PEMROGRAMAN WEB LANJUT

"Migration, Seeder, Db Façade, Query Builder, dan

Eloquent Orm"

Kelas : TI-2H

Disusun Oleh :

Fanesabhirawaning Sulistyo

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

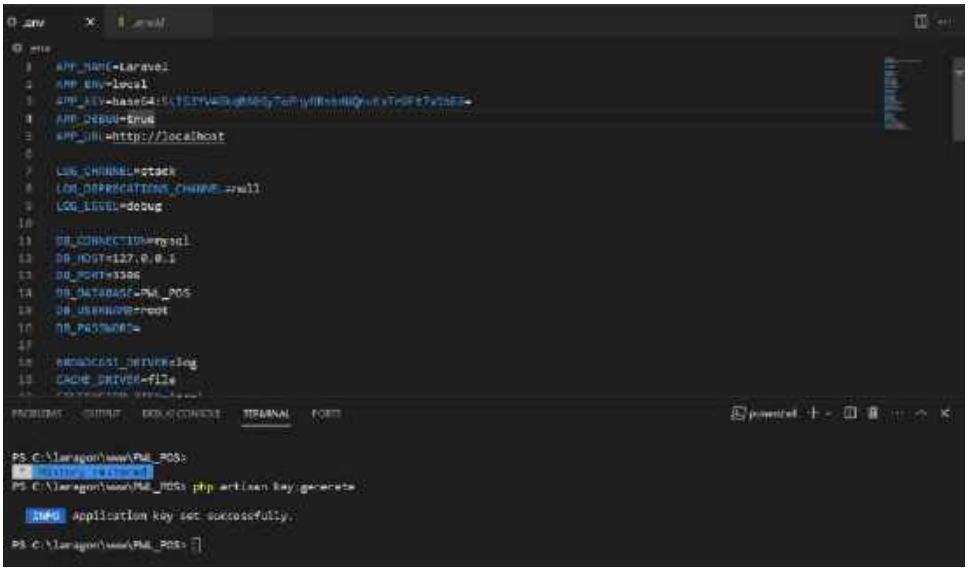
Jl. Soekarno Hatta No.9, Jatimulyo, Kec. Lowokwaru, Kota Malang, Jawa Timur 65141

A. PENGATURAN DATABASE

Praktikum 1 - pengaturan database:

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

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



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

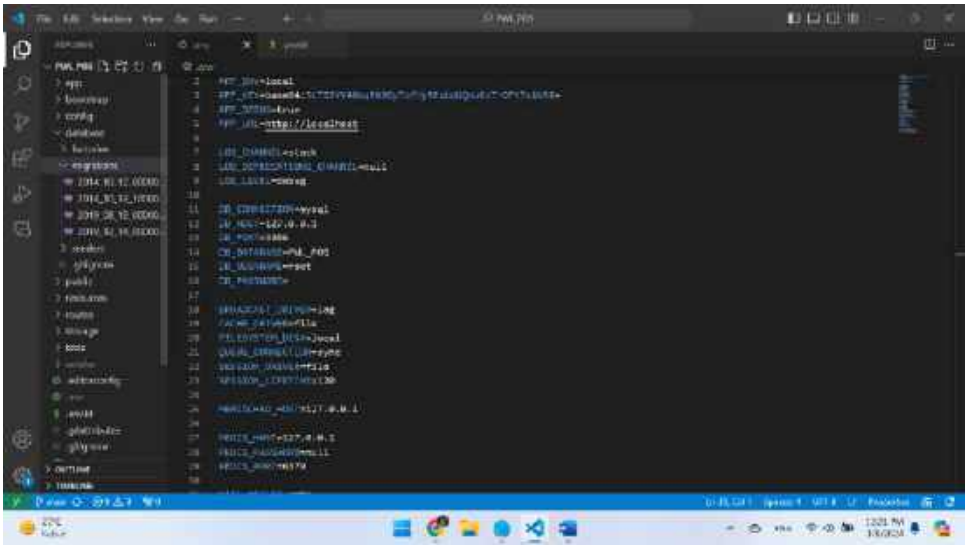
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=PWL_POS
DB_USERNAME=root
DB_PASSWORD=
```

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

B. MIGRATION

Praktikum 2.1 - Pembuatan file migrasi tanpa relasi

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



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

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

database > migrations > 2024_03_08_052340_create_m_level_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('m_level', function (Blueprint $table) {
            $table->id();
            $table->timestamps();
        });
    }

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

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

```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('m_level', function (Blueprint $table) {
            $table->id('level_id');
            $table->string('level_kode', 10)->unique();
            $table->string('level_nama', 100);
            $table->timestamps();
        });
    }

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

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

```
php artisan migrate
```

```
_reset_tokens_table  19ms DONE
  2019_08_19_000000_create_failed_jobs_table  46ms DONE
  2019_12_14_000001_create_personal_access_tokens_table  52ms DONE
  2024_03_08_052340_create_m_level_table  15ms DONE
```

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

7. Ok, table sudah dibuat di database
8. Buat table *database* dengan *migration* untuk table **`m_kategori`** yang sama-sama tidakmemiliki *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_08_053501_create_m_kategori_table.php] created succ
essfully.
```

```php
return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('m_kategori', function (Blueprint $table) {
            $table->id('kategori_id');
            $table->string('kategori_kode', 10)->unique();
            $table->string('kategori_nama', 100);
            $table->timestamps();
        });
    }
```

```
PS C:\laragon\www\PWL_POS> php artisan migrate
  INFO  Running migrations.
2024_03_08_053501_create_m_kategori_table ..... 40ms DONE
```

9. Laporkan hasil Praktikum-2.1 ini dan *commit* perubahan pada *git*.

Praktikum 2.2 - Pembuatan file migrasi dengan relasi

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

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

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

```php
public function up(): void
{
    Schema::create('m_user', function (Blueprint $table) {
        $table->id('user_id');
        $table->unsignedBigInteger('level_id')->index();
        $table->string('username',20)->unique();
        $table->string('nama',100);
        $table->string('password');
        $table->timestamps();

        $table->foreign('level_id')->references('level_id')-> on('m_level');
    });
}
```

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

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

m_barang
t_penjualan
t_stok
t_penjualan_detail

```
PS C:\laragon\www\PWL_POS> php artisan make:migration create_m_barang_table --create=m_barang
  INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2024_03_08_061009_create_m_barang_table.php] created successfully.
```

```php
public function up(): void
{
    Schema::create('m_barang', function (Blueprint $table) {
        $table->id('barang_id');
        $table->unsignedBigInteger('kategori_id')->index();
        $table->string('barang_kode', 10)->unique();
        $table->string('barang_nama', 100);
        $table->integer('harga_beli');
        $table->integer('harga_jual');
        $table->timestamps();

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

    });
}
```

```
PS C:\laragon\www\PWL_POS> php artisan migrate
  INFO  Running migrations.
2024_03_08_061009_create_m_barang_table ..... 9ms DONE
```

```
PS C:\laragon\www\PWL_POS> php artisan make:migration create_t_penjualan_table --create=t_penjualan
```

```php
public function up(): void
{
    Schema::create('t_penjualan', function (Blueprint $table) {
        $table->id('penjualan_id');
        $table->unsignedBigInteger('user_id')->index();
        $table->String('pembeli',50);
        $table->String('penjualan_kode',20);
        $table->date('penjualan_tanggal');
        $table->timestamps();

        $table->foreign('user_id')->references('user_id')->on('m_user');
    });
}
```

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

  INFO  Running migrations.

  2024_03_08_061622_create_t_penjualan_table  18ms DONE
```

```
PS C:\laragon\www\PWL_POS> php artisan make:migration create_t_stok_table --create=t_stok

  INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2024_03_08_063755_create_t_stok_table.php] created successfully.
```

```php
public function up(): void
{
    Schema::create('t_stok', function (Blueprint $table) {
        $table->id('stok_id');
        $table->unsignedBigInteger('barang_id')->index();
        $table->unsignedBigInteger('user_id')->index();
        $table->dateTime('stok_tanggal');
        $table->integer('stok_jumlah');
        $table->timestamps();

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

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

  INFO  Running migrations.

  2025_03_08_040758_create_t_stok_table ..... 134ms DONE
```

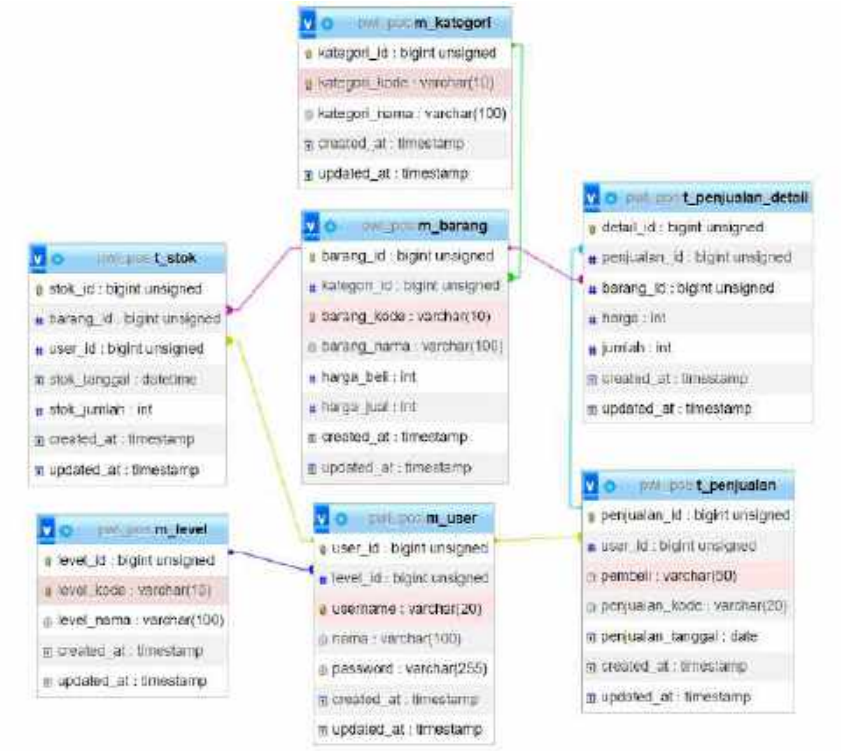
```
PS C:\laragon\www\PWL_POS> php artisan make:migration create_t_penjualan_detail_table --create=t_penjualan_detail

  INFO  Migration [C:\laragon\www\PWL_POS\database\migrations/2024_03_08_065226_create_t_penjualan_detail_table.php] created successfully.
```

```php
public function up(): void
{
    Schema::create('t_penjualan_detail', function (Blueprint $table) {
        $table->id('detail_id');
        $table->unsignedBigInteger('penjualan_id')->index();
        $table->unsignedBigInteger('barang_id')->index();
        $table->integer('harga');
        $table->integer('jumlah');
        $table->timestamps();

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

}
```

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

  INFO  Running migrations.

  2024_03_08_065226_create_t_penjualan_detail_table ..... 140ms DONE
```

5. Jika semua file migrasi sudah di buat dan dijalankan maka bisa kita lihat tampilan

designer pada **phpMyAdmin** seperti berikut



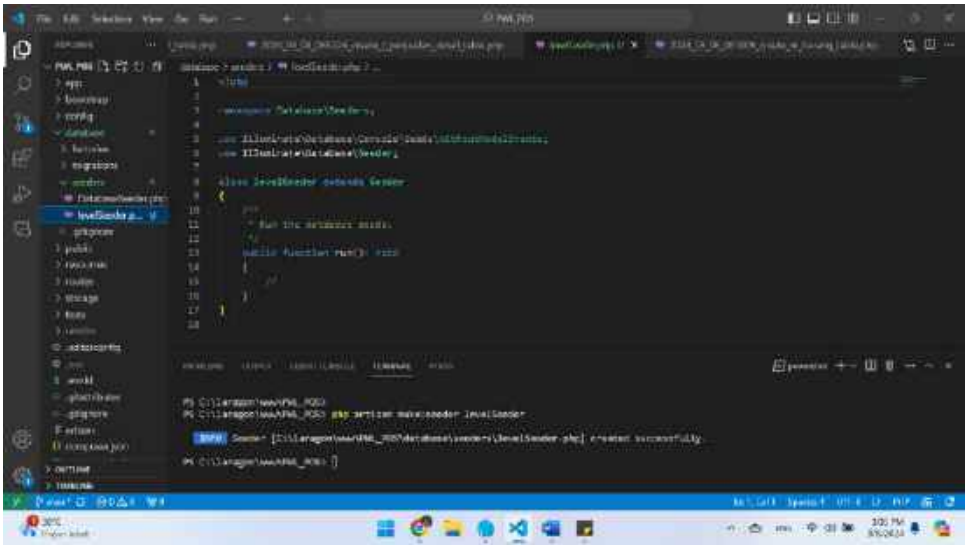
6. Laporkan hasil Praktikum-2.2 ini dan *commit* perubahan pada *git*.

C. SEEDER

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()`

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

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

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

  INFO  Seeding database.
```

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

	level_id	level_kode	level_nama	created_at	updated_at
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

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.
```

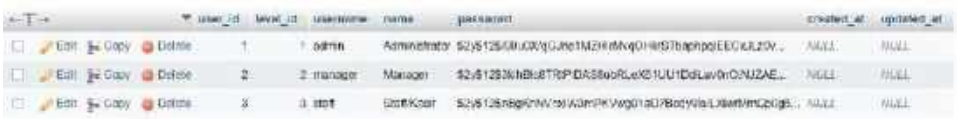
6. Modifikasi file `class UserSeeder` seperti berikut

```php
public function run(): void
{
    $data =[
        [
            'user_id' => 1,
            'level_id' => 1,
            'username' => 'admin',
            'nama' => 'Administrator',
            'password' => Hash::make('12345'),
        ],
        [
            'user_id' => 2,
            'level_id' => 2,
            'username' => 'manager',
            'nama' => 'Manager',
            'password' => Hash::make('12345'),
        ],
        [
            'user_id' => 3,
            'level_id' => 3,
            'username' => 'staff',
            'nama' => 'Staff/Kasir',
            'password' => Hash::make('12345'),
        ],
    ];
```

7. Jalankan perintah untuk mengeksekusi class `UserSeeder`

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

  INFO  Seeding database.
```

8. Perhatikan hasil seeder pada table `m_user`



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

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

```php
public function run(): void
{
    $data = [
        ['kategori_id' => 1, 'kategori_kode' => 'KTG001', 'kategori_nama' => 'Elektronik'],
        ['kategori_id' => 2, 'kategori_kode' => 'KTG002', 'kategori_nama' => 'Pakaian'],
        ['kategori_id' => 3, 'kategori_kode' => 'KTG003', 'kategori_nama' => 'Makanan'],
        ['kategori_id' => 4, 'kategori_kode' => 'KTG004', 'kategori_nama' => 'Buku'],
        ['kategori_id' => 5, 'kategori_kode' => 'KTG005', 'kategori_nama' => 'Mainan'],
    ];
    DB::table('m_kategori')->insert($data);
}
```

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

  INFO  Seeding database.
```

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
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	Buku	NULL	NULL
Edit Copy Delete	5	KTG005	Mainan	NULL	NULL

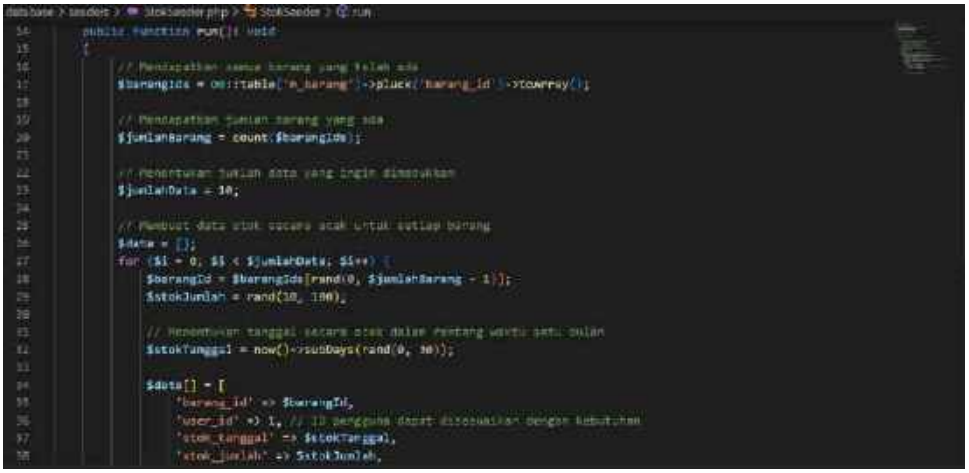
```
  INFO  Seeder [C:\laragon\www\PWL_POS\database\seeders\BarangSeeder.php] created successfully.
```

```php
$data = [
    ['barang_id' => 1, 'kategori_id' => 1, 'barang_kode' => 'BRG001', 'barang_nama' => 'Laptop', 'harga_beli' => 5000000, 'harga_jual' =>
    ['barang_id' => 2, 'kategori_id' => 1, 'barang_kode' => 'BRG002', 'barang_nama' => 'Smartphone', 'harga_beli' => 3000000, 'harga_jual
    ['barang_id' => 3, 'kategori_id' => 2, 'barang_kode' => 'BRG003', 'barang_nama' => 'Kemeja', 'harga_beli' => 200000, 'harga_jual' =>
    ['barang_id' => 4, 'kategori_id' => 2, 'barang_kode' => 'BRG004', 'barang_nama' => 'Celana Jeans', 'harga_beli' => 250000, 'harga_jua
    ['barang_id' => 5, 'kategori_id' => 3, 'barang_kode' => 'BRG005', 'barang_nama' => 'Mie Instan', 'harga_beli' => 5000, 'harga_jual' =>
    ['barang_id' => 6, 'kategori_id' => 3, 'barang_kode' => 'BRG006', 'barang_nama' => 'Susu', 'harga_beli' => 20000, 'harga_jual' => 250
    ['barang_id' => 7, 'kategori_id' => 4, 'barang_kode' => 'BRG007', 'barang_nama' => 'Buku Tulis', 'harga_beli' => 10000, 'harga_jual' =>
    ['barang_id' => 8, 'kategori_id' => 4, 'barang_kode' => 'BRG008', 'barang_nama' => 'Buku Pelajaran', 'harga_beli' => 50000, 'harga_jua
    ['barang_id' => 9, 'kategori_id' => 5, 'barang_kode' => 'BRG009', 'barang_nama' => 'Boneka', 'harga_beli' => 30000, 'harga_jual' => 4
    ['barang_id' => 10, 'kategori_id' => 5, 'barang_kode' => 'BRG010', 'barang_nama' => 'Puzzle', 'harga_beli' => 15000, 'harga_jual' =>
];
```

```
PS C:\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	updated_at
Edit Copy Delete	1	1	BRG001	Laptop	5000000	7000000	NULL	NULL
Edit Copy Delete	2	1	BRG002	Smartphone	3000000	4000000	NULL	NULL
Edit Copy Delete	3	2	BRG003	Kemeja	200000	300000	NULL	NULL
Edit Copy Delete	4	2	BRG004	Celana Jeans	250000	350000	NULL	NULL
Edit Copy Delete	5	3	BRG005	Mie instan	5000	7000	NULL	NULL
Edit Copy Delete	6	3	BRG006	Susu	20000	25000	NULL	NULL
Edit Copy Delete	7	4	BRG007	Buku Tulis	10000	12000	NULL	NULL
Edit Copy Delete	8	4	BRG008	Buku Pelajaran	50000	60000	NULL	NULL
Edit Copy Delete	9	5	BRG009	Boneka	30000	40000	NULL	NULL
Edit Copy Delete	10	5	BRG010	Puzzle	15000	20000	NULL	NULL

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



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

  INFO  Seeding database.
```

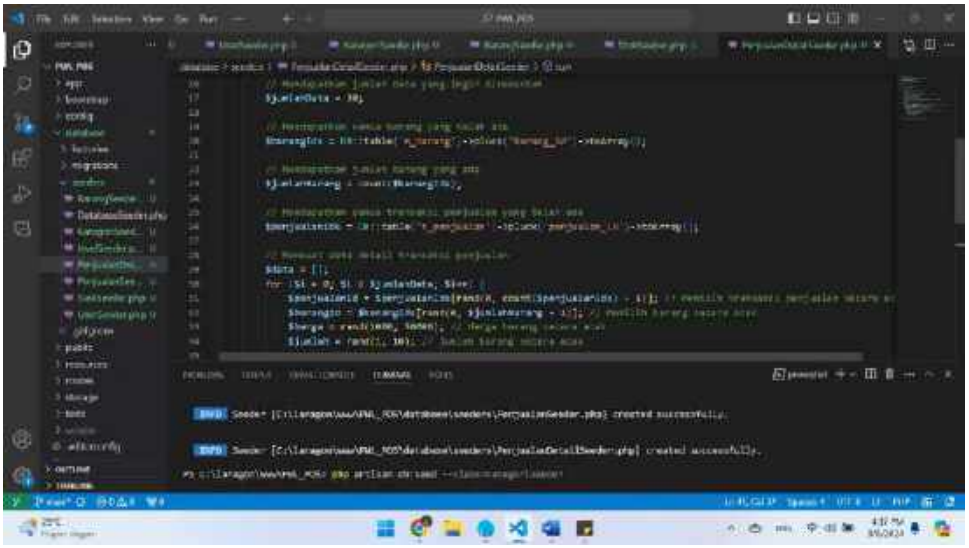
	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
Edit Copy Delete	1	5	1	2024-02-27 09:15:14	36	NULL	NULL
Edit Copy Delete	2	8	1	2024-02-07 09:15:14	66	NULL	NULL
Edit Copy Delete	3	5	1	2024-03-07 09:15:14	99	NULL	NULL
Edit Copy Delete	4	3	1	2024-02-16 09:15:14	60	NULL	NULL
Edit Copy Delete	5	2	1	2024-02-08 09:15:14	67	NULL	NULL
Edit Copy Delete	6	7	1	2024-02-15 09:15:14	82	NULL	NULL
Edit Copy Delete	7	1	1	2024-02-20 09:15:14	12	NULL	NULL
Edit Copy Delete	8	10	1	2024-03-07 09:15:14	32	NULL	NULL
Edit Copy Delete	9	10	1	2024-02-24 09:15:14	20	NULL	NULL
Edit Copy Delete	10	1	1	2024-02-29 09:15:14	53	NULL	NULL

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

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

  INFO  Seeding database.
```

	penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
Edit Copy Delete	1	1	Pelanggan 1	PJ00001	2024-02-18	NULL	NULL
Edit Copy Delete	2	1	Pelanggan 2	PJ00002	2024-02-07	NULL	NULL
Edit Copy Delete	3	1	Pelanggan 3	PJ00003	2024-02-09	NULL	NULL
Edit Copy Delete	4	1	Pelanggan 4	PJ00004	2024-02-08	NULL	NULL
Edit Copy Delete	5	1	Pelanggan 5	PJ00005	2024-02-11	NULL	NULL
Edit Copy Delete	6	1	Pelanggan 6	PJ00006	2024-02-25	NULL	NULL
Edit Copy Delete	7	1	Pelanggan 7	PJ00007	2024-03-01	NULL	NULL
Edit Copy Delete	8	1	Pelanggan 8	PJ00008	2024-03-02	NULL	NULL
Edit Copy Delete	9	1	Pelanggan 9	PJ00009	2024-03-06	NULL	NULL
Edit Copy Delete	10	1	Pelanggan 10	PJ00010	2024-02-19	NULL	NULL

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



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

  INFO  Seeding database.
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

1 > >> | Show all | Number of rows: 25 | Filter rows: Search this table | Sort by

Extra options

	detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
Edit Copy Delete	1	9	2	17593	5	NULL	NULL
Edit Copy Delete	2	9	3	20403	5	NULL	NULL
Edit Copy Delete	3	10	3	16451	10	NULL	NULL
Edit Copy Delete	4	6	5	34634	7	NULL	NULL
Edit Copy Delete	5	10	5	24409	4	NULL	NULL
Edit Copy Delete	6	2	5	8358	4	NULL	NULL
Edit Copy Delete	7	6	10	21346	1	NULL	NULL
Edit Copy Delete	8	6	5	35726	3	NULL	NULL

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

D. DB FACADE

Praktikum 4 – Implementasi DB Facade

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

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

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

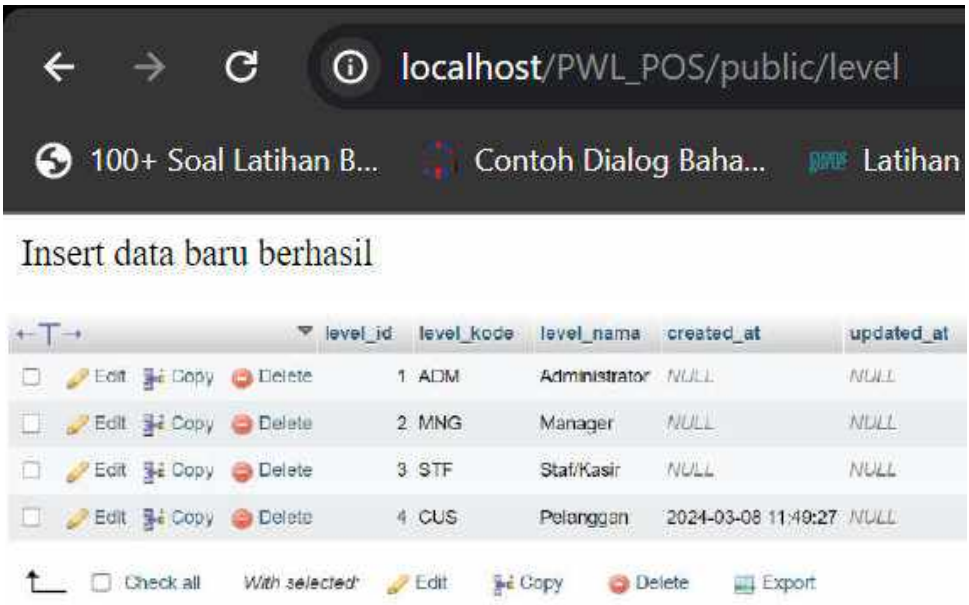
2. Kita modifikasi dulu untuk *routing*-nya, ada di `PWL_POS/routes/web.php`

```php
Route::get('/', function () {
    return view('welcome');
});

Route::get('/level',[levelController::class,'index']);
```

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

```php
class levelController extends Controller
{
    public function index()
    {
        DB::insert('insert into m_level (level_kode, level_nama,created_at) values (?, ?, ?)', ['CUS','Pelanggan', now
        return'inser data baru berhasil';
    }
}
```

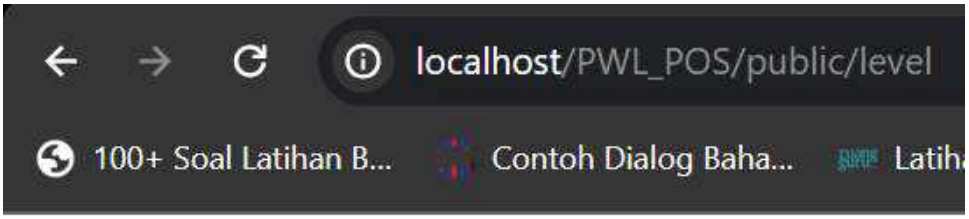
4. Kita coba jalankan di browser dengan url `localhost/PWL_POS/public/level` dan amatiapa yang terjadi pada table `m_level` di database, *screenshot* perubahan yang ada pada table `m_level`



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

```
class levelController extends Controller
{
    public function index()
    {
        // DB::insert('insert into m_level (level_kode, level_nama, created_at) values (?, ?, ?)', ['CUS', '
        // return 'Insert data baru berhasil';

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

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



Update data berhasil.Jumlah data yang diupdate: 1baris

	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	8	CUS	Customer	2024-03-08 12:17:37	2024-03-08 12:21:08

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

```
// Melakukan penghapusan data berdasarkan level_kode
$row = DB::delete('delete from m_level where level_kode = ?', ['CUS']);

// Mengembalikan pesan konfirmasi bahwa data berhasil dihapus
return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row . ' baris';
```

Delete data berhasil. Jumlah data yang dihapus: 1 baris

	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

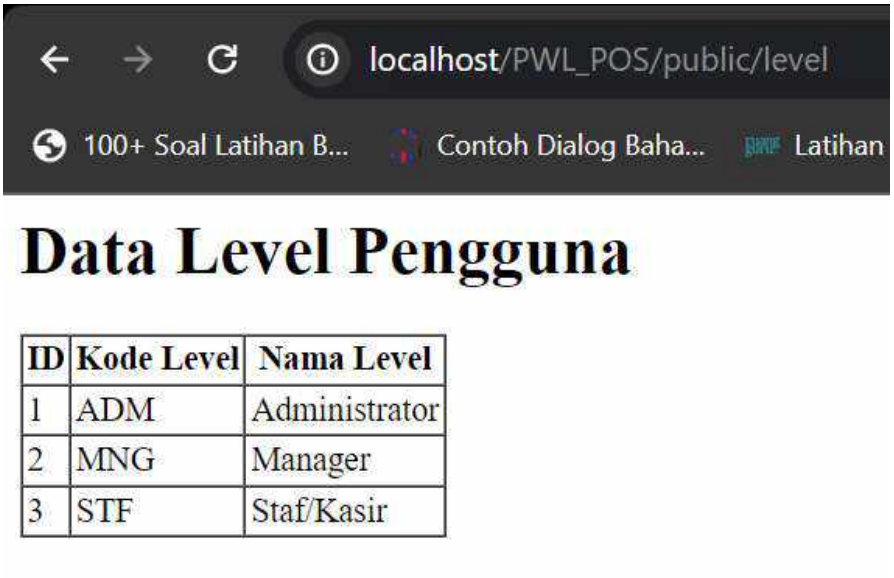
8. Method terakhir yang kita coba adalah untuk menampilkan data yang ada di table `m_level`. Kita modifikasi file `LevelController` seperti berikut

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

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

```
<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



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

E. QUERY BUILDER

Praktikum 5 – Implementasi *Query Builder*

1. Kita buat controller dahuku 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.
```

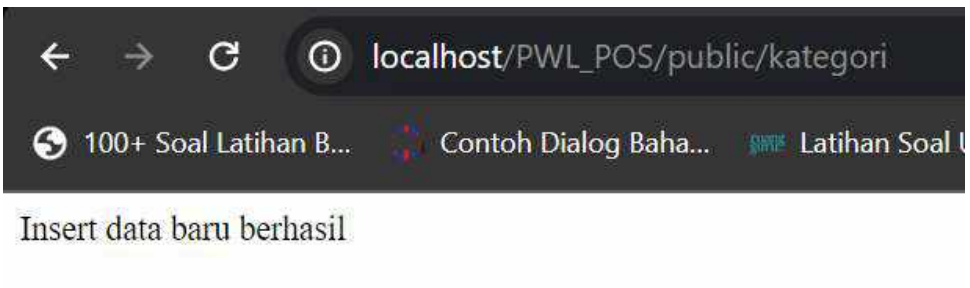
2. Kita modifikasi dulu untuk routing-nya, ada di `PWL_POS/routes/web.php`

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

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

```
namespace App\Http\Controllers;

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

class KategoriController extends Controller
{
    public function index()
    {
        $data = [
            'kategori_kode' => 'SMK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert data baru berhasil';
    }
}
```

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



	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
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	Buku	NULL	NULL
Edit Copy Delete	5	KTG005	Mainan	NULL	NULL
Edit Copy Delete	6	SMK	Snack/Makanan Ringan	2024-03-08 12:58:21	NULL

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

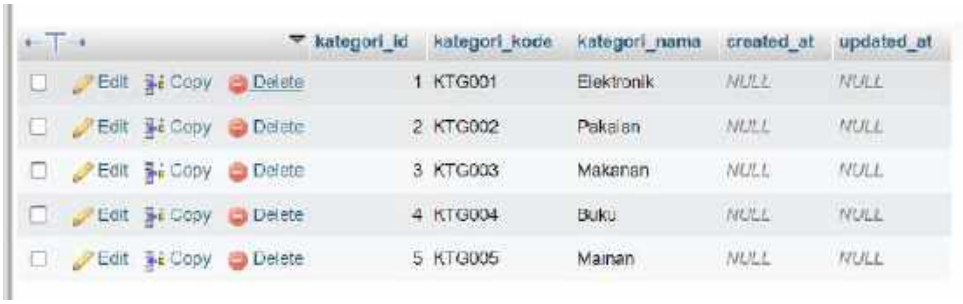
```
$row = DB::table('m_kategori')->where('kategori_kode', 'SMK')->update(['kategori_nama' => 'Camilan']);

// Mengembalikan pesan konfirmasi bahwa data berhasil diperbarui
return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
```

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

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
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	Buku	NULL	NULL
Edit Copy Delete	5	KTG005	Mainan	NULL	NULL
Edit Copy Delete	6	SMK	Camilan	2024-03-08 12:58:21	2024-03-08 13:05:40

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

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



	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
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	Buku	NULL	NULL
Edit Copy Delete	5	KTG005	Mainan	NULL	NULL

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

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

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

```
<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>
</html>
```

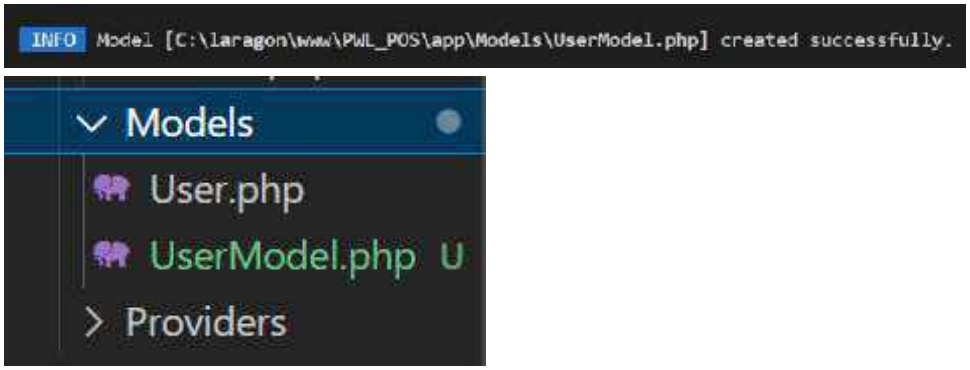
10. Silahkan dicoba pada browser dan amati apa yang terjadi.

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

F. ELOQUENT ORM

Praktikum 6 – Implementasi Eloquent ORM

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



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

```php
class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_user';
    protected $primaryKey = 'user_id';
}
```

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

```php
Route::get('/level',[levelController::class,'index']);
Route::get('/kategori', [KategoriController::class, 'index']);
Route::get('/user', [UserController::class, 'index']);
```

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

```php
class UserController extends Controller
{
    public function index()
    {
        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }
}
```

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

```
<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>
</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`

```php
class UserController extends Controller
{
    public function index()
    {
        // Menyiapkan data untuk dimasukkan ke dalam tabel m_user
        $data = [
            'username' => 'customer-1',
            'nama' => 'Pelanggan',
            'password' => Hash::make('12345'), // Menggunakan Hash::make() untuk mengenkripsi password
            'level_id' => 4
        ];

        // Memasukkan data baru ke dalam tabel menggunakan model UserModel
        UserModel::insert($data);

        // Mengambil semua data pengguna dari tabel m_user
        $users = UserModel::all();

        // Mengembalikan view 'user' dengan data pengguna
        return view('user', ['data' => $users]);
    }
}
```

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	Staff/Kasir	3
8	customer-1	Pelanggan	4

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

```php
class UserController extends Controller
{
    public function index()
    {
        // Menyiapkan data untuk dimasukkan ke dalam tabel m_user
        $data= [
         'nama' => 'Pelanggan Pertama',
        ];
        UserModel::where('username','customer-1')->update($data);

        // Mengambil semua data pengguna dari tabel m_user
        $users = UserModel::all();

        // Mengembalikan view 'user' dengan data pengguna
        return view('user', ['data' => $users]);
    }
}
```

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

Data User

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

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

G. Penutup

Jawablah pertanyaan berikut sesuai pemahaman materi di atas

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

   Jawab:

   Sebagai encryption key untuk aplikasi kita. Laravel menggunakan string acak tersebut untuk meng-encrypt data yang kita butuhkan, misalnya cookies ataupun password.

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

   Jawab:

   Dengan menggunakan perintah php:

   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:

   - **2014_10_12_000000_create_users_table.php**: File migrasi ini bertanggung jawab untuk membuat tabel users dalam database untuk menyimpan informasi seperti nama pengguna, alamat email, dan password.
   - **2014_10_12_100000_create_password_reset_tokens_table.php**: File migrasi ini digunakan untuk membuat tabel password_reset_tokens dalam database untuk menyimpan token yang digunakan untuk mengatur ulang password pengguna ketika lupa password.
   - **2019_08_19_000000_create_failed_jobs_table.php**: File migrasi ini bertugas untuk membuat tabel failed_jobs dalam database untuk menyimpan informasi tentang pekerjaan yang gagal dieksekusi oleh pekerjaan antrian.
   - **2019_12_14_000001_create_personal_access_tokens_table.php**: File migrasi ini digunakan untuk membuat tabel personal_access_tokens dalam database terkait dengan otentikasi dan otorisasi pengguna, dan digunakan khususnya untuk menyimpan token akses pribadi yang diperoleh oleh pengguna.

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

   Jawab:

   Kode $table->timestamps(); bertujuan untuk secara otomatis menambahkan dua

kolom ke dalam tabel yang sedang dibuat, yaitu created_at dan updated_at

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

   Jawab:

   fungsi $table->id(); dalam Laravel setara dengan membuat kolom dengan tipe data BIGINT UNSIGNED (64-bit unsigned integer), dengan opsi AUTO_INCREMENTdan PRIMARY KEY

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

   dengan menggunakan `$table->id('level_id');` ?

   Jawab:

   Saat menggunakan **$table->id();**, nama kolom yang digunakan adalah **id** secara default, sedangkan saat menggunakan **$table->id('level_id');**, kita dapat menentukan nama kolom sesuai dengan preferensi kita, dalam hal ini adalah **level_id**

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

   Jawab:

   Dalam sebuah file migrasi Laravel digunakan untuk menetapkan indeks unik pada kolom tertentu dalam tabel yang sedang dibuat. Ini berarti bahwa setiap nilai dalam kolom yang harus unik di antara semua baris dalam tabel.

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:

   tabel m_user, kolom level_id digunakan sebagai FK yang mengacu ke tabel lain, sementara pada tabel m_level, kolom level_id digunakan sebagai PK dari tabel itu sendiri.

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

   Jawab:

   Hash::make('1234');, fungsi make() digunakan untuk mengenkripsi string 1234

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

    Jawab:

Kita tanya (?) dalam query builder Laravel digunakan untuk menempatkan nilai-nilai yang akan disubstitusi ke dalam query saat eksekusi. Ini memungkinkan kita untuk membuat query dinamis dengan nilai-nilai yang berubah-ubah.

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

    Jawab:

    - Penulisan **protected $table = 'm_user';** memberitahu model Laravel bahwa model tersebut terkait dengan tabel database bernama 'm_user'. Ini membuat Laravel tahu ke mana harus mencari data ketika menggunakan model ini.
    - Sementara itu, penulisan **protected $primaryKey = 'user_id';** memberitahu Laravel bahwa kolom 'user_id' adalah kunci utama (primary key) untuk tabel 'm_user'. Ini penting karena Laravel akan menggunakan kolom ini untuk mengidentifikasi secara unik setiap baris data dalam tabel.

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

    Jawab:

    Tergantung pada kebutuhan dan preferensi, namun saya lebih memilih Eloquent ORM lebih mudah digunakan karena:

    1. Eloquent menyediakan cara yang mudah dimengerti untuk berinteraksi dengan database menggunakan objek PHP biasa, membuat penulisan kode menjadi lebih mudah dipahami.
    2. Dengan Eloquent, Kita dapat melakukan operasi CRUD tanpa harus menulis kueri SQL secara langsung. Ini mengurangi kompleksitas dan waktu yang diperlukan untuk mengembangkan aplikasi.
    3. Eloquent dilengkapi dengan berbagai fitur bawaan seperti relasi antar-model, eager loading, dan lainnya, yang mempercepat pengembangan aplikasi.