



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

JOBSHEET 03

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

Sebelumnya kita sudah membahas mengenai *Routing*, *Controller*, dan *View* yang ada di Laravel. Sebelum kita masuk pada pembuatan aplikasi berbasis website, alangkah baiknya kita perlu menyiapkan Basis data sebagai tempat menyimpan data-data pada aplikasi kita nanti. Selain itu, umumnya kita perlu menyiapkan juga data awal yang kita gunakan sebelum membuat aplikasi, seperti data user administrator, data pengaturan sistem, dll. Untuk itu, kita memerlukan teknik untuk merancang/membuat table basis data sebelum membuat aplikasi. Laravel memiliki fitur dalam pengelolaan basis data seperti, migration, seeder, model, dll.

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

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

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

A. PENGATURAN DATABASE

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



Praktikum 1 - pengaturan database:

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

Create database

PWL_POS utf8mb4_general_ci Create

2. Buka aplikasi VSCode dan buka folder project **PWL_POS** yang sudah kita buat

```
PS C:\Users\nizar> composer create-project laravel/laravel PWL_POS
Creating a "laravel/laravel" project at "./PWL_POS"
Installing laravel/laravel (v10.3.3)
- Installing laravel/laravel (v10.3.3): Extracting archive
Created project in C:\Users\nizar\PWL_POS
```

3. Copy file **.env.example** menjadi **.env**
4. Buka file **.env**, dan pastikan konfigurasi **APP_KEY** bernilai. Jika belum bernilai silahkan kalian *generate* menggunakan **php artisan**.

```
.env
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:Wtnip1CS28XuNrDteVZpAmmyNB1hQITgyZxUEbnoI7c=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=root

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailpit
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
```

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

```
.env
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:Wtnip1CS28XuNrDteVZpAmmyNB1hQITgyZxUEbnoI7c=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack
LOG_DEPRECATIONS_CHANNEL=null
LOG_LEVEL=debug

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=PWL_POS
DB_USERNAME=root
DB_PASSWORD=root

BROADCAST_DRIVER=log
CACHE_DRIVER=file
FILESYSTEM_DISK=local
QUEUE_CONNECTION=sync
SESSION_DRIVER=file
SESSION_LIFETIME=120

MEMCACHED_HOST=127.0.0.1

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_MAILER=smtp
MAIL_HOST=mailpit
MAIL_PORT=1025
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS="hello@example.com"
```

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

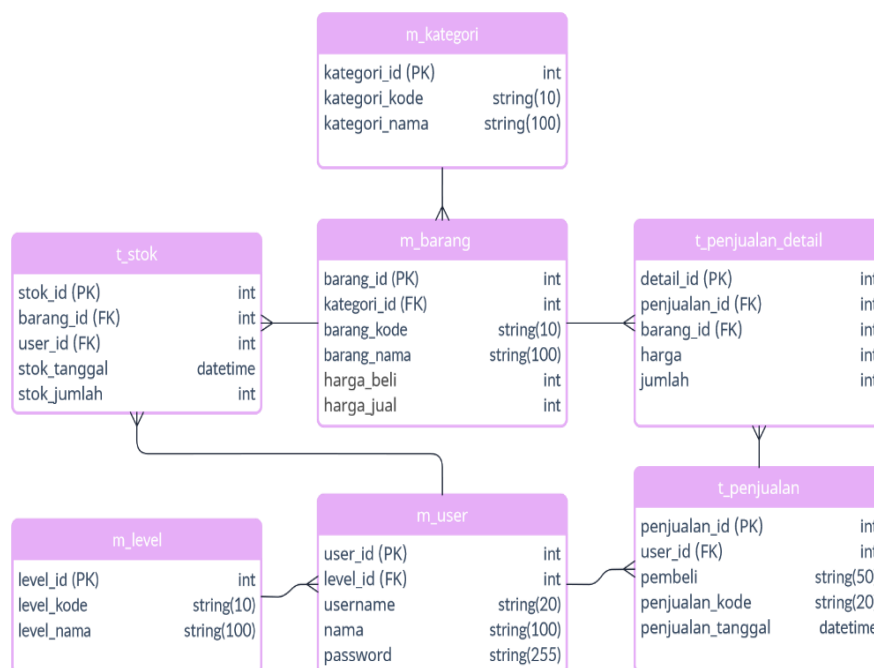


B. MIGRATION

Migration pada Laravel merupakan sebuah fitur yang dapat membantu kita mengelola database secara efisien dengan menggunakan kode program. Migration membantu kita dalam membuat (*create*), mengubah (*edit*), dan menghapus (*delete*) struktur tabel dan kolom pada database yang sudah kita buat dengan cepat dan mudah. Dengan Migration, kita juga dapat melakukan perubahan pada struktur database tanpa harus menghapus data yang ada. Salah satu keunggulan menggunakan migration adalah mempermudah proses instalasi aplikasi kita, Ketika aplikasi yang kita buat akan diimplementasikan di server/komputer lain.

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

Studi Kasus PWL.pdf



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

TIPS MIGRATION

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



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

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

INFO

Secara default Laravel sudah ada table [users](#) untuk menyimpan data pengguna, tapi pada praktikum ini, kita gunakan table sesuai dari file [Studi Kasus PWL.pdf](#) yaitu [m_user](#).

Pembuatan file migrasi bisa menggunakan 2 cara, yaitu

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

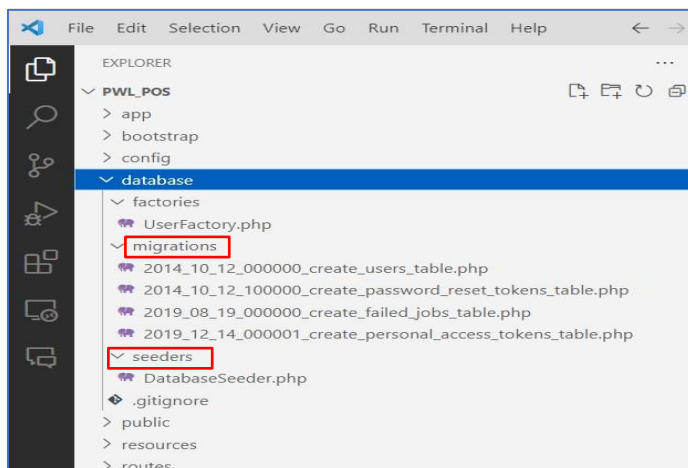
```
php artisan make:migration <nama-file-tabel> --create=<nama-tabel>
```

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

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

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

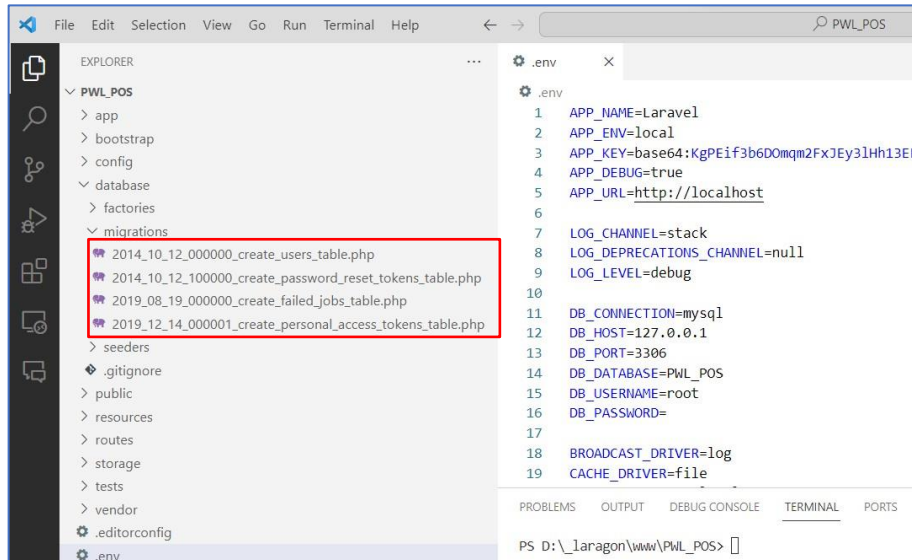
Pada Laravel, file-file *migration* ataupun *seeder* berada pada folder [PWL_POS/database](#)





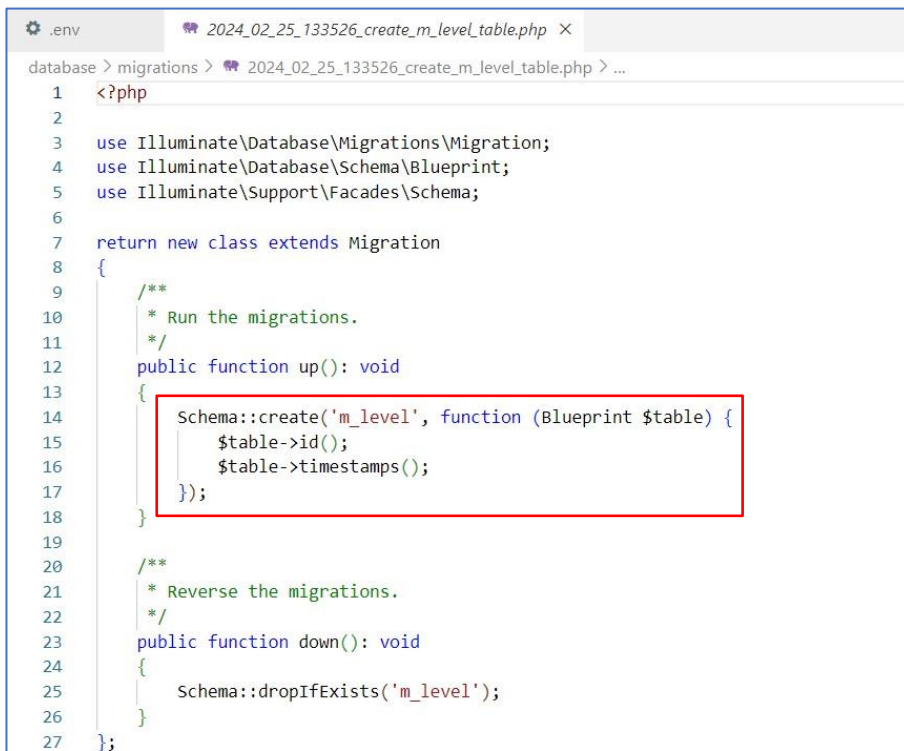
Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

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



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

```
PS C:\laragon\www\FWL_POS> php artisan make:migration create_m_level_table --create=m_level
INFO Migration [C:\laragon\www\FWL_POS\database\migrations\2024_03_10_044620_create_m_level_table.php] created successfully.
PS C:\laragon\www\FWL_POS>
```





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

```
database > migrations > 2024_03_10_044620_create_m_level_table.php > class > up > Closure
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
26     {
27         Schema::dropIfExists('m_level');
28     }
29 };
```

INFO

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

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

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

```
PS C:\laragon\www\FWL_POS> php artisan migrate

INFO Preparing database.

Creating migration table ..... 17ms DONE

INFO Running migrations.

2014_10_12_000000_create_users_table ..... 50ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 17ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 65ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 64ms DONE
2024_03_10_044620_create_m_level_table ..... 40ms DONE

PS C:\laragon\www\FWL_POS>
```



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

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

7. Ok, table sudah dibuat di database
8. Buat table *database* dengan *migration* untuk table **m_kategori** yang sama-sama tidak memiliki *foreign key*

```
PS C:\laragon\www\PWL_POS> php artisan make:migration create_m_kategori_table --create=m_kategori
INFO Migration [C:\laragon\www\PWL_POS\database\Migrations\2024_03_10_051442_create_m_kategori_table.php] created success
fully.
PS C:\laragon\www\PWL_POS>
```

```
database > migrations > 2024_03_10_051442_create_m_kategori_table.php > ...
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12       public function up(): void
13       {
14           Schema::create('m_kategori', function (Blueprint $table) {
15               $table->id('kategori_id');
16               $table->string('kategori_kode', 10)->unique();
17               $table->string('kategori_nama', 100);
18               $table->timestamps();
19           });
20       }
21
22       /**
23        * Reverse the migrations.
24        */
25       public function down(): void
26       {
27           Schema::dropIfExists('m_kategori');
28       }
29   };
```

```
PS C:\laragon\www\PWL_POS> php artisan migrate
INFO Running migrations.
2024_03_10_051442_create_m_kategori_table ..... 57ms DONE
PS C:\laragon\www\PWL_POS>
```

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**

```
PS C:\laragon\www\FWL_POS> php artisan make:migration create_m_user_table --table=m_user

INFO Migration [C:\laragon\www\FWL_POS\database\migrations\2024_03_10_051918_create_m_user_table.php] created successfully.

PS C:\laragon\www\FWL_POS>
```

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

```
database > migrations > 2024_03_10_051918_create_m_user_table.php > class > down
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12     public function up(): void
13     {
14         Schema::create('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.

```
PS C:\laragon\www\FWL_POS> php artisan migrate

INFO Running migrations.

2024_03_10_051918_create_m_user_table ..... 135ms DONE

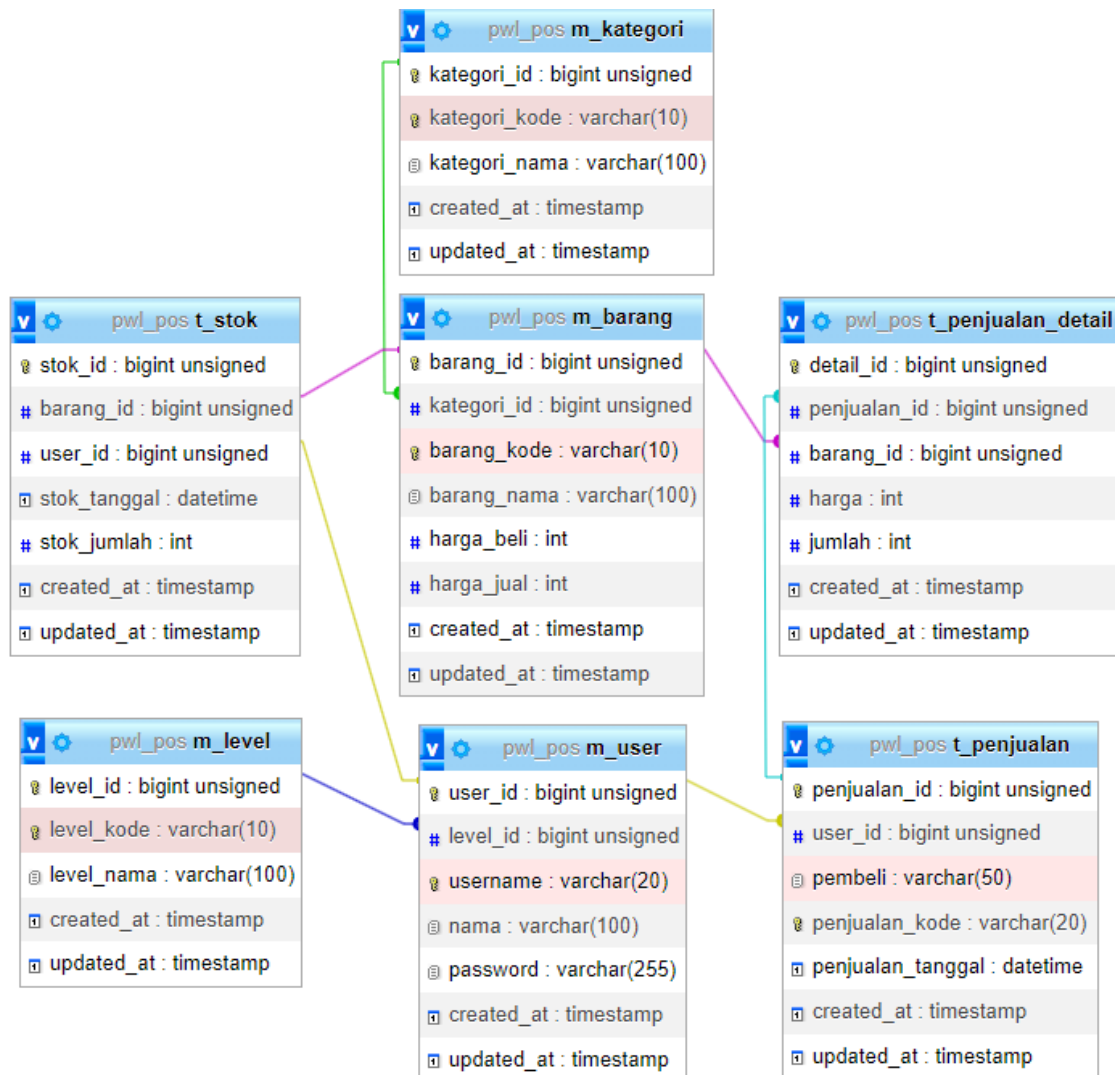
PS C:\laragon\www\FWL_POS>
```




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

m_barang
t_penjualan
t_stok
t_penjualan_detail

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



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



C. SEEDER

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

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

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

Perintah tersebut akan men-generate file seeder pada folder **PWL_POS/database/seeder**s

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

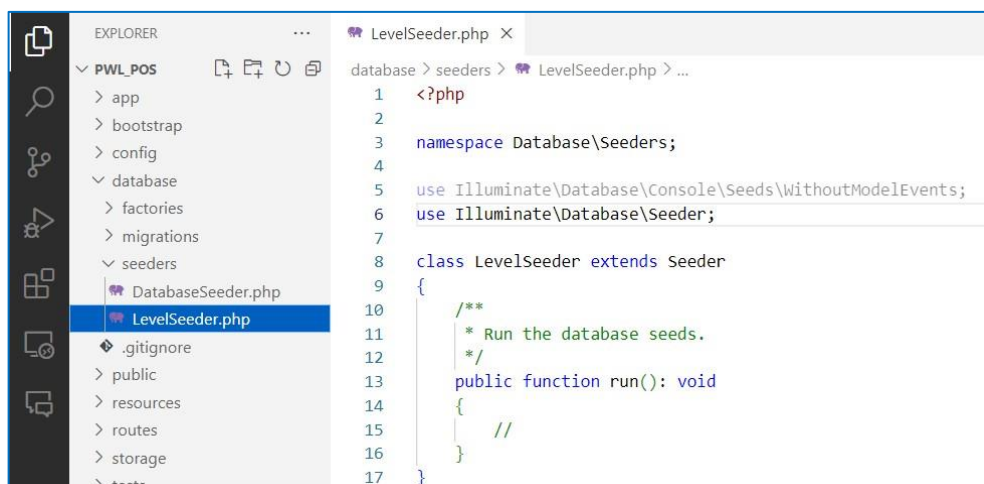
```
php artisan db:seed --class=<nama-class-seeder>
```

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

Praktikum 3 – Membuat file *seeder*

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

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





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

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

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

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

4. Ketika *seeder* berhasil dijalankan maka akan tampil data pada table `m_level`

<div><div><div>←</div><div>→</div></div></div>				level_id	level_kode	level_nama	created_at	updated_at
<div><div><div></div></div></div>	<div><div><div></div></div></div> Edit	<div><div><div></div></div></div> Copy	<div><div><div></div></div></div> Delete	1	ADM	Administrator	NULL	NULL
<div><div><div></div></div></div>	<div><div><div></div></div></div> Edit	<div><div><div></div></div></div> Copy	<div><div><div></div></div></div> Delete	2	MNG	Manager	NULL	NULL
<div><div><div></div></div></div>	<div><div><div></div></div></div> Edit	<div><div><div></div></div></div> Copy	<div><div><div></div></div></div> Delete	3	STF	Staff/Kasir	NULL	NULL

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

```
PS C:\laragon\www\PWL_POS> php artisan make:seeder UserSeeder
INFO Seeder [C:\laragon\www\PWL_POS\database\seeders\UserSeeder.php] created successfully.
PS C:\laragon\www\PWL_POS>
```



6. Modifikasi file `class UserSeeder` seperti berikut

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

7. Jalankan perintah untuk mengeksekusi class `UserSeeder`

```
PS C:\laragon\www\PWL_POS> php artisan db:seed --class=UserSeeder

INFO Seeding database.

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

8. Perhatikan hasil seeder pada table `m_user`












































































		user_id	level_id	username	nama	password	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	1	admin	Administrator	\$2y\$12\$Q3sMfxqwmPOB1V2sza2bQedvyzPJ0dwi1/YdRL8k5KY...	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	2	manager	Manager	\$2y\$12\$ShQBsq6Fhola/O0KLacyXler76u6xJ8lvEEFETw7Skil...	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	3	staff	Staff/Kasir	\$2y\$12\$Pqcbw4JJww/Q/8CwbGnnbuFIQfxLjhkgV72CH8HcY1P...	NULL	NULL

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



10. Sekarang coba kalian masukkan data *seeder* untuk table yang lain, dengan ketentuan seperti berikut

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

←T→				kategori_id	kategori_kode	kategori_nama	created_at	updated_at			
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	KAT001	Elektronik	NULL	NULL			
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	KAT002	Pakaian	NULL	NULL			
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	KAT003	Perabotan Rumah Tangga	NULL	NULL			
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	KAT004	Otomotif	NULL	NULL			
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	KAT005	Makanan	NULL	NULL			
←T→				barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	1	BRG001	Laptop Acer	5000000	6000000	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	2	BRG002	Kemeja Polos	150000	200000	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	3	BRG003	Meja Belajar Kayu	800000	1000000	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	4	BRG004	Ban Mobil Ring 17	500000	600000	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	5	BRG005	Beras 5kg	50000	60000	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	6	1	BRG006	Printer Canon	800000	1000000	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	7	2	BRG007	Celana Jeans	200000	250000	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	8	3	BRG008	Lemari Pakaian	1000000	1200000	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	9	4	BRG009	Oli Mobil	50000	60000	NULL	NULL
<input type="checkbox"/>	 Edit	 Copy	 Delete	10	5	BRG010	Susu Kental Manis	15000	20000	NULL	NULL
←T→				stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at	
<input type="checkbox"/>	 Edit	 Copy	 Delete	1	1	1	2024-03-10 00:00:00	100	NULL	NULL	
<input type="checkbox"/>	 Edit	 Copy	 Delete	2	2	1	2024-03-10 00:00:00	50	NULL	NULL	
<input type="checkbox"/>	 Edit	 Copy	 Delete	3	3	1	2024-03-10 00:00:00	200	NULL	NULL	
<input type="checkbox"/>	 Edit	 Copy	 Delete	4	4	1	2024-03-10 00:00:00	75	NULL	NULL	
<input type="checkbox"/>	 Edit	 Copy	 Delete	5	5	1	2024-03-10 00:00:00	120	NULL	NULL	
<input type="checkbox"/>	 Edit	 Copy	 Delete	6	6	1	2024-03-10 00:00:00	80	NULL	NULL	
<input type="checkbox"/>	 Edit	 Copy	 Delete	7	7	1	2024-03-10 00:00:00	60	NULL	NULL	
<input type="checkbox"/>	 Edit	 Copy	 Delete	8	8	1	2024-03-10 00:00:00	150	NULL	NULL	
<input type="checkbox"/>	 Edit	 Copy	 Delete	9	9	1	2024-03-10 00:00:00	90	NULL	NULL	
<input type="checkbox"/>	 Edit	 Copy	 Delete	10	10	1	2024-03-10 00:00:00	110	NULL	NULL	



←T→	penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	John Doe	PJN001	2024-03-10 08:00:00	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	1	Jane Doe	PJN002	2024-03-10 09:00:00	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	1	Michael Smith	PJN003	2024-03-10 10:00:00	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	1	Emma Johnson	PJN004	2024-03-10 11:00:00	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	1	David Williams	PJN005	2024-03-10 12:00:00	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	1	Sarah Brown	PJN006	2024-03-10 13:00:00	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	1	James Wilson	PJN007	2024-03-10 14:00:00	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	1	Olivia Taylor	PJN008	2024-03-10 15:00:00	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	9	1	Noah Anderson	PJN009	2024-03-10 16:00:00	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	10	1	Sophia Martinez	PJN010	2024-03-10 17:00:00	NULL	NULL

←T→	detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	1	1	1	10000	2	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	2	1	2	15000	3	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	3	2	3	12000	1	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	4	2	1	10000	2	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	5	3	2	15000	1	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	6	3	1	10000	3	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	7	4	3	12000	2	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	8	4	2	15000	1	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	9	5	1	10000	2	NULL	NULL
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	10	5	3	12000	1	NULL	NULL

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



D. DB FACADE

DB Façade merupakan fitur dari Laravel yang digunakan untuk melakukan *query* secara langsung dengan mengetikkan perintah SQL secara utuh (*raw query*). Disebut *raw query* (query mentah) karena penulisan query pada DB Façade langsung ditulis sebagaimana yang biasa dituliskan pada database, seperti “`select * from m_user`” atau “`insert into m_user...`” atau “`update m_user set ... Where ...`”

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

INFO

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

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

a. `DB::select()`

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

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

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

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

b. `DB::insert()`

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

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

c. `DB::update()`

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

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




d. `DB::delete()`

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

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

Praktikum 4 – Implementasi DB Facade

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

```
PS C:\laragon\www\FWL_POS> php artisan make:controller LevelController  
  
INFO Controller [C:\laragon\www\FWL_POS\app\Http\Controllers\LevelController.php] created successfully.  
PS C:\laragon\www\FWL_POS>
```

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`

```
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`

←T→	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

←T→	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	2024-03-12 05:03:59	NULL

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

```
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         $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['customer', 'CUS']);
16         return 'Update data berhasil. Jumlah data yang di update: '.$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`

←T→	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	2024-03-12 05:03:59	NULL

←T→	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	customer	2024-03-12 05:03:59	NULL



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

```
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         // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['customer', 'CUS']);
16         // return 'Update data berhasil. Jumlah data yang di update: '.$row.' baris';
17
18         $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
19         return 'Delete data berhasil. Jumlah data yang di hapus: '.$row.' baris';
20     }
21 }
```

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

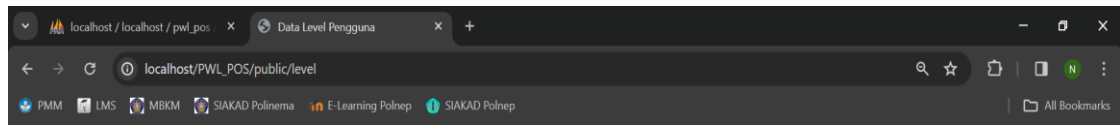
```
app > Http > Controllers > LevelController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class LevelController extends Controller
9  {
10     public function index()
11     {
12         // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13         // return 'Insert data baru berhasil';
14
15         // $row = DB::update('update m_level set level_nama = ? where level_kode = ?', ['customer', 'CUS']);
16         // return 'Update data berhasil. Jumlah data yang di update: '.$row.' baris';
17
18         // $row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);
19         // return 'Delete data berhasil. Jumlah data yang di hapus: '.$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 > html > body > table > tr > td
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <meta http-equiv="X-UA-Compatible" content="ie=edge">
7      <title>Data Level Pengguna</title>
8  </head>
9  <body>
10     <h1>Data Level Pengguna</h1>
11     <table border = "1" cellpadding = "2" cellspacing = "0">
12         <tr>
13             <th>ID</th>
14             <th>Kode Level</th>
15             <th>Nama Level</th>
16         </tr>
17         @foreach ($data as $d)
18             <tr>
19                 <td>{{ $d->level_id }}</td>
20                 <td>{{ $d->level_kode }}</td>
21                 <td>{{ $d->level_nama }}</td>
22             </tr>
23         @endforeach
24     </table>
25 </body>
26 </html>
```



10. Silahkan dicoba pada browser dan amati apa yang terjadi



Data Level Pengguna

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

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



E. QUERY BUILDER

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

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

INFO

Dokumentasi penggunaan Query Builder pada Laravel bisa dicek di laman ini

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

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

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

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

- 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 dahulu untuk mengelola data pada table `m_kategori`

```
PS C:\laragon\www\PWL_POS> php artisan make:controller KategoriController

INFO Controller [C:\laragon\www\PWL_POS\app\Http\Controllers\KategoriController.php] created successfully.

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

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\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`

```
app > Http > Controllers > KategoriController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController extends Controller
9  {
10     public function 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`

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KAT001	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KAT002	Pakaian	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KAT003	Perabotan Rumah Tangga	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KAT004	Otomotif	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KAT005	Makanan	NULL	NULL

		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KAT001	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KAT002	Pakaian	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KAT003	Perabotan Rumah Tangga	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KAT004	Otomotif	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KAT005	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	SNK	Snack/Makanan Ringan	2024-03-12 06:30:51	NULL



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

```
app > Http > Controllers > KategoriController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  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`

←T→		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KAT001	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KAT002	Pakaian	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KAT003	Perabotan Rumah Tangga	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KAT004	Otomotif	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KAT005	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	SNK	Snack/Makanan Ringan	2024-03-12 06:30:51	NULL

←T→		kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	Edit Copy Delete	1	KAT001	Elektronik	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	2	KAT002	Pakaian	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	3	KAT003	Perabotan Rumah Tangga	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	4	KAT004	Otomotif	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	5	KAT005	Makanan	NULL	NULL
<input type="checkbox"/>	Edit Copy Delete	6	SNK	Camilan	2024-03-12 06:30:51	NULL



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

```
app > Http > Controllers > KategoriController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController extends Controller
9  {
10     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 }
```

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

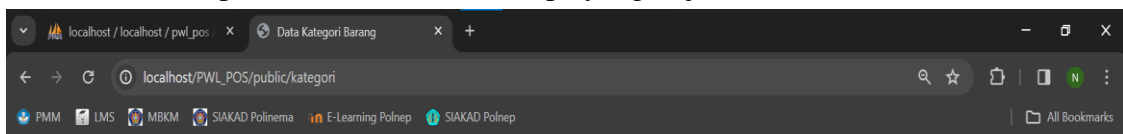
```
app > Http > Controllers > KategoriController.php > ...
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController extends Controller
9  {
10     public function index ()
11     {
12         /*$data = [
13             'kategori_kode' => "SNK",
14             'kategori_nama' => "Snack/Makanan Ringan",
15             'created_at' => now()
16         ];
17         DB::table('m_kategori')->insert($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         // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
24         // return 'Delete data berhasil. Jumlah data yang dihapus: '.$row.' baris';
25
26         $data = DB::table('m_kategori')->get();
27         return view('kategori', ['data' => $data]);
28     }
29 }
```



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

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



Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	KAT001	Elektronik
2	KAT002	Pakaian
3	KAT003	Perabotan Rumah Tangga
4	KAT004	Otomotif
5	KAT005	Makanan

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.

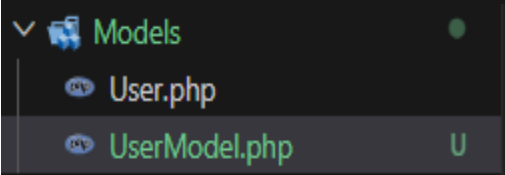
Praktikum 6 – Implementasi Eloquent ORM

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

```
PS C:\laragon\www\PWL_POS> php artisan make:model UserModel
```

INFO Model [C:\laragon\www\PWL_POS\app\Models\UserModel.php] created successfully.

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



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 > ...
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 }
```

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']);
```

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

```
PS C:\laragon\www\FWL_POS> php artisan make:controller UserController

INFO Controller [C:\laragon\www\FWL_POS\app\Http\Controllers\UserController.php] created successfully.

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

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

Data User

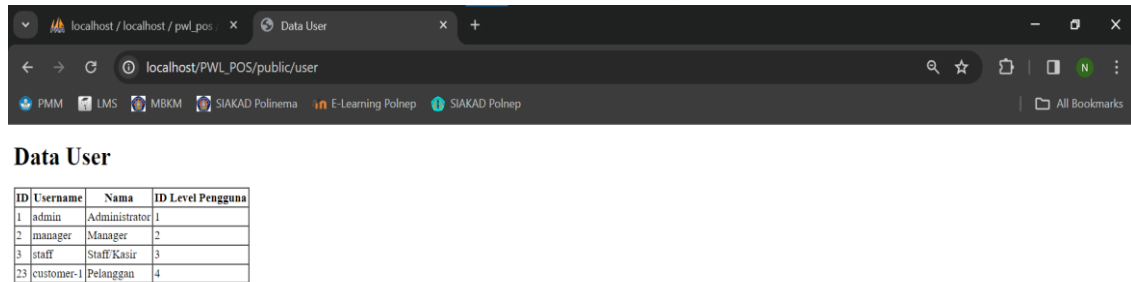
ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff Kasir	3

8. Setelah itu, kita modifikasi lagi file `UserController`

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



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

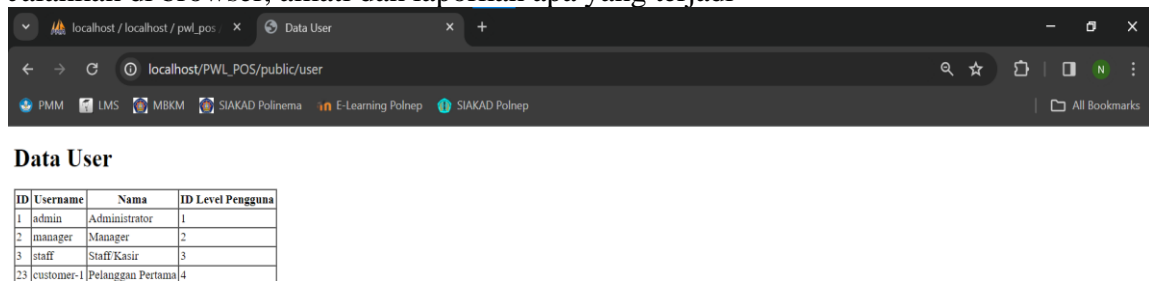


ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
23	customer-1	Pelanggan	4

10. Kita modifikasi lagi file `UserController` menjadi 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  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         $data = [
15             'nama' => 'Pelanggan Pertama',
16         ];
17         UserModel::where('username', 'customer-1')->update($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



ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
23	customer-1	Pelanggan Pertama	4

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

Link github : https://github.com/BucinBatagor/PWL_POS



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` dalam file `.env` pada Laravel adalah sebuah variabel yang menyimpan kunci rahasia yang digunakan untuk enkripsi data dan sesi dalam aplikasi Laravel.

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

Jawab: Untuk meng-*generate* nilai untuk `APP_KEY`, Anda dapat menggunakan perintah `artisan php artisan key:generate`. Perintah ini akan menghasilkan nilai acak untuk `APP_KEY` dan menyimpannya di file `.env` Anda.

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

Jawab: Secara default, Laravel memiliki dua file migrasi. Salah satunya adalah untuk membuat tabel pengguna (`users`) dan yang lainnya adalah untuk membuat password resets.

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

Jawab: Kode `$table->timestamps();` pada file migrasi akan menambahkan dua kolom ke tabel yang sedang dibuat, yaitu `created_at` dan `updated_at`. Kolom `created_at` akan merekam waktu pembuatan entri, sedangkan kolom `updated_at` akan merekam waktu ketika entri terakhir kali diubah.

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

Jawab: Fungsi `$table->id();` pada file migrasi akan membuat kolom dengan tipe data `UNSIGNED BIG INTEGER AUTO_INCREMENT PRIMARY KEY`. Ini biasanya digunakan sebagai primary key untuk tabel tersebut.

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

Jawab: Perbedaan antara menggunakan `$table->id();` dan `$table->id('level_id');` adalah bahwa pada yang pertama, Laravel akan menggunakan nama kolom default `id` untuk primary key, sedangkan pada yang kedua Anda bisa menentukan nama kolomnya sendiri, dalam hal ini `level_id`.

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

Jawab: Fungsi `->unique()` pada migration digunakan untuk menetapkan kolom tersebut sebagai unik, artinya tidak ada dua nilai yang sama yang diizinkan di 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: Pada tabel `m_user`, kolom `level_id` menggunakan `$table->unsignedBigInteger('level_id')` karena mengikuti konvensi penamaan Laravel untuk foreign key, sedangkan pada tabel `m_level`, kolom `level_id` menggunakan `$table->id('level_id')` karena merupakan primary key dari tabel tersebut.

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

Jawab : Class `Hash` pada Laravel digunakan untuk melakukan hash pada string, biasanya digunakan untuk mengenkripsi kata sandi pengguna. Kode program `Hash::make('1234')` akan menghasilkan hash dari string '1234', yang dapat digunakan untuk disimpan dalam database sebagai kata sandi terenkripsi.

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

Jawab: Tanda tanya (?) dalam query builder digunakan sebagai placeholder untuk parameter dalam prepared statement. Ini membantu mencegah serangan SQL injection dengan mengikuti praktik pengikatan parameter.

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

Jawab: Penulisan kode `protected $table = 'm_user';` digunakan untuk menentukan nama tabel yang terkait dengan model. Sedangkan `protected $primaryKey = 'user_id';` digunakan untuk menentukan nama primary key pada tabel tersebut.

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

Jawab: Penggunaan yang lebih mudah dalam melakukan operasi CRUD ke database antara DB Facade, Query Builder, dan Eloquent ORM dapat bervariasi tergantung pada kompleksitas aplikasi dan preferensi pengembang. Namun, secara umum, Eloquent ORM sering dianggap lebih mudah karena menyediakan model objek yang terstruktur dengan baik dan menangani banyak operasi database secara transparan. Query Builder memberikan fleksibilitas lebih besar dalam menulis kueri, sementara DB Facade lebih rendah tingkat abstraksinya dan biasanya digunakan untuk kueri sederhana.

*** Sekian, dan selamat belajar ***