LAPORAN PRAKTIKUM

JOBSHEET III : MIGRATION, SEEDER, DB FAÇADE, QUERY BUILDER, DAN ELOQUENT FORM

MATA KULIAH PEMROGRAMAN WEB LANJUT

Dosen Pengampu : Dimas Wahyu Wibowo, S.T., M.T.

Disusun oleh :

Nama : Vita Eka Saraswati

NIM : 2341760082

No. Absen : 29

JURUSAN TEKNOLOGI INFORMASI

PROGRAM STUDI D-IV SISTEM INFORMASI BISNIS

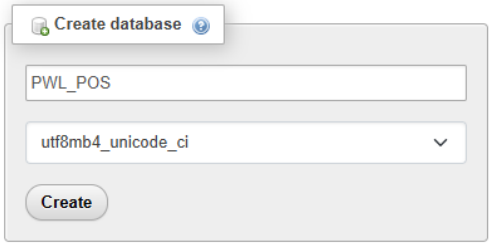
POLITEKNIK NEGERI MALANG

2025

A. PENGATURAN DATABASE

Praktikum 1

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



2. Buka aplikasi VSCode dan buka folder project PWL_POS yang sudah kita buat
3. Copy file .env.example menjadi .env
4. Buka file .env, dan pastikan konfigurasi **APP_KEY** bernilai. Jika belum bernilai silahkan kalian *generate* menggunakan php artisan.

```
.env
1  APP_NAME=Laravel
2  APP_ENV=local
3  APP_KEY=base64:dxYUQu0F6ITUIL1fP7ir0A5jM453ILk/314e5P5/m
```

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

```
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=
```

B. MIGRATION

Praktikum 2.1

1. Buka *terminal* VSCode kalian, untuk yang di kotak merah adalah default dari Laravel
2. Buat file migrasi untuk table m_level dengan perintah

```
PS F:\laragon\www\PWL_POS> php artisan make:migration create_m_level_table --create=m_level

  INFO  Migration [F:\laragon\www\PWL_POS\database\migrations/2025_03_05_073556_create_m_level_table.php] cre
ated successfully.
```

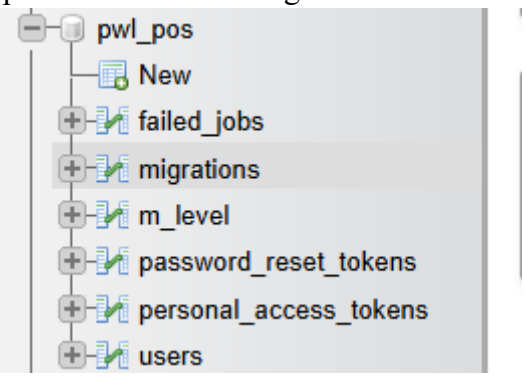
3. Perhatikan bagian yang di kotak merah, bagian tersebut yang akan kita modifikasi sesuai desain database yang sudah ada

```
public function up(): void
{
    Schema::create(table: 'm_level', callback: function (Blueprint $table): void {
        $table->id(column: 'level_id');
        $table->string(column: 'level_kode', length: 10)->unique();
        $table->string(column: 'level_nama', length: 100);
        $table->timestamps();
    });
}
```

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

```
PS F:\laragon\www\PWL_POS> php artisan migrate

  INFO  Preparing database.

  Creating migration table ..................................................... 769ms DONE

  INFO  Running migrations.

  2014_10_12_000000_create_users_table ......................................... 540ms DONE
  2014_10_12_100000_create_password_reset_tokens_table ......................... 79ms DONE
  2019_08_19_000000_create_failed_jobs_table ................................... 372ms DONE
  2019_12_14_000001_create_personal_access_tokens_table ........................ 301ms DONE
  2025_03_05_073556_create_m_level_table ....................................... 254ms DONE
```

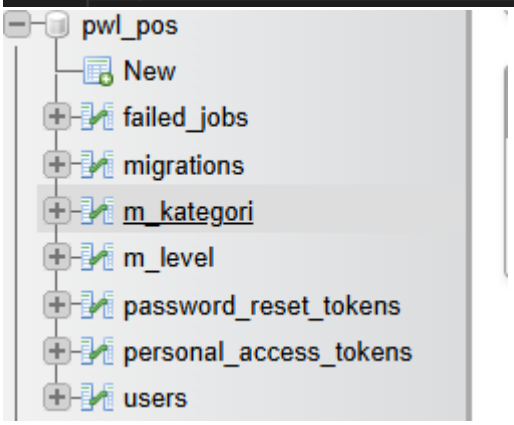
5. Cek di phpMyAdmin apakah table sudah ter-generate



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

```
PS F:\laragon\www\PWL_POS> php artisan make:migration create_m_kategori_table --create=m_kategori
 * History restored

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

  INFO  Running migrations.

  2025_03_05_074929_create_m_kategori_table ..................................... 110ms DONE
```

```
14        Schema::create(table: 'm_level', callback: function (Blueprint $table): void {
15            $table->id(column: 'kategori_id');
16            $table->string(column: 'kategori_kode', length: 10)->unique();
17            $table->string(column: 'kategori_nama', length: 100);
18            $table->timestamps();
19        });
```



Praktikum 2.2

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

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

  INFO  Migration [F:\laragon\www\PWL_POS\database\migrations/2025_03_05_075648_create_m_user_table.php
] created successfully.
```

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

```php
Schema::create(table: 'm_user', callback: function (Blueprint $table): void {
    $table->id(column: 'user_id');
    $table->unsignedBigInteger(column: 'level_id')->index(); //indexing untuk ForeignKey
    $table->string(column: 'username', length: 20)->unique(); //unique untuk memastikan tidak ada username yang sama
    $table->string(column: 'nama', length: 100);
    $table->string(column: 'password');
    $table->timestamps();

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

```php
    public function down(): void
    {
        Schema::table(table: 'm_user', callback: function (Blueprint $table): void {
            Schema::dropIfExists(table: 'm_user');
        });
    }
};

```

```
PS F:\laragon\www\PWL_POS> php artisan migrate

  INFO  Running migrations.

  2025_03_05_075648_create_m_user_table ..... 495ms DONE
```

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

m_barang
t_penjualan
t_stok
t_penjualan_detail

- Tabel m_barang

```
PS F:\laragon\www\PWL_POS> php artisan make:migration create_m_barang_table -
-table=m_barang

533_create_m_barang_table.php] created successfully.
```

```php
Schema::create(table: 'm_barang', callback: function (Blueprint $table): void {
    $table->id(column: 'barang_id');
    $table->unsignedBigInteger(column: 'kategori_id')->index(); // indexing untuk ForeignKey
    $table->string(column: 'barang_kode', length: 10)->unique();
    $table->string(column: 'barang_nama', length: 100);
    $table->integer(column: 'harga_beli');
    $table->integer(column: 'harga_jual');
    $table->timestamps();

    $table->foreign(columns: 'kategori_id')->references(columns: 'kategori_id')->on(table: 'm_kategori');
});
```

```php
public function down(): void
{
    Schema::table(table: 'm_barang', callback: function (Blueprint $table): void {
        Schema::dropIfExists(table: 'm_barang');
    });
}
```

2025_03_05_095021_create_m_barang_table.php

database > migrations > 2025_03_05_095021_create_m_barang_table.php > class

```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 'm_barang', callback: function (Blueprint $table): void {
            $table->unsignedBigInteger(column: 'barang_id')->autoIncrement();
            $table->unsignedBigInteger(column: 'kategori_id')->index();
            $table->string(column: 'barang_kode', length: 20)->unique();
            $table->string(column: 'barang_nama', length: 100); // Pastikan kolom ini ada
            $table->integer(column: 'harga_beli', autoIncrement: false, unsigned: true); // harga beli dengan tipe integer
            $table->integer(column: 'harga_jual', autoIncrement: false, unsigned: true); // harga jual dengan tipe integer
            $table->timestamps();

            $table->foreign(columns: 'kategori_id')->references(columns: 'kategori_id')->on(table: 'm_kategori');
        });
    }

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

- Buat migrations/tabel t_penjualan

```
  INFO  Migration [F:\laragon\www\PWL_POS\database\migrations/2025_03_05_095356_create_t_penjualan
__table.php] created successfully.

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

  INFO  Running migrations.

  2025_03_05_095356_create_t_penjualan__table ........................................ 521ms DONE
```

2025_03_05_095356_create_t_penjualan__table.php

database > migrations > 2025_03_05_095356_create_t_penjualan__table.php > ...

```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 't_penjualan', callback: function (Blueprint $table): void {
            $table->unsignedBigInteger(column: 'penjualan_id')->autoIncrement();
            $table->unsignedBigInteger(column: 'user_id')->index();
            $table->unsignedBigInteger(column: 'barang_id')->index(); // Foreign key to m_barang
            $table->string(column: 'pembeli', length: 50);
            $table->string(column: 'penjualan_kode', length: 20);
            $table->date(column: 'penjualan_tanggal');
            $table->timestamps();

            $table->foreign(columns: 'user_id')->references(columns: 'user_id')->on(table: 'm_user'); //Define foreign key
        });
    }

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

- Buat migrations untuk t_stok

```
PS F:\laragon\www\PWL_POS> php artisan make:migration create_t_stok__ta
ble --table=t_stok

  INFO  Migration [F:\laragon\www\PWL_POS\database\migrations/2025_03_
05_095703_create_t_stok__table.php] created successfully.
```

```
PS F:\laragon\www\PWL_POS> php artisan migrate

  INFO  Running migrations.

  2025_03_05_095703_create_t_stok__table ..................................... 635ms DONE
```

2025_03_05_095703_create_t_stok__table.php

database > migrations > 2025_03_05_095703_create_t_stok__table.php > ...

```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 't_stok', callback: function (Blueprint $table): void {
            $table->unsignedBigInteger(column: 'stok_id')->autoIncrement();
            $table->unsignedBigInteger(column: 'barang_id')->index();
            $table->unsignedBigInteger(column: 'user_id')->index(); // Foreign key to m_user
            $table->integer(column: 'jumlah');
            $table->timestamps();

            $table->foreign(columns: 'barang_id')->references(columns: 'barang_id')->on(table: 'm_barang');
            $table->foreign(columns: 'user_id')->references(columns: 'user_id')->on(table: 'm_user'); // Define foreign key
        });
    }

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

- Buat migrations untuk t_penjualan_detail

```
PS F:\laragon\www\PWL_POS> php artisan make:migration create_t_penjualan_detail_table --table=t_p
enjualan_detail

  INFO  Migration [F:\laragon\www\PWL_POS\database\migrations/2025_03_05_100226_create_t_penjual
an_detail_table.php] created successfully.
```

2025_03_05_095703_create_t_stok__table.php | 2025_03_05_100226_create_t_penjualan_detail_table.php

database > migrations > 2025_03_05_100226_create_t_penjualan_detail_table.php > class > up

```php
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create(table: 't_penjualan_detail', callback: function (Blueprint $table): void {
            $table->unsignedBigInteger(column: 'detail_id')->autoIncrement();
            $table->unsignedBigInteger(column: 'penjualan_id')->index();
            $table->unsignedBigInteger(column: 'barang_id')->index();
            $table->integer(column: 'jumlah');
            $table->integer(column: 'harga');
            $table->timestamps();

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

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

```
PS F:\laragon\www\PWL_POS> php artisan migrate

  INFO  Running migrations.

  2025_03_05_100226_create_t_penjualan_detail_table ........................ 565ms DONE
```

Tampilan tabel – tabel pada database PWL_POS

Structure | SQL | Search | Query | Export | Import | Operations | Privileges | Routines | Events | Triggers

Table	Action	Rows	Type	Collation	Size	Overhead
barang	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
migrations	Browse Structure Search Insert Empty Drop	13	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
m_barang	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
m_kategori	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
m_level	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
m_user	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
password_reset_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
personal_access_tokens	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
t_penjualan	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
t_penjualan_detail	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
t_stok	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
users	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
13 tables	Sum	13	InnoDB	utf8mb4_unicode_ci	320.0 KiB	0 B

C. SEEDER

Praktikum 3 – Membuat file seeder

1. Membuat file seeder untuk table m_level dengan mengetikkan perintah

```
PS F:\laragon\www\PWL_POS> php artisan make:seeder LevelSeeder

  INFO  Seeder [F:\laragon\www\PWL_POS\database\seeders\LevelSeeder.php] creat
ed successfully.
```

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

```
LevelSeeder.php ×
database > seeders > LevelSeeder.php > PHP > LevelSeeder
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
   0 references | 0 implementations
9  class LevelSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
       0 references | 0 overrides
14     public function run(): void
15     {
16         DB::table(table: 'm_level')->insert(values: [
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' => 'Staf/Kasir'],
20
21         ]);
22     }
23 }
```

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

```
PS F:\laragon\www\PWL_POS> php artisan db:seed --class=LevelSeede
r

  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
1	ADM	Administrator	NULL	NULL
2	MNG	Manager	NULL	NULL
3	STF	Staf/Kasir	NULL	NULL

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

```
PS F:\laragon\www\PWL_POS> php artisan make:seeder UserSeeder

  INFO  Seeder [F:\laragon\www\PWL_POS\database\seeders\UserSeeder.php] created successfully.
```

6. Modifikasi file class UserSeeder seperti berikut

```php
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\DB;

0 references | 0 implementations
class UserSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $data = [
            [
                'user_id' => 1,
                'level_id' => 1,
                'username' => 'admin',
                'nama' => 'Administrator',
                'password' => Hash::make(value: '12345'), // class untuk mengenkripsi/hash password
            ],
            [
                'user_id' => 2,
                'level_id' => 2,
                'username' => 'manager',
                'nama' => 'Manager',
                'password' => Hash::make(value: '12345'),
            ],
            [
                'user_id' => 3,
                'level_id' => 3,
                'username' => 'staff',
                'nama' => 'Staff/Kasir',
                'password' => Hash::make(value: '12345'),
            ],
        ];

        DB::table(table: 'm_user')->insert(values: $data);
    }
```

7. Jalankan perintah untuk mengeksekusi class UserSeeder

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

  INFO  Seeding database.
```

8. Perhatikan hasil seeder pada table m_user

			user_id	level_id	username	nama	password
	Edit	Copy	Delete	1	1	admin	Administrator	$2y$12$hi6jFJzD4/
	Edit	Copy	Delete	2	2	manager	Manager	$2y$12$iJt6fdWAJt
	Edit	Copy	Delete	3	3	staff	Staff/Kasir	$2y$12$Wu2LPoU

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

No	Nama Tabel	Jumlah Data	Keterangan
1	m_kategori	5	5 kategori barang
2	m_barang	10	10 barang yang berbeda
3	t_stok	10	Stok untuk 10 barang
4	t_penjualan	10	10 transaksi penjualan
5	t_penjualan_detail	30	3 barang untuk setiap transaksi penjualan

- **KategoriSeeder**

```
PS F:\laragon\www\PWL_POS> php artisan make:seeder KategoriSeeder

  INFO  Seeder [F:\laragon\www\PWL_POS\database\seeders\KategoriSeeder.php] crea
ted successfully.
```

KategoriSeeder.php

database > seeders > KategoriSeeder.php > PHP > KategoriSeeder

```php
<?php

namespace Database\Seeders;

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

0 references | 0 implementations
class KategoriSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        DB::table(table: 'm_kategori')->insert(values: [
            ['kategori_id' => 1, 'kategori_kode' => 'KTG001', 'kategori_nama' => 'Elektronik'],
            ['kategori_id' => 2, 'kategori_kode' => 'KTG002', 'kategori_nama' => 'Pakaian'],
            ['kategori_id' => 3, 'kategori_kode' => 'KTG003', 'kategori_nama' => 'Makanan'],
            ['kategori_id' => 4, 'kategori_kode' => 'KTG004', 'kategori_nama' => 'Minuman'],
            ['kategori_id' => 5, 'kategori_kode' => 'KTG005', 'kategori_nama' => 'Obat-obatan'],
        ]);
    }
}
```

```
PS F:\laragon\www\PWL_POS> php artisan db:seed --class=KategoriSeeder

  INFO  Seeding database.
```

			kategori_id	kategori_kode	kategori_nama	created_at	update
	Edit	Copy	Delete	1	KTG001	Elektronik	NULL	NULL
	Edit	Copy	Delete	2	KTG002	Pakaian	NULL	NULL
	Edit	Copy	Delete	3	KTG003	Makanan	NULL	NULL
	Edit	Copy	Delete	4	KTG004	Minuman	NULL	NULL
	Edit	Copy	Delete	5	KTG005	Obat-obatan	NULL	NULL

- **BarangSeeder**

```
PS F:\laragon\www\PWL_POS> php artisan make:seeder BarangSeeder

  INFO  Seeder [F:\laragon\www\PWL_POS\database\seeders\BarangSeeder.php] c
reated successfully.
```

Activate Windows
Go to Settings to activate Windows.

BarangSeeder.php

database > seeders > BarangSeeder.php > PHP Intelephense > BarangSeeder > run

```php
<?php

namespace Database\Seeders;

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

0 references | 0 implementations
class BarangSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        DB::table(table: 'm_barang')->insert(values: [
            [
                'barang_id' => 1,
                'kategori_id' => 1,
                'barang_kode' => 'BRG001',
                'barang_nama' => 'Samsung Smart TV',
                'harga_beli' => 3000000,
                'harga_jual' => 3500000
            ],
```

```
PS F:\laragon\www\PWL_POS> php artisan db:seed --class=BarangSeeder

  INFO  Seeding database.
```

	barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	up
Edit Copy Delete	1	1	BRG001	Samsung Smart TV	3000000	3500000	NULL	Nu
Edit Copy Delete	2	1	BRG002	Air Conditioner Sharp	4000000	4200000	NULL	Nu
Edit Copy Delete	3	1	BRG003	Hair Dryer Dyson	4000000	4500000	NULL	Nu
Edit Copy Delete	4	2	BRG004	Cardigan Rajut	120000	125000	NULL	Nu
Edit Copy Delete	5	2	BRG005	Jaket Polos	150000	155000	NULL	Nu

- **StokSeeder**

```
PS F:\laragon\www\PWL_POS> php artisan make:seeder StokSeeder

  INFO  Seeder [F:\laragon\www\PWL_POS\database\seeders\StokSeeder.php] created successfully.
```

database > seeders > StokSeeder.php > ...

```php
<?php

namespace Database\Seeders;

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

0 references | 0 implementations
class StokSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $stok = [];

        for ($i = 1; $i <= 15; $i++) {
            $stok[] = [
                'stok_id' => $i,
                'barang_id' => $i, // Pastikan barang_id ada di m_barang
                'user_id' => rand(min: 1, max: 3),
                'stok_tanggal' => Carbon::now()->subDays(value: rand(min: 1, max: 30))->toDateTimeString(),
                'jumlah' => rand(min: 1, max: 50),
            ];
        }

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

```
PS F:\laragon\www\PWL_POS> php artisan db:seed --class=StokSeeder

  INFO  Seeding database.
```

	stok_id	barang_id	user_id	jumlah	stok_tanggal	created_at	updated_at
Edit Copy Delete	1	1	2	33	2025-02-05 14:15:17	NULL	NULL
Edit Copy Delete	2	2	1	45	2025-02-26 14:15:17	NULL	NULL
Edit Copy Delete	3	3	1	26	2025-02-03 14:15:17	NULL	NULL
Edit Copy Delete	4	4	1	26	2025-02-19 14:15:17	NULL	NULL
Edit Copy Delete	5	5	1	46	2025-02-18 14:15:17	NULL	NULL
Edit Copy Delete	6	6	1	26	2025-03-03 14:15:17	NULL	NULL

- **Seeder Penjualan**

```
PS F:\laragon\www\PWL_POS> php artisan make:seeder PenjualanSeeder

  INFO  Seeder [F:\laragon\www\PWL_POS\database\seeders\PenjualanSeeder.php] created successfully.
```

PenjualanSeeder.php

database > seeders > PenjualanSeeder.php > ...

```php
use Illuminate\Support\Facades\DB;

0 references | 0 implementations
class PenjualanSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $faker = Factory::create();
        $penjualan = [];

        for ($i = 1; $i <= 15; $i++) {
            $penjualan[] = [
                'penjualan_id' => $i,
                'pembeli' => $faker->unique()->name(),
                'penjualan_kode' => $faker->unique()->word(),
                'penjualan_tanggal' => Carbon::now()->subDays(value: rand(min: 1, max: 30))->toDateTimeString(),
                'user_id' => rand(min: 1, max: 3),
                'barang_id' => rand(min: 1, max: 10), // Pastikan barang_id ada dalam tabel m_barang
            ];
        }

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

```
PS F:\laragon\www\PWL_POS> php artisan db:seed --class=PenjualanSeeder

  INFO  Seeding database.
```

	penjualan_id	user_id	barang_id	pembeli	penjualan_kode	penjualan_tanggal	created
Edit Copy Delete	1	1	10	Mr. Zechariah Roob PhD	voluptate	2025-02-08	NULL
Edit Copy Delete	2	3	6	Concepcion Jerde V	ex	2025-02-24	NULL
Edit Copy Delete	3	3	9	Brendan Hahn	laudantium	2025-02-22	NULL
Edit Copy Delete	4	2	10	Anabel McCullough	qui	2025-02-23	NULL

- **Seeder PenjualanDetail**

```
PS F:\laragon\www\PWL_POS> php artisan make:seeder PenjualanDetailSeeder

  INFO  Seeder [F:\laragon\www\PWL_POS\database\seeders\PenjualanDetailSeeder.php] created successfully.
```

Open file in editor (ctrl + click)

database > seeders > PenjualanDetailSeeder.php > PHP > PenjualanDetailSeeder

```php
<?php

namespace Database\Seeders;

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

0 references | 0 implementations
class PenjualanDetailSeeder extends Seeder
{
    0 references | 0 overrides
    public function run(): void
    {
        $data = [];
        for ($i = 1; $i <= 10; $i++) {
            for ($j = 1; $j <= 3; $j++) {
                $data[] = [
                    'penjualan_id' => $i,
                    'barang_id' => $i,
                    'harga' => random_int(min: 10000, max: 50000),
                    'jumlah' => random_int(min: 1, max: 10),
                ];
            }
        }

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

```
PS F:\laragon\www\PWL_POS> php artisan db:seed --class=PenjualanDetailSeeder

  INFO  Seeding database.
```

←T→				detail_id	penjualan_id	barang_id	jumlah	harga	created_at	updated_at
Edit	Copy	Delete		1	1	1	5	11726	NULL	NULL
Edit	Copy	Delete		2	1	1	7	30570	NULL	NULL
Edit	Copy	Delete		3	1	1	1	42966	NULL	NULL
Edit	Copy	Delete		4	2	2	4	24813	NULL	NULL
Edit	Copy	Delete		5	2	2	3	26249	NULL	NULL

D. DB FAÇADE

Praktikum 4 - Implementasi DB Facade

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

```
PS F:\laragon\www\PWL_POS> php artisan make:controller
LevelController

  INFO  Controller [F:\laragon\www\PWL_POS\app\Http\Co
ntrollers\LevelController.php] created successfully.
```

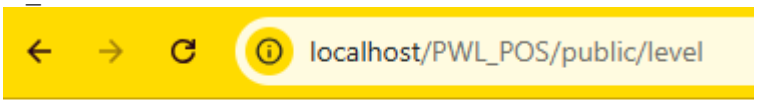
2. Kita modifikasi dulu untuk routing-nya, ada di PWL_POS/routes/web.php

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

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

```php
<?php

namespace App\Http\Controllers;

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

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

        return "Insert data baru berhasil";
    }
}
```

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

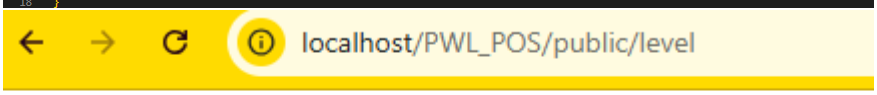


Insert data baru berhasil

	level_id	level_kode	level_nama	created_at	updated_at
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staf/Kasir	NULL	NULL
Edit Copy Delete	4	CUS	Pelanggan	2025-03-05 14:38:42	NULL

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

```php
<?php

namespace App\Http\Controllers;

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

2 references | 0 implementations
class LevelController extends Controller
{
    1 reference | 0 overrides
    public function index(): string
    {
        // DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
        // return "Insert data baru berhasil";

        $row = DB::update(query: 'update m_level set level_nama = ? where level_kode = ?', bindings: ['Customer', 'CUS']);
        return "update data berhasil. Jumlah data yang diupdate: ".$row." baris";
    }
}
```



update data berhasil. Jumlah data yang diupdate: 1 baris

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

	level_id	level_kode	level_nama	created_at	updated_at
Edit Copy Delete	1	ADM	Administrator	NULL	NULL
Edit Copy Delete	2	MNG	Manager	NULL	NULL
Edit Copy Delete	3	STF	Staf/Kasir	NULL	NULL
Edit Copy Delete	4	CUS	Customer	2025-03-05 14:38:42	NULL

Data dengan level kode 4 yang semula level_namanya Pelanggan saat ini menjadi Customer

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

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

localhost/PWL_POS/public/level

Delete data berhasil. Jumlah data yang dihapus: 1 baris

←T→			level_id	level_kode	level_nama	created_at	updated_at
Edit	Copy	Delete	1	ADM	Administrator	NULL	NULL
Edit	Copy	Delete	2	MNG	Manager	NULL	NULL
Edit	Copy	Delete	3	STF	Staf/Kasir	NULL	NULL

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

```
web.php    LevelController.php ×
app > Http > Controllers > LevelController.php > PHP > LevelController
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
   2 references | 0 implementations
8  class LevelController extends Controller
9  {
       1 reference | 0 overrides
10     public function index(): Factory|View
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(query: 'select * from m_level');
22         return view(view: 'level', data: ['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

```
<!DOCTYPE html>
<html>
<head>
    <title>Data Level Pengguna</title>
</head>
<body>
    <h1>Data Level Pengguna</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <th>ID</th>
            <th>Kode Level</th>
            <th>Nama Level</th>
        </tr>
        @foreach ($data as $d)
        <tr>
            <td>{{ $d->level_id }}</td>
            <td>{{ $d->level_kode }}</td>
            <td>{{ $d->level_nama }}</td>
        </tr>
        @endforeach
    </table>
</body>
</html>
```

10. Silahkan dicoba pada browser dan amati apa yang terjadi

localhost/PWL_POS/public/level

Data Level Pengguna

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

E. QUERY BUILDER

Praktikum 5 - Implementasi Quey Builder

1. Kita buat controller dahuku untuk mengelola data pada table m_kategori

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

  INFO  Controller [F:\laragon\www\PWL_POS\app\Http\Controllers\KategoriController.php] created succe
ssfully.
```

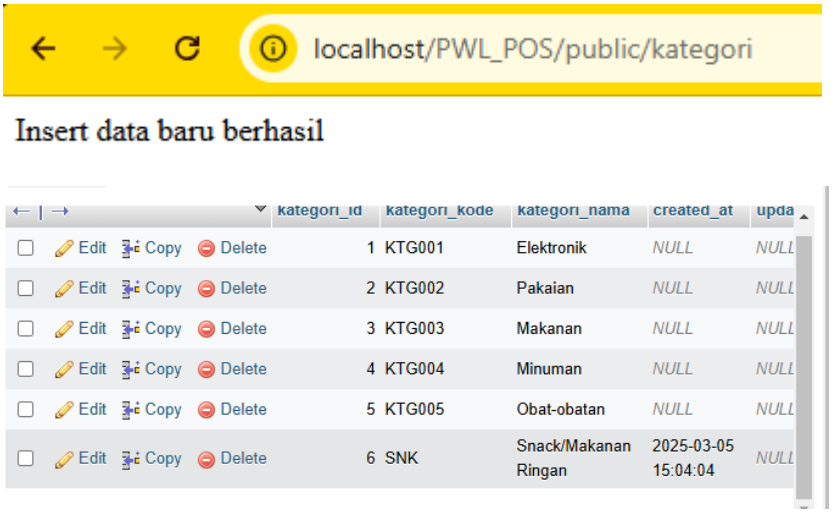
2. Kita modifikasi dulu untuk routing-nya, ada di PWL_POS/routes/web.php

```
web.php ×
routes > web.php > ...
1  <?php
2  use App\Http\Controllers\KategoriController;
3  use App\Http\Controllers\LevelController;
4  use Illuminate\Support\Facades\Route;
5
6  /*
7  |--------------------------------------------------------------------------
8  | Web Routes
9  |--------------------------------------------------------------------------
10 |
11 | Here is where you can register web routes for your application. These
12 | routes are loaded by the RouteServiceProvider and all of them will
13 | be assigned to the "web" middleware group. Make something great!
14 |
15 */
16
17 Route::get(uri: '/', action: function (): Factory|View {
18     return view(view: 'welcome');
19 });
20
21 Route::get(uri: '/level', action: [LevelController::class, 'index']);
22 Route::get(uri: '/kategori', action: [KategoriController::class, 'index']);
```

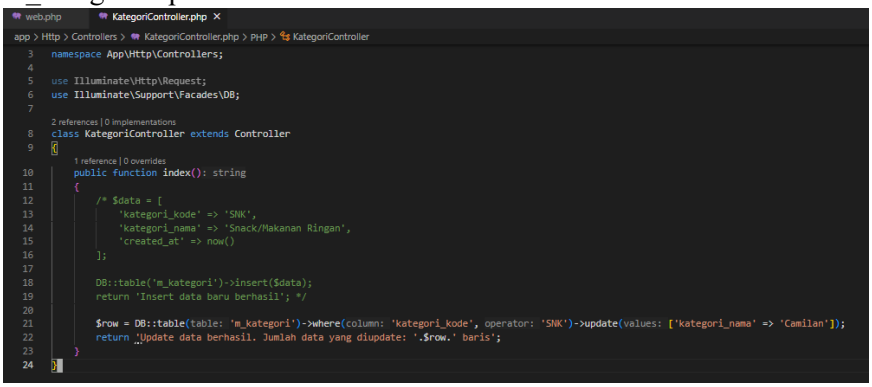
3. Selanjutnya, kita modifikasi file KategoriController untuk menambahkan 1 data ke table m_kategori

```
web.php    KategoriController.php ×
app > Http > Controllers > KategoriController.php > PHP > KategoriController
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
   2 references | 0 implementations
8  class KategoriController extends Controller
9  {
       1 reference | 0 overrides
10     public function index(): string
11     {
12         $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17
18         DB::table(table: 'm_kategori')->insert(values: $data);
19         return 'Insert data baru berhasil';
20     }
21 }
```

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



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



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

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

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

localhost/PWL_POS/public/kategori

Delete data berhasil. Jumlah data yang dihapus: 1 baris

←T→				kategori_id	kategori_kode	kategori_nama	created_at	upda
☐	Edit	Copy	Delete	1	KTG001	Elektronik	NULL	NULL
☐	Edit	Copy	Delete	2	KTG002	Pakaian	NULL	NULL
☐	Edit	Copy	Delete	3	KTG003	Makanan	NULL	NULL
☐	Edit	Copy	Delete	4	KTG004	Minuman	NULL	NULL
☐	Edit	Copy	Delete	5	KTG005	Obat-obatan	NULL	NULL

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

```
17
18         DB::table('m_kategori')->insert($data);
19         return 'Insert data baru berhasil'; */
20
21     // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kat
22     // return 'Update data berhasil. Jumlah data yang diupdate: '.$row.' baris';
23
24     // $row DB::table('m_kategori')->where('kategori_kode', 'SNK')->delete();
25     // return 'Delete data berhasil. Jumlah data yang dihapus: $row baris';
26
27     $data = DB::table(table: 'm_kategori')->get();
28     return view(view: 'kategori', data: ['data' => $data]);
29     }
30 }
```

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
<!DOCTYPE html>
<html>
<head>
    <title>Data Kategori Barang</title>
</head>
<body>
    <h1>Data Kategori Barang</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <th>ID</th>
            <th>Kode Kategori</th>
            <th>Nama Kategori</th>
        </tr>
        @foreach ($data as $d)
        <tr>
            <td>{{ $d->kategori_id }}</td>
            <td>{{ $d->kategori_kode }}</td>
            <td>{{ $d->kategori_nama }}</td>
        </tr>
        @endforeach
    </table>
</body>
```

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

localhost/PWL_POS/public/kategori

Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	KTG001	Elektronik
2	KTG002	Pakaian
3	KTG003	Makanan
4	KTG004	Minuman
5	KTG005	Obat-obatan

F. ELOQUENT ORM

Praktikum 5 - Implementasi Eloqunt ORM

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

```
PS F:\laragon\www\PWL_POS> php artisan make:model UserMod
el

  INFO  Model [F:\laragon\www\PWL_POS\app\Models\UserMod
el.php] created successfully.
```

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

```
web.php    UserModel.php ×
app > Models > UserModel.php > PHP > UserModel
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
   0 references | 0 implementations
8  class UserModel extends Model
9  {
10     use HasFactory;
11
       0 references
12     protected $table = 'm_user';       // Mendefinisikan nama tabel yang digunakan oleh model ini
       0 references
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  |--------------------------------------------------------------------------
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(uri: '/', action: function (): Factory|View {
20     return view(view: 'welcome');
21 });
22
23 Route::get(uri: '/level', action: [LevelController::class, 'index']);
24 Route::get(uri: '/kategori', action: [KategoriController::class, 'index']);
25 Route::get(uri: '/user', action: [UserController::class, 'index']);
```

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

```
PS F:\laragon\www\PWL_POS> php artisan make:controller UserController

  INFO  Controller [F:\laragon\www\PWL_POS\app\Http\Controllers\UserController.p
hp] created successfully.
```

app > Http > Controllers > UserController.php > PHP > UserController

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;

2 references | 0 implementations
class UserController extends Controller
{
    1 reference | 0 overrides
    public function index(): Factory|View
    {
        // coba akses model UserModel
        $user = UserModel::all(); // ambil semua data dari tabel m_user
        return view(view: 'user', data: ['data' => $user]);
    }
}
```

6. Kemudian kita buat view user.blade.php

web.php | user.blade.php | UserModel.php

resources > views > user.blade.php

```blade
<!DOCTYPE html>
<html>
<head>
    <title>Data User</title>
</head>
<body>
    <h1>Data User</h1>
    <table border="1" cellpadding="2" cellspacing="0">
        <tr>
            <th>ID</th>
            <th>Username</th>
            <th>Nama</th>
            <th>ID Level Pengguna</th>
        </tr>
        @foreach ($data as $d)
        <tr>
            <td>{{ $d->user_id }}</td>
            <td>{{ $d->username }}</td>
            <td>{{ $d->nama }}</td>
            <td>{{ $d->level_id }}</td>
        </tr>
        @endforeach
    </table>
</body>
</html>
```

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

127.0.0.1:8000/user

Data User

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

8. Setelah itu, kita modifikasi lagi file UserController

```php
<?php

namespace App\Http\Controllers;

use Illuminate\HTTP\Request;
use App\Models\UserModel;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;

2 references | 0 implementations
class UserController extends Controller
{
    1 reference | 0 overrides
    public function index(): Factory|View
    {
        //tambah data user dengan Eloquent Model
        $data = [

            'username' => 'customer-1',
            'nama' => 'Pelanggan',
            'password' => Hash::make(value: '12345'),
            'level_id' => 4
        ];
        UserModel::insert(values: $data);//tambahkan data ke tabel user

        // coba akses model UserModel
        $user = UserModel::all(); // ambil semua data dari tabel m_user
        return view(view: 'user', data: ['data' => $user]);
    }
```

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

Data User

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

10. Kita modifikasi lagi file UserController menjadi seperti berikut

app > Http > Controllers > UserController.php > PHP > UserController > index()

```php
<?php

namespace App\Http\Controllers;

use Illuminate\HTTP\Request;
use App\Models\UserModel;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Hash;

2 references | 0 implementations
class UserController extends Controller
{
    1 reference | 0 overrides
    public function index(): Factory|View
    {
        //tambah data user dengan Eloquent Model
        $data = [

            'nama' => 'Pelanggan Pertama';
        ];
        UserModel::insert(values: $data);//tambahkan data ke tabel user

        // coba akses model UserModel
        $user = UserModel::all(); // ambil semua data dari tabel m_user
        return view(view: 'user', data: ['data' => $user]);
    }
}
```

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

3	staff	Staf/Kasir	$2y$12$7uXHG2s9Ga9vzwxVXIBxH.mTo2kjlP3pXYgA65O(
4	customer-1	Customer 1	$2y$12$2UmDOikust5XLg3Pvwh/E.vfgzxGWXshEwcvZbfrm

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 di Laravel digunakan untuk **enkripsi data, session cookie, dan keamanan aplikasi**.

2. Pada Praktikum 1, bagaimana kita men-generate nilai untuk APP_KEY?
   Jawab : Cara untuk melakukan generate APP_KEY dengan menggunakan perintah 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 : Memiliki 3 file migrasi yaitu
   - create_users_table
   - Create_password_reset_token.php
   - create_failed_jobs_table.php
4. Secara default, file migrasi terdapat kode $table->timestamps();, apa tujuan/output dari fungsi tersebut?
   Jawab :
   Kode **$table->timestamps();** di file migrasi Laravel secara otomatis membuat **dua kolom** di tabel database:
   - **created_at** → Menyimpan waktu saat data dibuat.
   - **updated_at** → Menyimpan waktu saat data diperbarui
     Kedua kolom ini akan diisi **otomatis** oleh Eloquent saat insert atau update data

5. Pada File Migrasi, terdapat fungsi $table->id(); Tipe data apa yang dihasilkan dari fungsi tersebut?
   Jawab :
   Fungsi **$table->id();** di file migrasi Laravel menghasilkan kolom dengan **tipe data**: **BIGINT UNSIGNED AUTO INCREMENT**
   - Sebagai **primary key**
   - Secara default bernama **id**
   - Nilainya bertambah otomatis (auto-increment)

6. Apa bedanya hasil migrasi pada table m_level, antara menggunakan $table->id(); dengan menggunakan $table->id('level_id'); ?
   Jawab :
   Fungsinya sama, perbedaan pada **nama kolom primary key** di database:
   - **$table->id();** → kolom akan bernama **id**.
   - **$table->id('level_id');** → kolom akan bernama **level_id**.

7. Pada migration, Fungsi ->unique() digunakan untuk apa?
   Jawab : Dalam migration Laravel, fungsi **->unique()** digunakan untuk membuat **kolom dengan nilai unik**, artinya **tidak boleh ada data yang duplikat** 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 :
   - **Di m_level** → level_id dibuat dengan $table->id('level_id'), yang berarti **primary key** dengan tipe BIGINT UNSIGNED AUTO_INCREMENT.
   - **Di m_user** → level_id dibuat dengan $table >unsignedBigInteger('level_id'), karena kolom ini adalah **foreign key** yang merujuk ke level_id di m_level.
   - Sehingga m_user.level_id harus **tipe yang sama** dengan m_level.level_id untuk menjaga hubungan antar tabel.
   -
9. Pada Praktikum 3 - Tahap 6, apa tujuan dari Class Hash? dan apa maksud dari kode program Hash::make('1234');?
   Jawab :
   - **Class Hash** digunakan untuk mengenkripsi password sebelum disimpan di database.
   - **Hash::make('1234')** akan mengubah '1234' menjadi string hash **bcrypt** yang aman.

10. Pada Praktikum 4 - Tahap 3/5/7, pada query builder terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?
    Jawab :
    - Tanda **?** adalah **placeholder untuk binding parameter**.
    - Fungsinya untuk **mencegah SQL Injection** dengan menggantikan ? dengan nilai yang aman.

11. Pada Praktikum 6 - Tahap 3, apa tujuan penulisan kode protected $table = ‘m_user’; dan protected $primaryKey = ‘user_id’; ?
    Jawab:
    - **protected $table = 'm_user';**
      → Menentukan nama tabel di database yang digunakan oleh model **Eloquent ORM**.
    - **protected $primaryKey = 'user_id';**
      → Menentukan kolom yang digunakan sebagai **primary key** di tabel m_user.
    - **Tujuannya:** Agar Laravel tahu tabel dan primary key yang harus digunakan saat operasi CRUD.

12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (DB Façade / Query Builder / Eloquent ORM) ? jelaskan
    Jawab : Menggunakan query builder karena lebih fleksibel dan lebih mudah dibaca