-vs16-x64\ext\php_sqlsrv_81_nts_x64.dll (The specified module could not be found), C:\laragon\bin\php\php-8.1.10-Win32-vs16-x64\ext\php_php_sqlsrv_81_nts_x64.dll.dll (The specified module could not be found)) in Unknown on line 0

INFO Migration [C:\laragon\www\PWL_POS\database\migrations\2025_03_03_190222_create_m_supplier_table.php] created successfully.
```



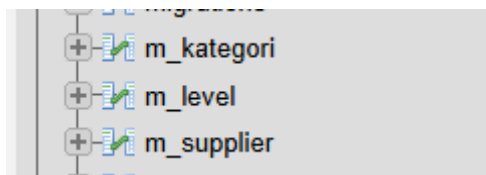
2 table berhasil dibuat

```
p-8.1.10-Win32-vs16-x64/ext\php_sqlsrv_81_nts_x64.dll (The specified module could not be found); C:\laragon\bin\n/php/php-8.1.10-Win32-vs16-x64/ext\php_php_sqlsrv_81_nts_x64.dll (The specified module could not be found)) in Unknown on line 0

INFO Running migrations.

2025_03_03_185929_create_m_kategori_table ..... 22ms DONE
2025_03_03_190222_create_m_supplier_table ..... 23ms DONE
```

Cek database



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
```

Membuat file migrasi berhasil

```
) in Unknown on line 0

INFO Migration [C:\laragon\www\PWL_FOS\database\migrations\2025_03_03_190748_create_m_user_table.php] created successfully.
```

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 }
```

Modifikasi table m_user

```
public function up(): void
{
    Schema::table('m_user', function (Blueprint $table) {
        $table->id('user_id');
        $table->unsignedBigInteger('level_id')->index(); //indexing untuk foreignkey(FK)
        $table->string('username', 20)->unique(); //unique() dibuat agar tidak duplikat
        $table->string('nama', 100);
        $table->string('password');
        $table->timestamps();

        //mendefinisikan FK pada kolom level_id mengacu pada kolom level_id di tabel m_level
        $table->foreign('level_id')->references('level_id')->on('m_level');
    });
}
```

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

Hasil :

```
fafiq@AfqFirdaus MINGW64 /c:/laragon/www/PWL_POS (main)
• $ php artisan migrate

Warning: PHP Startup: Unable to load dynamic library 'php_sqlsrv_81_nts_x64.dll' (tried: C:/laragon/bin/php/php-8.1.10-Win32-vs16-x64/ext/php_sqlsrv_81_nts_x64.dll (The specified module could not be found), C:/laragon/bin/php/php-8.1.10-Win32-vs16-x64/ext/php_php_sqlsrv_81_nts_x64.dll (The specified module could not be found)) in Unknown on line 0

INFO Running migrations.

2025_03_03_192301_create_m_user_table ..... 3ms DONE
```



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

m_barang
t_penjualan
t_stok
t_penjualan_detail

Hasil

```
n/php/php-8.1.10-Win32-vs16-x64/ext\php_php_sqlsrv_81_nts_x64.dll (The specified module could not be found
)) in Unknown on line 0

INFO Running migrations.

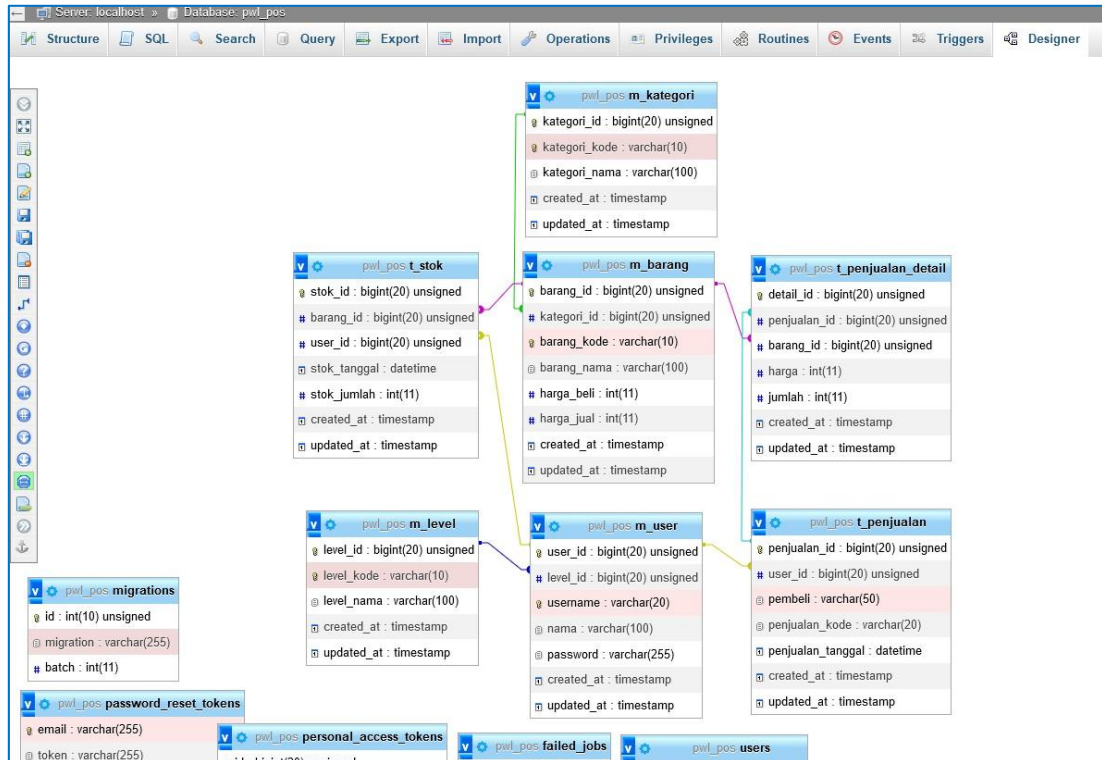
2025_03_03_200126_create_m_barang_table ..... 4ms DONE
2025_03_03_200233_create_t_penjualan_table ..... 0ms DONE
2025_03_03_200409_create_t_stok_table ..... 0ms DONE
2025_03_03_200505_create_t_penjualan_detail_table ..... 0ms DONE
```

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	15	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> m_barang	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> m_kategori	★ Browse Structure Search Insert Empty Drop	0	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"/> m_supplier	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
<input type="checkbox"/> m_user	★ 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"/> t_penjualan	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
<input type="checkbox"/> t_penjualan_detail	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
<input type="checkbox"/> t_stok	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	64.0 KiB	-
<input type="checkbox"/> users	★ Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
13 tables	Sum	15	InnoDB	utf8mb4_0900_ai_ci	320.0 KiB	0 B

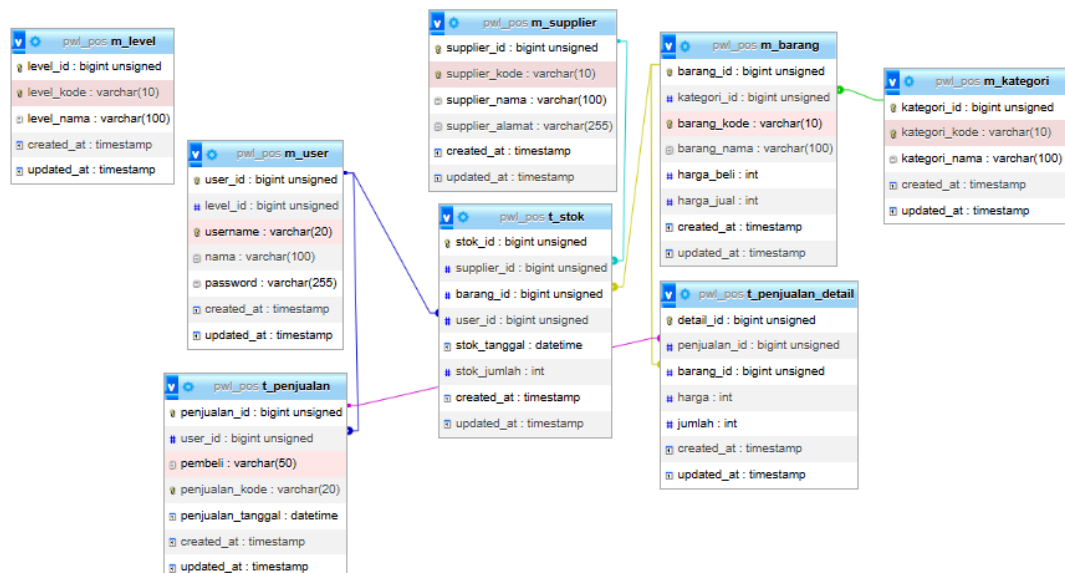
⬆ ☐ Check all With selected: ▼



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



Hasil :





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>
```

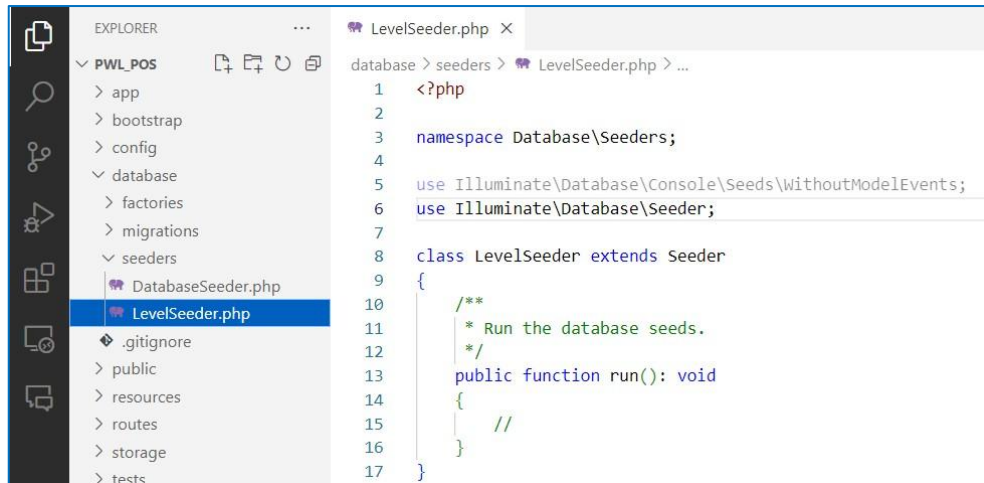
Dalam proses pengembangan suatu aplikasi, seringkali kita membutuhkan data awal tiruan atau *dummy* data untuk memudahkan pengujian dan pengembangan aplikasi kita. Sehingga fitur *seeder* bisa kita pakai dalam membuat sebuah aplikasi web.

Praktikum 3 – Membuat file *seeder*

1. Kita akan membuat file seeder untuk table [m_level](#) dengan mengetikkan perintah



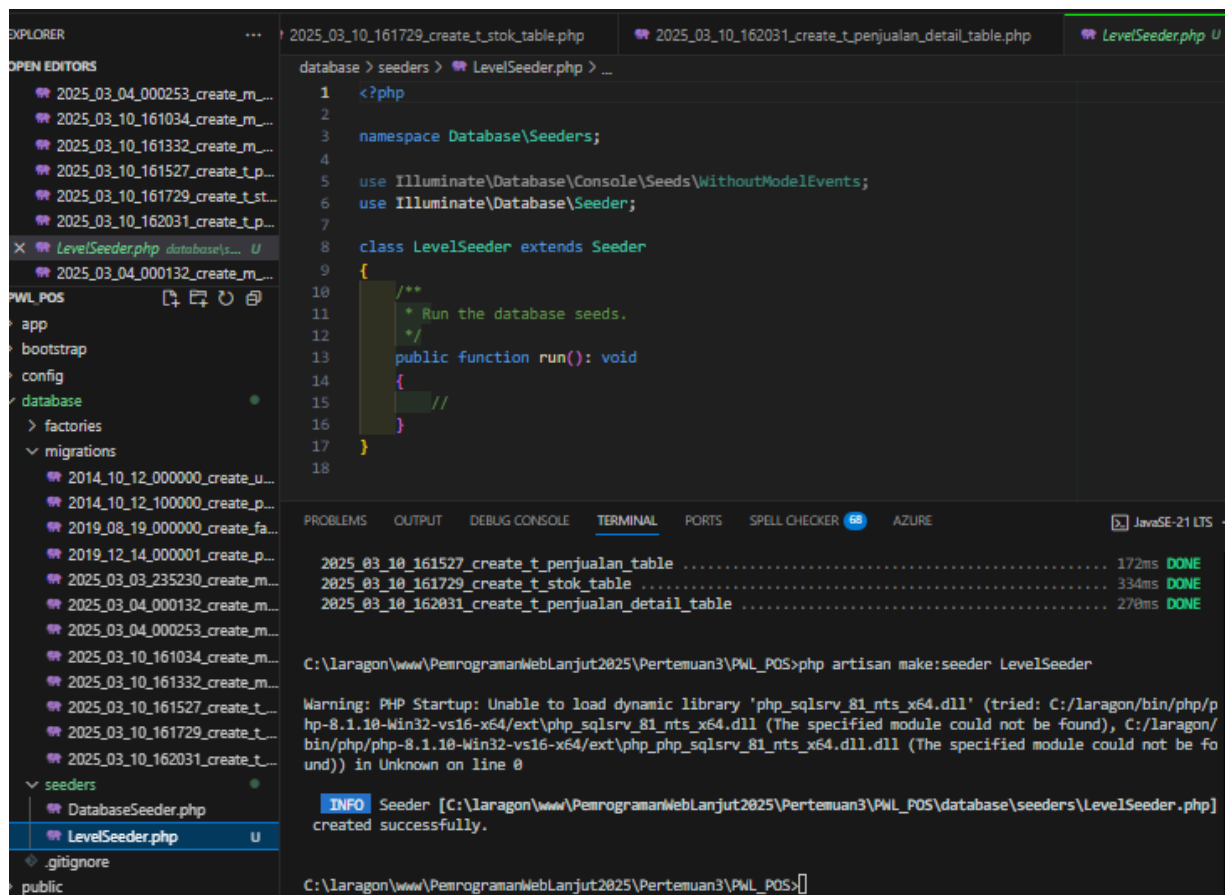
```
php artisan make:seeder LevelSeeder
```



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure of a Laravel project. The 'seeder' directory is expanded, showing 'DatabaseSeeder.php' and 'LevelSeeder.php'. The 'LevelSeeder.php' file is selected and its content is visible in the editor. The code defines a 'LevelSeeder' class that extends 'Seeder' and implements the 'run()' method, which is currently empty. The code is as follows:

```
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 }
```

Hasil :



The screenshot shows the VS Code interface with the Explorer sidebar on the left displaying the file structure of a Laravel project. The 'seeder' directory is expanded, showing 'DatabaseSeeder.php' and 'LevelSeeder.php'. The 'LevelSeeder.php' file is selected and its content is visible in the editor. The code defines a 'LevelSeeder' class that extends 'Seeder' and implements the 'run()' method, which is currently empty. The code is as follows:

```
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 terminal output shows the command 'php artisan make:seeder LevelSeeder' being executed successfully. The output is as follows:

```
2025_03_10_161527_create_t_penjualan_table ..... 172ms DONE
2025_03_10_161729_create_t_stok_table ..... 334ms DONE
2025_03_10_162031_create_t_penjualan_detail_table ..... 278ms DONE

C:\laragon\www\PemrogramanWebLanjut2025\Pertemuan3\PWL_POS>php artisan make:seeder LevelSeeder

Warning: PHP Startup: Unable to load dynamic library 'php_sqlsrv_81_nts_x64.dll' (tried: C:\laragon\bin\php\hp-8.1.10-Win32-vs16-x64\ext\php_sqlsrv_81_nts_x64.dll (The specified module could not be found), C:\laragon\bin\php\php-8.1.10-Win32-vs16-x64\ext\php_php_sqlsrv_81_nts_x64.dll.dll (The specified module could not be found)) in Unknown on line 0

[INFO] Seeder [C:\laragon\www\PemrogramanWebLanjut2025\Pertemuan3\PWL_POS\database\seeders\LevelSeeder.php] created successfully.

C:\laragon\www\PemrogramanWebLanjut2025\Pertemuan3\PWL_POS>
```



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 }
```

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

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



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

Hasil :



```
database > seeders > LevelSeeder.php > LevelSeeder > 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     public function run(): void
12     {
13         $data = [
14             ['level_id' =>1, 'level_kode' => 'ADM','level_nama'=>'Administrator'],
15             ['level_id' =>2, 'level_kode' => 'MNG','level_nama'=>'Manager'],
16             ['level_id' =>3, 'level_kode' => 'STF','level_nama'=>'Staff/Kasir']
17         ];
18         DB::table('m_level')->insert($data);
19     }
20 }
21
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS SPELL CHECKER 72 AZURE JavaSE-21 LTS

C:\laragon\www\PemrogramanWebLanjut2025\Pertemuan3\PWL_POS>php artisan make:seeder LevelSeeder

Warning: PHP Startup: Unable to load dynamic library 'php_sqlsrv_81_nts_x64.dll' (tried: C:/laragon/bin/php/php-8.1.10-Win32-vs16-x64/ext/php_sqlsrv_81_nts_x64.dll (The specified module could not be found), C:/laragon/bin/php/php-8.1.10-Win32-vs16-x64/ext/php_php_sqlsrv_81_nts_x64.dll.dll (The specified module could not be found)) in Unknown on line 0

INFO Seeder [C:\laragon\www\PemrogramanWebLanjut2025\Pertemuan3\PWL_POS\database\seeders\LevelSeeder.php] created successfully.

C:\laragon\www\PemrogramanWebLanjut2025\Pertemuan3\PWL_POS>php artisan db:seed --class=LevelSeeder

Warning: PHP Startup: Unable to load dynamic library 'php_sqlsrv_81_nts_x64.dll' (tried: C:/laragon/bin/php/php-8.1.10-Win32-vs16-x64/ext/php_sqlsrv_81_nts_x64.dll (The specified module could not be found), C:/laragon/bin/php/php-8.1.10-Win32-vs16-x64/ext/php_php_sqlsrv_81_nts_x64.dll.dll (The specified module could not be found)) in Unknown on line 0

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



Hasil :

		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 }
```



```
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 Copy Delete	1	1	admin	Administrator	\$2y\$12\$Tevu4dDO1CUAQpeM6H.Vp.LySwhY.4oAKU7FzwS6fXV...
<input type="checkbox"/>	Edit Copy Delete	2	2	manager	Manager	\$2y\$12\$Ajfns20/FdPTeUgghz31muEhIFaruLxkh5wvZ9NGRpu...
<input type="checkbox"/>	Edit Copy Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Gi23TqGclW5pYeR0VL4o5OxPwb3Osk99VMY/BHnbJ9W...
	<input type="checkbox"/> Check all	With selected:		Edit	Copy	Delete Export

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

Hasil :

```
database > seeders > UseSeeder.php > UseSeeder > run
9
10 class UseSeeder extends Seeder
11 {
12     public function run(): void
13     {
14         $data = [
15             [
16                 'user_id' => 1,
17                 'level_id' => 1,
18                 'username' => 'admin',
19                 'nama' => 'Administrator',
20                 'password' => Hash::make('12345') //class untuk mengenskripsi/hash password
21             ],
22             [
23                 'user_id' => 2,
24                 'level_id' => 2,
25                 'username' => 'manager',
26                 'nama' => 'Manager',
27                 'password' => Hash::make('12345') //class untuk mengenskripsi/hash password
28             ],
29             [
30                 'user_id' => 3,
31                 'level_id' => 3,
32                 'username' => 'staff',
33                 'nama' => 'Staff/Kasir',
34                 'password' => Hash::make('12345') //class untuk mengenskripsi/hash password
35             ],
36         ];
37         DB::table('m_user') -> insert($data);
38     }
39 }
40
41
```

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

Warning: PHP Startup: Unable to load dynamic library 'php_sqlsrv_81_nts_x64.dll' (tried: C:/laragon/bin/php/p
hp-8.1.10-Win32-vs16-x64/ext/php_sqlsrv_81_nts_x64.dll (The specified module could not be found), C:/laragon/
bin/php/php-8.1.10-Win32-vs16-x64/ext/php_php_sqlsrv_81_nts_x64.dll.dll (The specified module could not be fo
und)) in Unknown on line 0

INFO Seeding database.

faFic@AfiqFirdaus MINGW64 /c:/laragon/www/PemrogramanWebLanjutan2025/Pertemuan3/PML_POS (main)
$
```

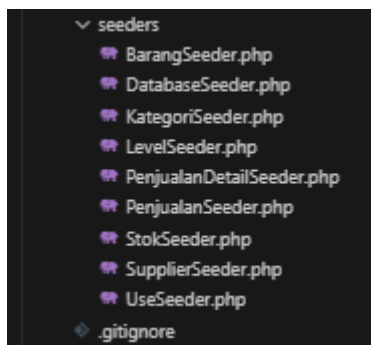


9. Ok, data seeder berhasil di masukkan ke database.

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_supplier	3	3 supplier barang
3	m_barang	15	15 barang berbeda (5 barang/supplier)
4	t_stok	15	Stok untuk 15 barang
5	t_penjualan	10	10 transaksi penjualan
6	t_penjualan_detail	30	3 barang untuk setiap transaksi penjualan

Hasil :



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**

```
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']);
```

(*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`

```
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**

<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
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	CUS	Pelanggan	2024-02-26 08:20:00	NULL

Hasil :

Insert data baru berhasil

				level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>				1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>				2	MNG	Manager	NULL	NULL
<input type="checkbox"/>				3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>				4	cus	pelanggan	2025-03-10 18:58:16	NULL
<input type="checkbox"/> Check all With selected:								

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 }
```

Hasil :

```
10 {
11     public function index(){
12         //Awal
13         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?,?,?)', ['cus', 'pelan
14         // return 'Insert data baru berhasil';
15
16         //Setelah Modifikasi
17         $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'cus']);
18         return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
19     }
20 }
21 }
22 }
```

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`

Hasil

:

Update data berhasil. Jumlah data yang diupdate: 1 baris



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 }
```

Hasil :

```
20
21 //Modifikasi 2
22 $row = DB::delete('delete from m_level where level_kode = ?',['cus']);
23 return 'Delete data berhasil. jumlah data yang dihapus: ' . $row . 'baris';
24
25 }
```

localhost/PemrogramanWebLanjut2025/Pertemuan3/PWL_POS/public/level

Delete data berhasil. jumlah data yang dihapus: 1baris

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 }
```



Hasil :

```
10 {  
11     public function index()  
12     {  
13         //Awal  
14         // DB::insert('insert into m_level(level_kode, level_nama, created_at) val  
15         // return 'Insert data baru berhasil';  
16  
17         //Setelah Modifikasi  
18         // $row = DB::update('update m_level set level_nama = ? where level_kode =  
19         // return 'Update data berhasil.jumlah data yang diupdate: ' . $row. 'baris'  
20  
21         //Modifikasi 2  
22         // $row = DB::delete('delete from m_level where level_kode = ?', ['cus']);  
23         // return 'Delete data berhasil. jumlah data yang dihapus: ' . $row. 'baris'  
24  
25         //Modifikasi 3  
26         $data = DB::select('select * from m_level');  
27         return view('level', ['data' => $data]);  
28     }  
29 }
```

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>
```

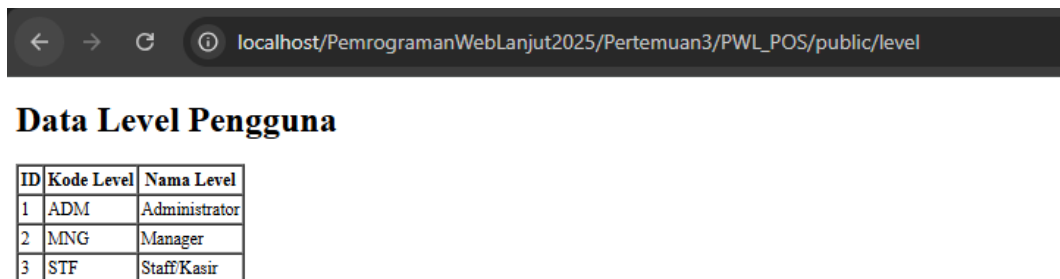
View :



```
resources > views > level.blade.php > ...
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5 <title>Data Level Pengguna</title>
6 </head>
7
8 <body>
9 <h1>Data Level Pengguna</h1>
10 <table border="1" cellpadding="2" cellspacing="0">
11 <tr>
12 <th>ID</th>
13 <th>Kode Level</th>
14 <th>Nama Level</th>
15 </tr>
16 @foreach ($data as $d)
17 <tr>
18 <td>{{ $d->level_id }}</td>
19 <td>{{ $d->level_kode }}</td>
20 <td>{{ $d->level_nama }}</td>
21 </tr>
22 @endforeach
23 </table>
24 </body>
25
26 </html>
27
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi

Hasil :



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

- 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']);
```

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



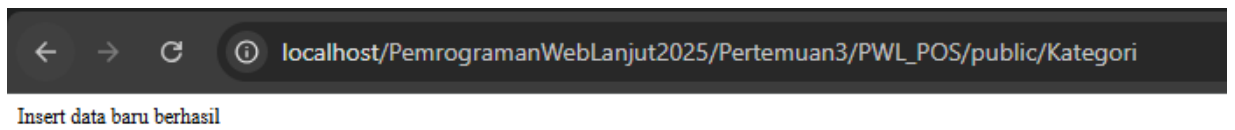
```
LevelController.php KategoriController.php X level.blade.php web.php
app > Http > Controllers > KategoriController.php > KategoriController > index:
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController 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 }
```

Hasil :

```
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  use App\Http\Controllers\UserProfileController;
7
8  /*
9  |-----
10 | Web Routes
11 |-----
12 |
13 | Here is where you can register web routes for your application. These
14 | routes are loaded by the RouteServiceProvider and all of them will
15 | be assigned to the "web" middleware group. Make something great!
16 |
17 */
18
19 Route::get('/', function () {
20     return view('welcome');
21 });
22
23 Route::get('/level', [LevelController::class, 'index']);
24 Route::get('/kategori', [KategoriController::class, 'index']);
25
```

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`

Hasil :

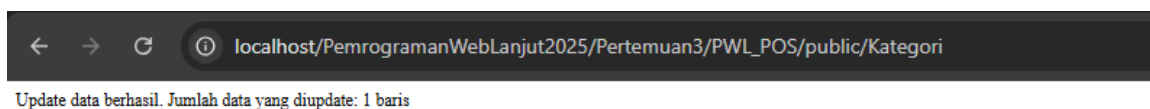


5. Selanjutnya, kita modifikasi lagi file `KategoriController` untuk meng-*update* data di table `m_kategori` seperti berikut

```
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`

Hasil :



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 }
```

Hasil :

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

        // DB::table('m_kategori')->insert($data);

        // return 'Insert data baru berhasil';

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

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

localhost/PemrogramanWebLanjut2025/Pertemuan3/PWL_POS/public/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

Hasil :

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

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

Hasil :



localhost/PemrogramanWebLanjut2025/Pertemuan3/PWL_POS/public/Kategori

Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	ELK	Elektronik
2	PAK	Pakaian
3	MNM	Makanan/Minuman
4	PRT	Peralatan Rumah Tangga
5	KES	Kesehatan/Kecantikan

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>
```

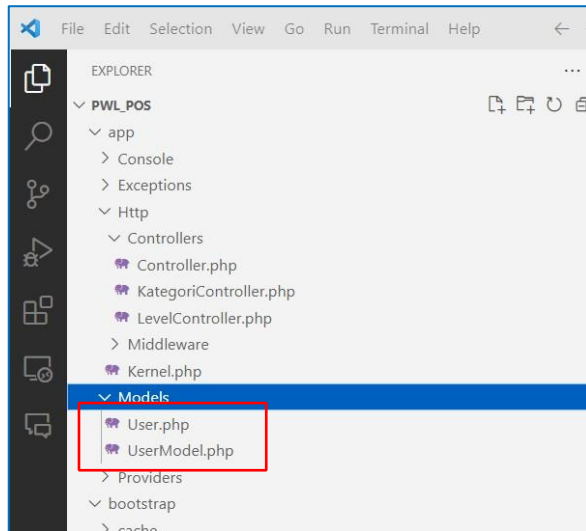
Untuk bisa melakukan operasi **CRUD** (*create, read/retrieve, update, delete*), kita harus membuat sebuah model sesuai dengan target tabel yang ingin digunakan. Jadi, **dalam 1 model, merepresentasikan 1 tabel database.**

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
```

- 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 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

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

Hasil :

localhost/PemrogramanWebLanjut2025/Pertemuan3/PWL_POS/public/user			
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?

Jawab :

`APP_KEY` pada file `.env` Laravel berfungsi untuk menjaga keamanan aplikasi, terutama dalam proses enkripsi data seperti session cookies dan token CSRF. Kunci ini digunakan untuk memastikan bahwa data yang terenkripsi hanya dapat diakses dengan kunci yang benar. Jika kunci tidak diatur atau diubah, data terenkripsi tidak bisa dibaca dan aplikasi mungkin menjadi tidak aman. Untuk menghasilkan kunci baru, dapat digunakan perintah `php artisan key:generate`.

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

Jawab :

Gunakan `php artisan key:generate`

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

Jawab :

1. **2014_10_12_000000_create_users_table**

- Digunakan untuk membuat tabel users, yang menyimpan data pengguna (user) aplikasi, seperti nama, email, password, dan timestamp.

2. **2014_10_12_100000_create_password_reset_tokens_table**

- Digunakan untuk membuat tabel password_reset_tokens, yang menyimpan token untuk proses reset password ketika pengguna ingin mengubah kata sandi mereka.

3. **2019_08_19_000000_create_failed_jobs_table**

- Digunakan untuk membuat tabel failed_jobs, yang menyimpan informasi tentang job yang gagal diproses dalam sistem queue Laravel.

4. **2019_12_14_000001_create_personal_access_tokens_table**

- Digunakan untuk membuat tabel personal_access_tokens, yang menyimpan token akses pribadi yang digunakan untuk mengelola autentikasi API di Laravel menggunakan fitur Laravel Sanctum.

4. Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/output dari fungsi tersebut?



Jawab :

Kode ``$table->timestamps();`` pada migrasi Laravel berfungsi untuk secara otomatis menambahkan dua kolom timestamp ke dalam tabel, yaitu:

1. `created_at`: Menyimpan waktu ketika record (baris data) pertama kali dibuat.
2. `updated_at`: Menyimpan waktu ketika record terakhir kali diperbarui.

Kolom-kolom ini sangat berguna untuk melacak kapan sebuah data dibuat dan kapan terakhir kali diperbarui, yang sering diperlukan dalam pengelolaan data. Laravel akan secara otomatis mengisi dan memperbarui nilai kedua kolom ini ketika record ditambahkan atau diperbarui.

5. Pada File Migrasi, terdapat fungsi `$table->id();` Tipe data apa yang dihasilkan dari fungsi tersebut?

Jawab :

Fungsi ``$table->id();`` pada file migrasi Laravel menghasilkan kolom dengan tipe data BIGINT (big integer) yang otomatis dijadikan sebagai primary key untuk tabel tersebut. Kolom ini juga bersifat auto-increment, sehingga nilai ID akan bertambah secara otomatis setiap kali record baru ditambahkan.

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

Jawab :

Perbedaan hasil migrasi pada tabel ``m_level`` antara menggunakan ``$table->id();`` dan ``$table->id('level_id');`` terletak pada nama kolom primary key yang dihasilkan:

1. `$table->id();`
 - Akan membuat kolom dengan nama ``id`` sebagai primary key dan auto-increment
 - Default tipe data adalah BIGINT .
2. `$table->id('level_id');`
 - Akan membuat kolom dengan nama ``level_id`` sebagai primary key dan autoincrement .
 - Default tipe data tetap BIGINT , tetapi nama kolomnya berubah menjadi ``level_id``



Intinya, perbedaannya hanya pada nama kolom yang digunakan sebagai primary key. Fungsi dan tipe datanya tetap sama.

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

Jawab :

Pada migrasi Laravel, fungsi `->unique()` digunakan untuk menetapkan **kolom** sebagai kolom **unik**, yang berarti bahwa **nilai-nilai dalam kolom tersebut tidak boleh ada yang duplikat**. Setiap nilai yang dimasukkan ke kolom yang diatur sebagai unik harus berbeda dari nilai-nilai lainnya dalam kolom tersebut.

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

Jawab :

1. Pada tabel `m_level`, `$table->id('level_id')` digunakan untuk mendefinisikan **primary key**.
 2. Pada tabel `m_user`, `$table->unsignedBigInteger('level_id')` digunakan untuk mendefinisikan **foreign key** yang mengacu pada `level_id` di tabel `m_level`.
9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');`?

Jawab :

Tujuan dari **Class Hash** di Laravel adalah untuk menyediakan fungsi-fungsi yang terkait dengan **hashing** (pembuatan hash) dan **verifikasi hash**, terutama untuk memastikan keamanan data sensitif, seperti password. Dengan hashing, data akan diubah menjadi bentuk yang tidak dapat dibaca dan sulit untuk direkonstruksi kembali, sehingga lebih aman dari serangan seperti pencurian data.

Maksud dari kode `Hash::make('1234');`:

Kode ini digunakan untuk **meng-hash** string `'1234'`. Fungsi `Hash::make()` akan mengubah string tersebut menjadi hash yang lebih aman untuk penyimpanan, misalnya ketika menyimpan password ke dalam database.

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

Jawab :



Pada **query builder** di Laravel, tanda tanya (?) digunakan sebagai **placeholder** dalam query untuk menghindari risiko **SQL injection**. Placeholder ini akan digantikan dengan nilai yang aman secara otomatis oleh query builder ketika query dijalankan.

Penggunaan tanda tanya (?) membantu dalam **binding parameter** secara aman. Nilainilai yang dimasukkan ke dalam query akan di-*sanitize* sehingga tidak bisa dimanipulasi untuk merusak query atau melakukan serangan berbahaya.

11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode `protected $table =`

`'m_user';` dan `protected $primaryKey = 'user_id';` ?

Jawab :

Penulisan `protected $table = 'm_user';` dalam model Laravel digunakan untuk menetapkan nama tabel database yang berbeda dari default. Sementara `protected $primaryKey = 'user_id';` digunakan untuk menentukan kolom primary key yang berbeda dari default (`'id'`). Ini memungkinkan model untuk berfungsi dengan tabel dan kolom primary key yang tidak sesuai dengan konvensi default Laravel.

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

Jawab :

Saya lebih suka memakai Query Builder karena tidak perlu menulis query Panjang seperti di DB Façade, kemudian tatanan code yang lebih rapi dan mudah sekali dibaca sehingga memudahkan saya untuk memahami maksud dari query yang ingin saya buat

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