TUGAS

Web Desain Lanjut

PERTEMUAN 3

Migration

OLEH

MUHAMMAD NUR AZIZ NIM. 2341720237

PROGRAM STUDI D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

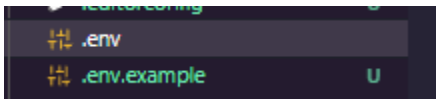
POLITEKNIK NEGERI MALANG

24 February 2025

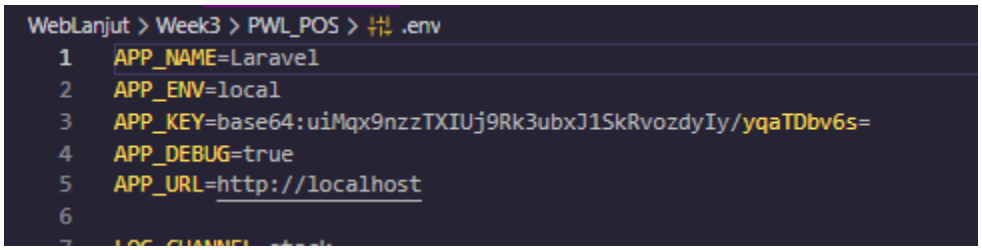
A. Pengaturan Database

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 silahkan kalian generate menggunakan php artisan.
5. Edit file .env dan sesuaikan dengan database yang telah dibuat

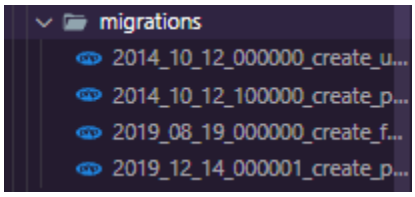


6. Laporkan hasil Praktikum-1 ini dan commit perubahan pada git.

B. Migration

Praktikum-2.1 Pembuatan File 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

```
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS> php artisan make:migration
create_m_level_table --create=m_level

  INFO  Migration [C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS\database
\migrations/2025_02_25_012647_create_m_level_table.php] created successf
ully.
```

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

```php
    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();
        });
    }
```

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

```
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>php artisan migrate

  INFO  Preparing database.

  Creating migration table ................................. 35ms DONE

  INFO  Running migrations.

  2014_10_12_000000_create_users_table ...................... 42ms DONE
  2014_10_12_100000_create_password_reset_tokens_table ..... 14ms DONE
  2019_08_19_000000_create_failed_jobs_table ................ 35ms DONE
  2019_12_14_000001_create_personal_access_tokens_table .... 46ms DONE
  2025_02_25_012647_create_m_level_table .................... 31ms DONE
```

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

	Table	Action	Rows	Type	Collation	Size	Overhead
☐	failed_jobs	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	migrations	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	m_level	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 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	-
☐	users	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
	6 tables	Sum	5	InnoDB	utf8mb4_0900_ai_ci	96.0 KiB	0 B

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

```
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS> php artisan make:migration
create_m_kategori_table --create=m_kategori

  INFO  Migration [C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS\database
\migrations/2025_02_25_013632_create_m_kategori_table.php] created succe
ssfully.
```

	Table	Action						Rows	Type	Collation	Size	Overhead
☐	failed_jobs	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
☐	migrations	Browse	Structure	Search	Insert	Empty	Drop	6	InnoDB	utf8mb4_unicode_ci	16.0 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	-
☐	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	-
☐	users	Browse	Structure	Search	Insert	Empty	Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-

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

Praktikum-2.2 Pembuatan File Dengan Relasi

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

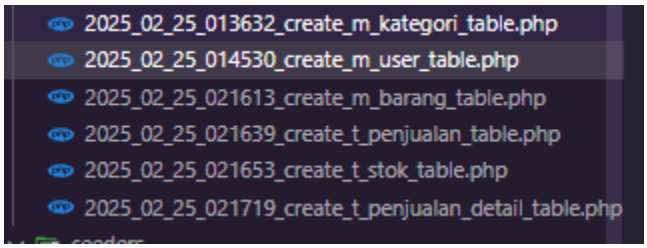
```
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS> php artisan make:migration
create_m_user_table --create=m_user

  INFO  Migration [C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS\database
\migrations/2025_02_25_014530_create_m_user_table.php] created successfu
lly.
```

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

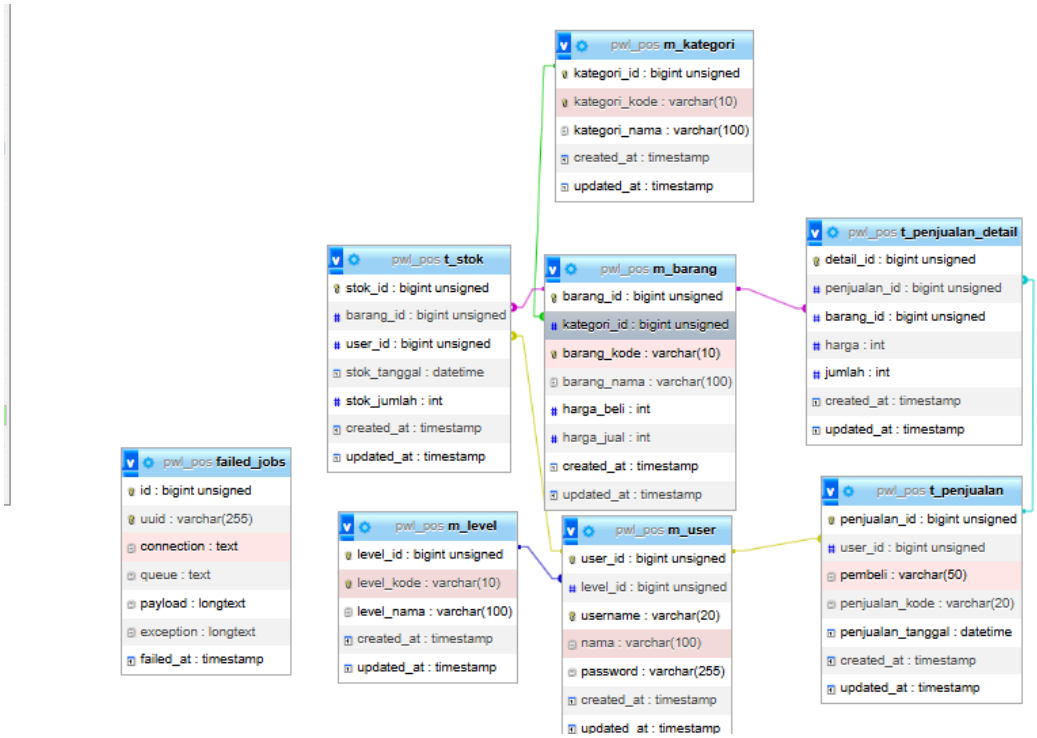
```php
public function up(): void
{
    Schema::create('m_user', function (Blueprint $table) {
        $table->id('user_id');
        $table->unsignedBigInteger('level_id')->index(); //index fk
        $table->string('username', 20)->unique(); // unique untuk tidak ada yang sama
        $table->string('nama', 100);
        $table->string('password', 255);
        $table->timestamps();

        // define fk pada kolom level_id ke kolom level_id ditabel m_level
        $table->foreign('level_id')->references('level_id')->on('m_level');
    });
}
```

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



C. Seeder

Praktikum-3 Membuat File Seeder

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

```
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>php artisan make:seeder leve
lseeder

  INFO  Seeder [C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS\database\se
eders\levelseeder.php] created successfully.


C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>
```

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

```
Codeium: Refactor | Explain | X
    public function run(): void
    {
        $data = [
            ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
            ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
            ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir'],
        ];
        DB::table('m_level')->insert($data);
    }
}
```

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

```
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>php artisan db:seed --class=
LevelSeeder

  INFO  Seeding database.
```

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

Extra options

	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	Staff/Kasir	NULL	NULL

Check all *With selected:* Edit Copy Delete Export

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

```
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>php artisan make:seeder User
Seeder

  INFO  Seeder [C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS\database\se
eders\UserSeeder.php] created successfully.
```

6. Modifikasi file class UserSeeder seperti berikut

```php
Codeium: Refactor | Explain | X
public function run(): void
{
    $data = [
        [
            'user_id' => 1,
            'level_id' => 1,
            'username' => 'admin',
            'nama' => 'Administrator',
            'password' => Hash::make('12345'),
        ],
        [
            'user_id' => 2,
            'level_id' => 2,
            'username' => 'manager',
            'nama' => 'Manager',
            'password' => Hash::make('12345'),
        ],
        [
            'user_id' => 3,
            'level_id' => 3,
            'username' => 'staff',
            'nama' => 'Staff/Kasir',
            'password' => Hash::make('12345'),
        ],
    ];
    DB::table('m_user')->insert($data);
}
```

7. Jalankan perintah untuk mengeksekusi class UserSeeder

```
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>php artisan db:seed --class=
UserSeeder

  INFO  Seeding database.


C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>
```

8. Perhatikan hasil seeder pada table m_user

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	user_id	level_id	username	nama	password	created_at	updated_at
Edit Copy Delete	1	1	admin	Administrator	$2y$12$.c7TKpSGyz8858z/vFDkuOBBYVK4Pb3YldBOAcAIK7e...	NULL	NULL
Edit Copy Delete	2	2	manager	Manager	$2y$12$Iv.kWhUVPlewL/IfADbLTOZI00kM8Bk.8oFsUhrqBJo...	NULL	NULL
Edit Copy Delete	3	3	staff	Staff/Kasir	$2y$12$G5FM2wAqGz3rZGm8RJlgce9Kat1sEsBSmolqz/WfnVW...	NULL	NULL

Check all With selected: Edit Copy Delete Export

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

- m_kategori

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
Edit Copy Delete	1	makanan	Makanan	NULL	NULL
Edit Copy Delete	2	minuman	Minuman	NULL	NULL
Edit Copy Delete	3	snack	Snack	NULL	NULL
Edit Copy Delete	4	alat_tulis	Alat Tulis	NULL	NULL
Edit Copy Delete	5	elektronik	Elektronik	NULL	NULL

- m_barang

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

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
Edit Copy Delete	1	1	MK001	Nasi Goreng	8000	10000	NULL	NULL
Edit Copy Delete	2	1	MK002	Mie Sedaap	5000	8000	NULL	NULL
Edit Copy Delete	3	2	MN001	Air Mineral	3000	4000	NULL	NULL
Edit Copy Delete	4	2	MN002	Jus Apel	8000	10000	NULL	NULL
Edit Copy Delete	5	3	SK001	Krupuk	2000	2500	NULL	NULL
Edit Copy Delete	6	3	SK002	Kacang	4000	5000	NULL	NULL
Edit Copy Delete	7	4	AT001	Pensil	2500	3000	NULL	NULL
Edit Copy Delete	8	4	AT002	Penghapus	3500	4000	NULL	NULL
Edit Copy Delete	9	5	EL001	Laptop	4000000	4500000	NULL	NULL
Edit Copy Delete	10	5	EL002	Handphone	2500000	2800000	NULL	NULL

Check all *With selected:* Edit Copy Delete Export

- t_stok

Extra options

	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
Edit Copy Delete	1	1	3	2025-02-28 14:23:18	112	NULL	NULL
Edit Copy Delete	2	2	3	2025-02-28 14:23:18	79	NULL	NULL
Edit Copy Delete	3	3	2	2025-02-28 14:23:18	214	NULL	NULL
Edit Copy Delete	4	4	2	2025-02-28 14:23:18	234	NULL	NULL
Edit Copy Delete	5	5	3	2025-02-28 14:23:18	126	NULL	NULL
Edit Copy Delete	6	6	2	2025-02-28 14:23:18	159	NULL	NULL
Edit Copy Delete	7	7	2	2025-02-28 14:23:18	60	NULL	NULL
Edit Copy Delete	8	8	2	2025-02-28 14:23:18	23	NULL	NULL
Edit Copy Delete	9	9	2	2025-02-28 14:23:18	123	NULL	NULL
Edit Copy Delete	10	10	1	2025-02-28 14:23:18	108	NULL	NULL

Check all *With selected:* Edit Copy Delete Export

- t_penjualan

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

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
Edit Copy Delete	1	3	Staff/Kasir	penjualan0	2025-02-28 15:27:00	NULL	NULL
Edit Copy Delete	2	1	Administrator	penjualan1	2025-02-28 15:27:00	NULL	NULL
Edit Copy Delete	3	1	Administrator	penjualan2	2025-02-28 15:27:00	NULL	NULL
Edit Copy Delete	4	1	Administrator	penjualan3	2025-02-28 15:27:00	NULL	NULL
Edit Copy Delete	5	3	Staff/Kasir	penjualan4	2025-02-28 15:27:00	NULL	NULL
Edit Copy Delete	6	3	Staff/Kasir	penjualan5	2025-02-28 15:27:00	NULL	NULL
Edit Copy Delete	7	1	Administrator	penjualan6	2025-02-28 15:27:00	NULL	NULL
Edit Copy Delete	8	2	Manager	penjualan7	2025-02-28 15:27:00	NULL	NULL
Edit Copy Delete	9	1	Administrator	penjualan8	2025-02-28 15:27:00	NULL	NULL
Edit Copy Delete	10	1	Administrator	penjualan9	2025-02-28 15:27:00	NULL	NULL

Check all *With selected:* Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

t_penjualan_detail

1 > >> | Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

Extra options

	detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
Edit	1	1	4	10000	15	NULL	NULL
Edit Copy Delete	2	1	4	10000	7	NULL	NULL
Edit Copy Delete	3	1	4	10000	6	NULL	NULL
Edit Copy Delete	4	2	9	4500000	13	NULL	NULL
Edit Copy Delete	5	2	9	4500000	5	NULL	NULL
Edit Copy Delete	6	2	9	4500000	4	NULL	NULL
Edit Copy Delete	7	3	1	10000	13	NULL	NULL
Edit Copy Delete	8	3	1	10000	7	NULL	NULL
Edit Copy Delete	9	3	1	10000	5	NULL	NULL
Edit Copy Delete	10	4	7	3000	5	NULL	NULL
Edit Copy Delete	11	4	7	3000	4	NULL	NULL
Edit Copy Delete	12	4	7	3000	11	NULL	NULL
Edit Copy Delete	13	5	2	8000	6	NULL	NULL
Edit Copy Delete	14	5	2	8000	10	NULL	NULL
Edit Copy Delete	15	5	2	8000	6	NULL	NULL
Edit Copy Delete	16	6	1	10000	10	NULL	NULL

Click the drop-down arrow to toggle column's visibility.

Console

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

D. DB Façade

Praktikum 4 Implementasi DB Facade

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

```
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>php artisan make:controller LevelController

  INFO  Controller [C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS\app\Http\Controllers\LevelController.php] created successfully.
```

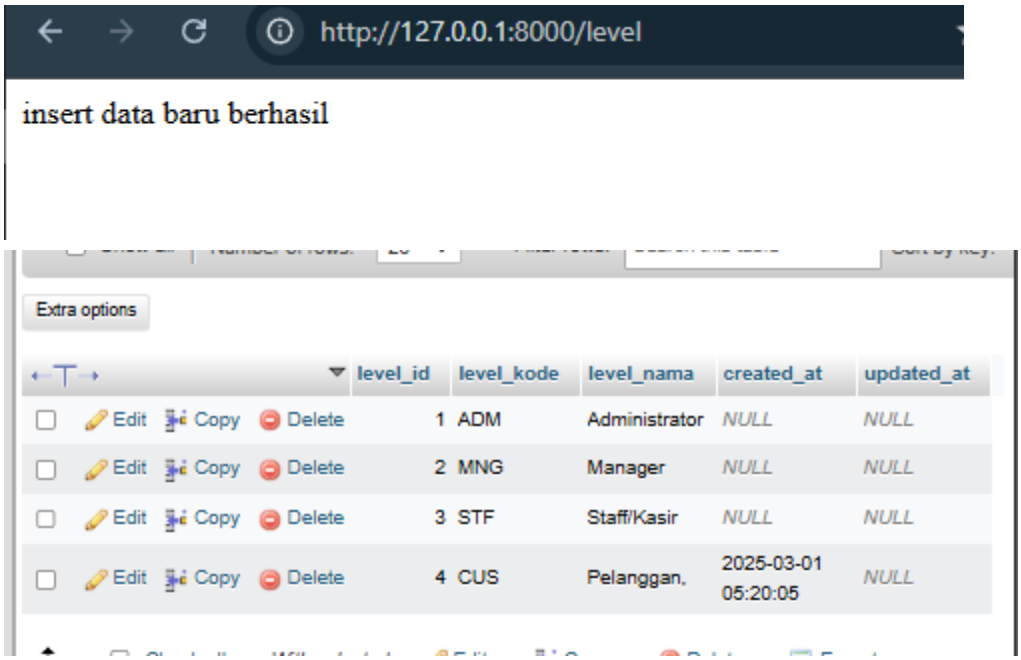
2. Kita modifikasi dulu untuk routing-nya, ada di PWL_POS/routes/web.php

```
14  */
15
16  Route::get('/', function () {
17      return view('welcome');
18  });
19
20  Route::get('/level', [LevelController::class, 'index']);
21
```

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

```
{
    // insert data baru
    DB::insert(
        'insert into m_level(level_kode, level_nama, created_at) values (?,?,?)',
        ['CUS', 'Pelanggan,', now()]
    );
    return 'insert data baru berhasil';
```

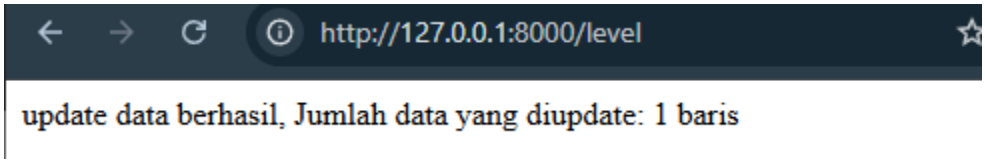
4. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/level dan amati apa yang terjadi pada table m_level di database, screenshot perubahan yang ada pada table m_level

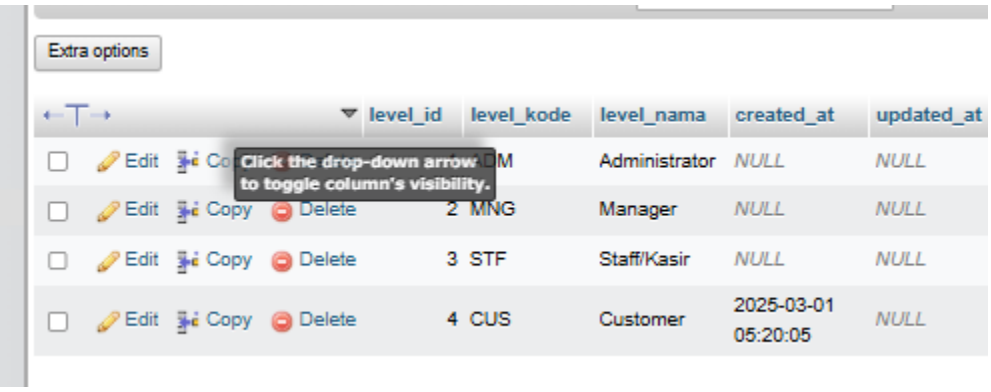


5. Selanjutnya, kita modifikasi lagi file LevelController untuk meng-update data di table m_level seperti berikut

```
// update data
$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





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

```
// hapus data
$row = DB::delete(
    'delete from m_level where level_kode = ?',
    ['CUS']
);
return 'hapus data berhasil, Jumlah data yang dihapus: ' . $row . ' baris';
```

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

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

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

```php
<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>
```

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

Praktikum-5 Implementasi Query Builder

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

```
^C
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>php artisan make:controller
KategoriController

  INFO  Controller [C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS\app\Http\Controllers\KategoriController.php] created successfully.
```

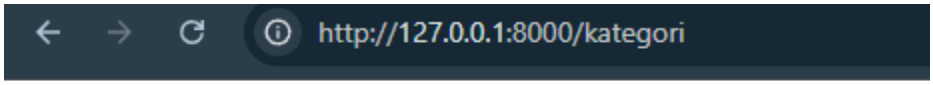
2. Kita modifikasi dulu untuk routing-nya, ada di PWL_POS/routes/web.php

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

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

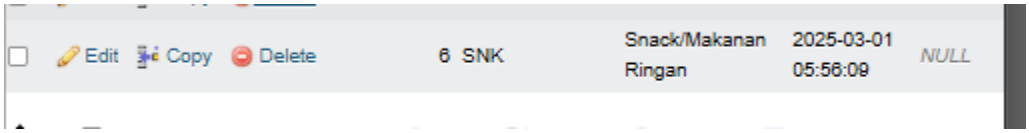
3. Selanjutnya, kita modifikasi file KategoriController untuk menambahkan 1 data ke table m_kategori

```php
class KategoriController extends Controller
{
    Codeium: Refactor | Explain | Generate Function Comment | X
    public function index()
    {
        $data = [
            'kategori_kode' => 'SNK',
            'kategori_nama' => 'Snack/Makanan Ringan',
            'created_at' => now()
        ];
        DB::table('m_kategori')->insert($data);
        return 'Insert Data 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



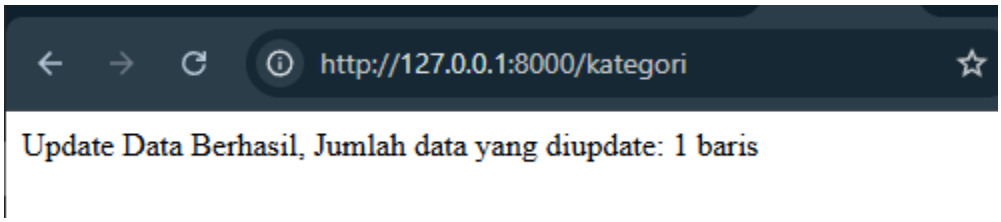
Insert Data Berhasil



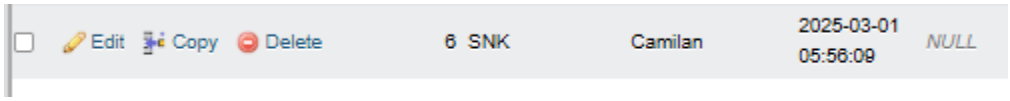
5. Selanjutnya, kita modifikasi lagi file KategoriController untuk meng-update data di table m_kategori seperti berikut

```php
// Update Data KAtegori
$row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
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



Update Data Berhasil, Jumlah data yang diupdate: 1 baris



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

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

http://127.0.0.1:8000/kategori

Hapus Data Berhasil, Jumlah data yang dihapus: 1 baris

Extra options

	kategori_id	kategori_kode	kategori_nama	created_at	updated_
Edit Copy Delete	1	makanan	Makanan	NULL	NULL
Edit Copy Delete	2	minuman	Minuman	NULL	NULL
Edit Copy Delete	3	snack	Snack	NULL	NULL
Edit Copy Delete	4	alat_tulis	Alat Tulis	NULL	NULL
Edit Copy Delete	5	elektronik	Elektronik	NULL	NULL

Check all *With selected:* Edit Copy Delete Export

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

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

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

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

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

Data Kategori Barang

ID

Kode Kategori	Nama Kategori	
1	makanan	Makanan
2	minuman	Minuman
3	snack	Snack
4	alat_tulis	Alat Tulis
5	elektronik	Elektronik

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

```
^C
C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS>php artisan make:model UserM
odel

  INFO  Model [C:\laragon\www\smt4\WebLanjut\Week3\PWL_POS\app\Models\UserModel.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

```
Codeium: Refactor | Explain
class UserModel extends Model
{
    use HasFactory;

    protected $table = 'm_users'; // mendefinisikan tabel yang akan digunakan
    protected $primaryKey = 'user_id'; // mendefinisikan primary key
}
```

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

```
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

```
Codeium: Refactor | Explain
class UserController extends Controller
{
    Codeium: Refactor | Explain | Generate Function Comment | X
    public function index()
    {
        // akses model UserModel
        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }
}
```

6. Kemudian kita buat view user.blade.php

```
<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>
```

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

```
{
    $data = [
        'username' => 'customer-1',
        'nama' => 'Pelanggan',
        'password' => Hash::make('12345'),
        'level_id' => 4
    ];
    UserModel::insert($data);
```

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
9	customer-1	Pelanggan	4

- Menambahkan data dengan username customer-1 dengan nama pelanggan id level 4

10. Kita modifikasi lagi file UserController menjadi seperti berikut

```
// Tambah Data user dengan Eloquent Model
$data = [
    'nama' => 'Pelanggan Pertama',
];
UserModel::where('username', 'customer-1')->update($data); //update data user
// akses model UserModel
```

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
9	customer-1	Pelanggan Pertama	4

- Mengupdate nama user customer-1 dengan pelanggan pertama

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

G. Penutup

Pertanyaan

1. Pada Praktikum 1 - Tahap 5, apakah fungsi dari APP_KEY pada file setting .env Laravel?
   - digunakan oleh Laravel untuk mengenkripsi data,
2. Pada Praktikum 1, bagaimana kita men-generate nilai untuk APP_KEY?
   - Php artisan key:generate
3. Pada Praktikum 2.1 - Tahap 1, secara default Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?
   - Ada 4
   - 2014_10_12_000000_create_users_table, Membuat tabel users untuk menyimpan data pengguna.
   - 2014_10_12_100000_create_password_reset_tokens_table, Membuat tabel password_reset_tokens untuk menyimpan token reset password
   - 2019_12_14_000001_create_personal_access_tokens_table, Membuat tabel personal_access_tokens untuk menyimpan token akses personal (biasanya digunakan untuk API authentication dengan Laravel Sanctum).
   - 2019_08_19_000000_create_failed_jobs_table, berfungsi untuk membuat tabel **failed_jobs**, yang digunakan untuk menyimpan informasi tentang **pekerjaan (jobs) yang gagal dieksekusi** dalam sistem **queue (antrian)** Laravel.
4. Secara default, file migrasi terdapat kode $table->timestamps();, apa tujuan/output dari fungsi tersebut?
   - Untuk membuatkan kolom create_at dan update_at secara default
5. Pada File Migrasi, terdapat fungsi $table->id(); Tipe data apa yang dihasilkan dari fungsi tersebut?
   - Memberikan id secara otomatis dengan tipe data BIGINT
6. Apa bedanya hasil migrasi pada table m_level, antara menggunakan $table->id(); dengan menggunakan $table->id('level_id'); ?
   - Untuk mengkustom nama pada judul kolom table $table->id(); (id), $table->id('level_id')

     level_id
7. Pada migration, Fungsi ->unique() digunakan untuk apa?
   - Agar tidak ada data yang memiliki kesamaan dengan data yang lain
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') ?
   - Karena pada kolom level_id pada table m_user merupakan foreignkey dari tabel m_level
9. Pada Praktikum 3 - Tahap 6, apa tujuan dari Class Hash? dan apa maksud dari kode program Hash::make('1234');?

- Untuk mengenkripsi password

10. Pada Praktikum 4 - Tahap 3/5/7, pada query builder terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?
    - Untuk melakukan insert data baru dengan parameter yang diinginkan
11. Pada Praktikum 6 - Tahap 3, apa tujuan penulisan kode protected $table = ‘m_user’; dan protected $primaryKey = ‘user_id’; ?
    - Digunakan dalam Model Eloquent Laravel untuk menyesuaikan nama tabel dan primary key yang digunakan oleh model
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (DB Façade / Query Builder / Eloquent ORM) ? jelaskan
    - DB Façade, karena eksekusi SQL langsung tanpa model. Cocok untuk saya yang masih pemula