PEMROGRAMAN WEB LANJUT

PERTEMUAN 3

MUHAMMAD IQBAL MAKMUR AL-MUNIRI

2241720099

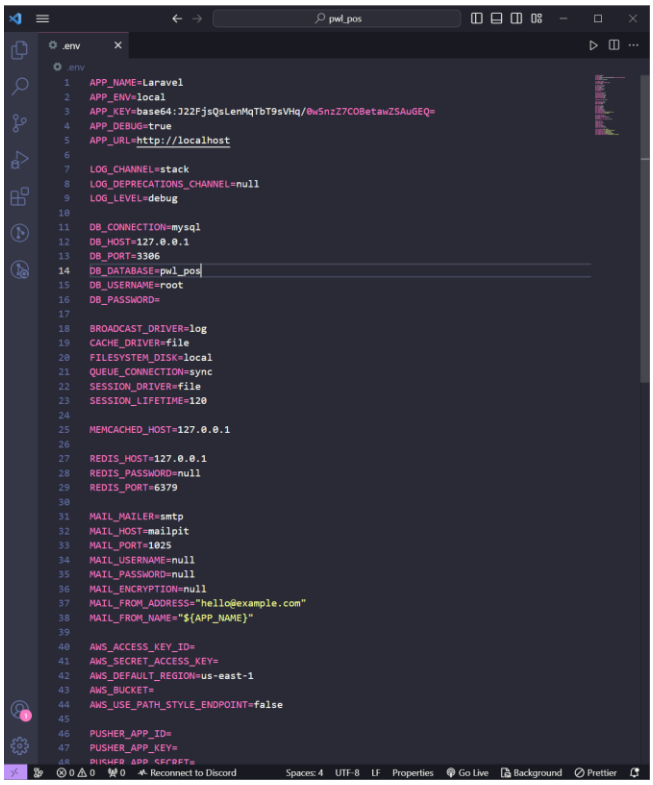
TEKNIK INFORMATIKA

TEKNOLOGI INFORMASI

LINK GITHUB : https://github.com/GalaAldebara/PemrogramanWebLanjut.git

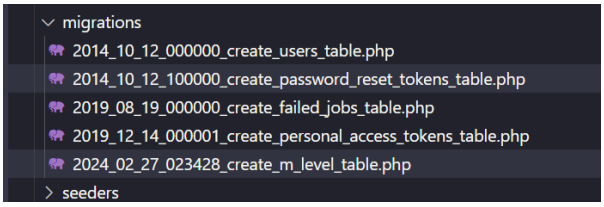
A. PENGATURAN DATABASE

Konfigurasi APP_KEY dan mengubah DB_DATABASE = pwl_pos

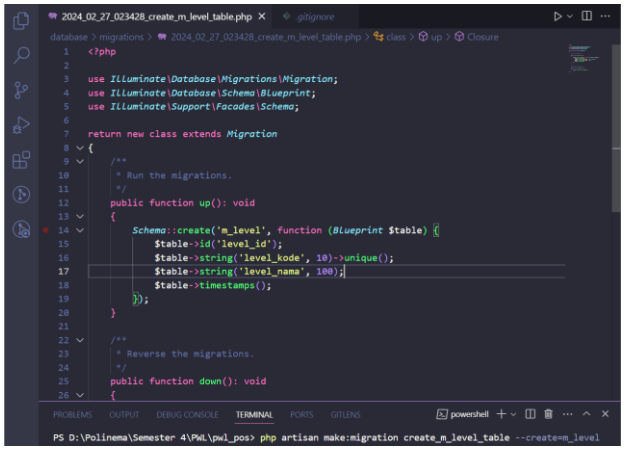


B. MIGRATION

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

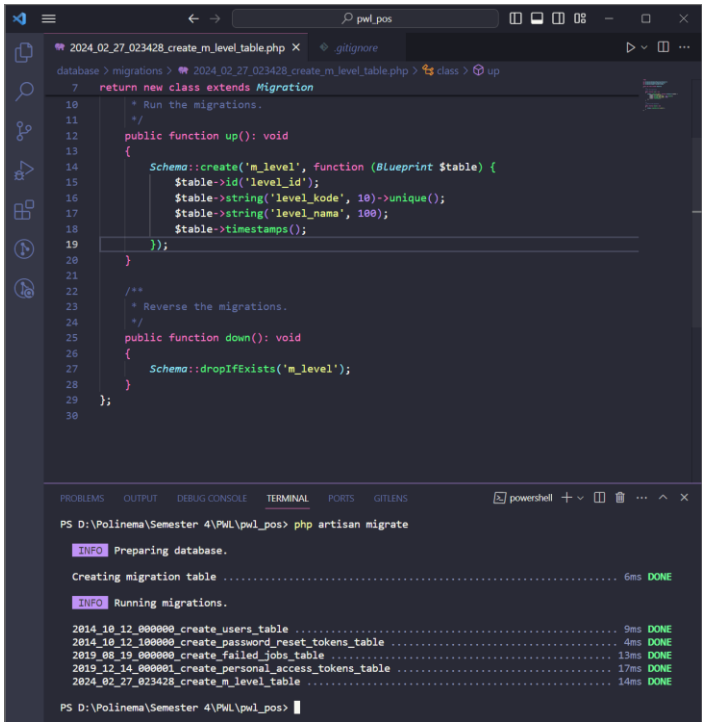


2. Kita abaikan dulu yang di kotak merah (jangan di hapus).
3. Kita buat file migrasi untuk table m_level dengan perintah

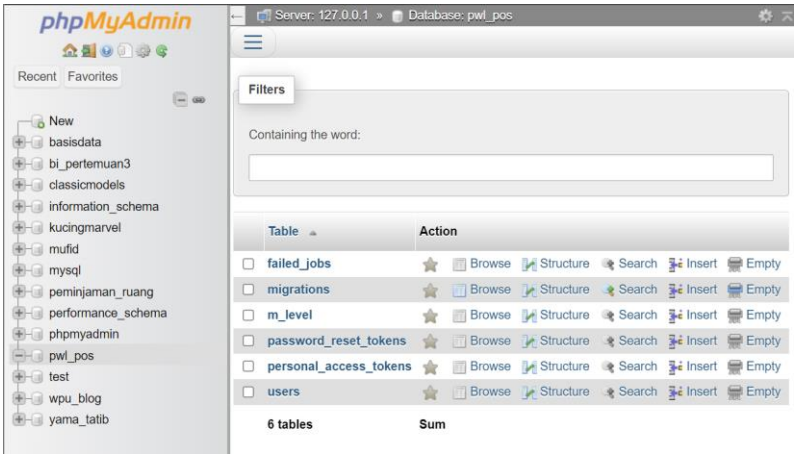


4. Kita perhatikan bagian yang di kotak merah, bagian tersebut yang akan kita modifikasi sesuai desain database yang sudah ada.

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

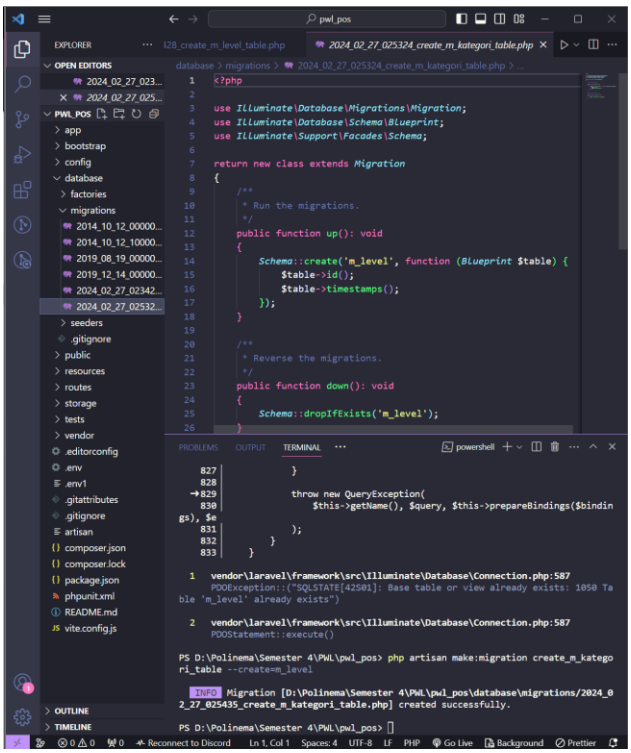


6. Kemudian kita cek di phpMyAdmin apakah table sudah ter-generate atau belum.



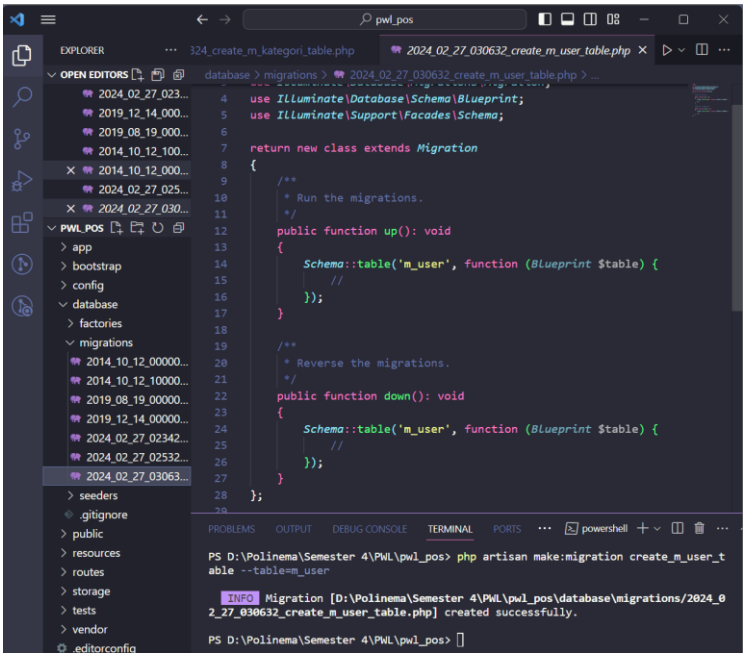
7. Ok, table sudah dibuat di database.

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



Praktikum 2.2

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



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

```
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. Amati apa yang terjadi pada database.

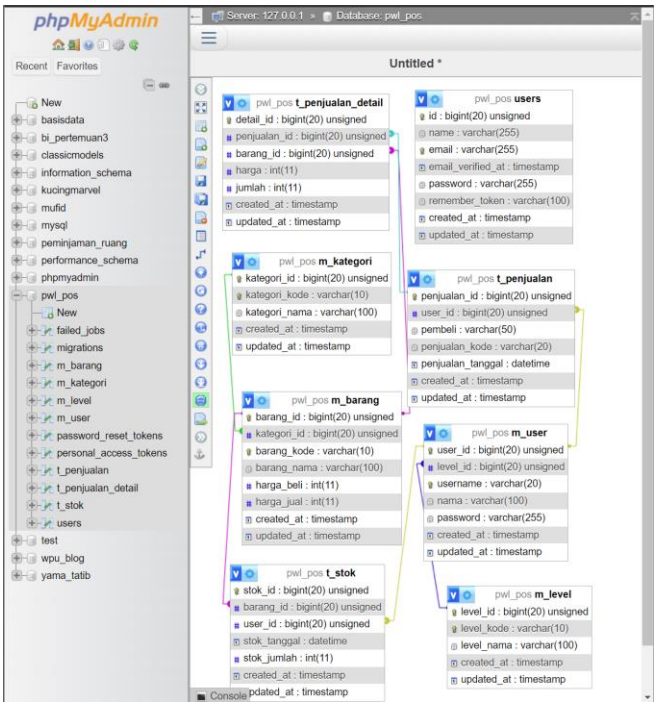
```
PS D:\Polinema\Semester 4\PWL\pwl_pos> php artisan migrate

  INFO  Running migrations.

  2024_02_27_031758_create_m_user_table ..................................... 39ms DONE
```

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

```
  INFO  Running migrations.

  2024_02_27_032457_create_m_barang_table ..................................... 6ms DONE
  2024_02_27_032525_create_t_penjualan_table ................................. 4ms DONE
  2024_02_27_032604_create_t_stok_table ...................................... 4ms DONE
  2024_02_27_032631_create_t_penjualan_detail_table .......................... 6ms DONE
```

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

```
2014_10_12_000000_create_users_table ..............................  14ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..............   3ms DONE
2019_08_19_000000_create_failed_jobs_table ........................  14ms DONE
2019_12_14_000001_create_personal_access_tokens_table .............  17ms DONE
2024_02_27_023428_create_m_level_table ............................  13ms DONE
2024_02_27_025324_create_m_kategori_table .........................  21ms DONE
2024_02_27_031758_create_m_user_table .............................  31ms DONE
2024_02_27_032457_create_m_barang_table ...........................  31ms DONE
2024_02_27_032525_create_t_penjualan_table ........................  27ms DONE
2024_02_27_032604_create_t_stok_table .............................  60ms DONE
2024_02_27_032631_create_t_penjualan_detail_table .................  43ms DONE
```



C. SEEDER

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

database > seeders > LevelSeeder.php > PHP Intelephense > LevelSeeder

```php
<?php

namespace Database\Seeders;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;

class LevelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        //
    }
}
```

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

```php
class LevelSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    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.

```
PS D:\Polinema\Semester 4\PWL\pwl_pos> php artisan db:seed --class=LevelSeeder

  INFO  Seeding database.

PS D:\Polinema\Semester 4\PWL\pwl_pos>
```

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	Staff/Kasir	NULL	NULL

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

```
PS D:\Polinema\Semester 4\PWL\pwl_pos> php artisan make:seeder UserSeeder

  INFO  Seeder [D:\Polinema\Semester 4\PWL\pwl_pos\database\seeders\UserSeeder.php] created successfull
y.
```

6. Modifikasi file class UserSeeder seperti berikut.

```php
class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [

            [
                'user_id' => 1,
                'level_id' => 1,
                'username' => 'administrator',
                '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. Jalankah perintah untuk mengeksekusi class UserSeeder

```
PS D:\Polinema\Semester 4\Pemrograman Web Lanjut\pertemuan2> php artisan db:seed
--class=UserSeeder
```

8. Perhatikan hasil seeder pada table m_user

			user_id	level_id	username	nama	password	created_at	updated_at
Edit	Copy	Delete	1	1	administrator	Administrator	$2y$12$kiN2eq603gy6FU7eZZmb9uXhrI2KF9KOAub4/mTfiLl...	NULL	NULL
Edit	Copy	Delete	2	2	manager	Manager	$2y$12$c3OQ7Dze0v9BescArkVhsefo/lBzhwyS.cWPbQCGEzC...	NULL	NULL
Edit	Copy	Delete	3	3	staff	Staff/Kasir	$2y$12$dybEoXfWNJyLDEEZCauqve.rNLmSLzsMWoE1cVXKWQJ...	NULL	NULL

9. 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

←T→	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
Edit Copy Delete	1	Lenovo	laptop	NULL	NULL
Edit Copy Delete	2	Redmi	Redmi Note 13 Pro	NULL	NULL
Edit Copy Delete	3	Toshiba	Televisi	NULL	NULL
Edit Copy Delete	4	Samsung	Air Purifier	NULL	NULL
Edit Copy Delete	5	Ferrari	Mobil	NULL	NULL

←T→	barang_id	kategori_id	barang_kode	barang_nama	harga_beli	harga_jual	created_at	updated_at
Edit Copy Delete	1	1	Lenovo	Lenovo Legion 5 Pro	20000000	15000000	NULL	NULL
Edit Copy Delete	2	1	HP	HP Victus 16	20000000	15000000	NULL	NULL
Edit Copy Delete	3	2	Apple	Iphone 15 Promax	20000000	15000000	NULL	NULL
Edit Copy Delete	4	2	Redmi	Redmi Note 13	20000000	15000000	NULL	NULL
Edit Copy Delete	5	3	Toshiba	Canvio Basic 1TB	20000000	15000000	NULL	NULL
Edit Copy Delete	6	3	LG	Mesin Cuci	20000000	15000000	NULL	NULL
Edit Copy Delete	7	4	Samsung	Galaxy S23 Ultra	20000000	15000000	NULL	NULL
Edit Copy Delete	8	4	Poco	Poco X6 Pro	20000000	15000000	NULL	NULL
Edit Copy Delete	9	5	Honda	Civic Turbo	20000000	15000000	NULL	NULL
Edit Copy Delete	10	5	Lambo	Aventador	20000000	15000000	NULL	NULL

←T→	stok_id	barang_id	user_id	stok_tanggal	stok_jumlah	created_at	updated_at
Edit Copy Delete	1	1	1	2005-01-17 00:00:00	10	NULL	NULL
Edit Copy Delete	2	2	1	2005-01-17 00:00:00	10	NULL	NULL
Edit Copy Delete	3	3	1	2005-01-17 00:00:00	10	NULL	NULL
Edit Copy Delete	4	4	1	2005-01-17 00:00:00	10	NULL	NULL
Edit Copy Delete	5	5	2	2005-01-17 00:00:00	10	NULL	NULL
Edit Copy Delete	6	6	2	2005-01-17 00:00:00	10	NULL	NULL
Edit Copy Delete	7	7	2	2005-01-17 00:00:00	10	NULL	NULL
Edit Copy Delete	8	8	3	2005-01-17 00:00:00	10	NULL	NULL
Edit Copy Delete	9	9	3	2005-01-17 00:00:00	10	NULL	NULL
Edit Copy Delete	10	10	3	2005-01-17 00:00:00	10	NULL	NULL

←T→	penjualan_id	user_id	pembeli	penjualan_kode	penjualan_tanggal	created_at	updated_at
Edit Copy Delete	1	3	Arya Bagus	KODE1	2005-01-17 00:00:00	NULL	NULL
Edit Copy Delete	2	3	Yusriyah Firjatullah	KODE2	2005-01-17 00:00:00	NULL	NULL
Edit Copy Delete	3	3	Dido Imam	KODE3	2005-01-17 00:00:00	NULL	NULL
Edit Copy Delete	4	2	Atha	KODE4	2005-01-17 00:00:00	NULL	NULL
Edit Copy Delete	5	2	Naufal	KODE5	2005-01-17 00:00:00	NULL	NULL
Edit Copy Delete	6	2	Iqbal	KODE6	2005-01-17 00:00:00	NULL	NULL
Edit Copy Delete	7	1	Makmur	KODE7	2005-01-17 00:00:00	NULL	NULL
Edit Copy Delete	8	3	Royhan	KODE8	2005-01-17 00:00:00	NULL	NULL
Edit Copy Delete	9	3	Chevalerie	KODE9	2005-01-17 00:00:00	NULL	NULL
Edit Copy Delete	10	3	Akhyar	KODE10	2005-01-17 00:00:00	NULL	NULL

←T→	detail_id	penjualan_id	barang_id	harga	jumlah	created_at	updated_at
Edit Copy Delete	1	1	1	10000	5	NULL	NULL
Edit Copy Delete	2	1	2	20000	5	NULL	NULL
Edit Copy Delete	3	1	3	30000	5	NULL	NULL
Edit Copy Delete	4	2	1	10000	5	NULL	NULL
Edit Copy Delete	5	2	2	10000	5	NULL	NULL
Edit Copy Delete	6	2	3	10000	5	NULL	NULL
Edit Copy Delete	7	3	1	10000	5	NULL	NULL
Edit Copy Delete	8	3	2	10000	5	NULL	NULL
Edit Copy Delete	9	3	3	10000	5	NULL	NULL
Edit Copy Delete	10	4	1	10000	5	NULL	NULL
Edit Copy Delete	11	4	2	20000	5	NULL	NULL
Edit Copy Delete	12	4	3	30000	5	NULL	NULL
Edit Copy Delete	13	5	1	10000	5	NULL	NULL
Edit Copy Delete	14	5	2	20000	5	NULL	NULL
Edit Copy Delete	15	5	3	30000	5	NULL	NULL
Edit Copy Delete	16	6	1	10000	5	NULL	NULL
Edit Copy Delete	17	6	2	20000	5	NULL	NULL
Edit Copy Delete	18	6	3	30000	5	NULL	NULL
Edit Copy Delete	19	7	1	10000	5	NULL	NULL
Edit Copy Delete	20	7	2	20000	5	NULL	NULL
Edit Copy Delete	21	7	3	30000	5	NULL	NULL
Edit Copy Delete	22	8	1	10000	5	NULL	NULL
Edit Copy Delete	23	8	2	20000	5	NULL	NULL
Edit Copy Delete	24	8	3	30000	5	NULL	NULL
Edit Copy Delete	25	9	1	10000	5	NULL	NULL

D. Implementasi DB Façade

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

```
PS D:\Polinema\Semester 4\PWL\pwl_pos> php artisan make:controller LevelController

  INFO  Controller [D:\Polinema\Semester 4\PWL\pwl_pos\app\Http\Controllers\LevelController.php] create
d successfully.

PS D:\Polinema\Semester 4\PWL\pwl_pos>
```

2. Kita modifikasi dulu untuk routingnya, ada di PWL_POS/routes/web.php

```
17  Route::get('/', function () {
18      return view('welcome');
19  });
20
21  Route::get('/level', LevelController::class, 'index');
22
```

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()
        ]);
    }
}
```

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

	level_id	level_kode	level_nama	created_at	updated_
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
Edit Copy Delete	4	CUS	Pelanggan	2024-03-10 09:02:40	NULL

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

```php
class LevelController extends Controller
{
    public function index()
    {
        // DB::insert('insert into m_level (level_kode, level_nama, created_at) value
        //     'CUS', 'Pelanggan', now()
        // ]);

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

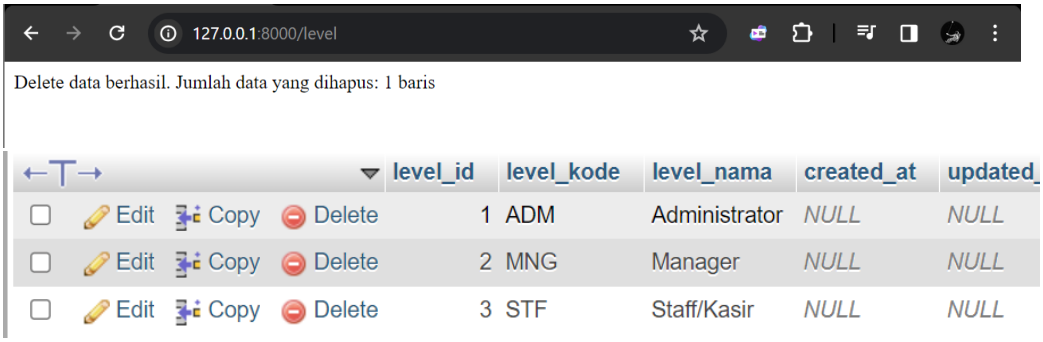
6. Kita coba jalankan di browser dengan url localhost/PWL_POS/public/level lagi dan amati apa yang terjadi pada table m_level di database, screenshot perubahan yang ada pada table m_level

127.0.0.1:8000/level

Update data berhasil. Jumlah data yang diupdate: 1 baris

	level_id	level_kode	level_nama	created_at	updated_
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
Edit Copy Delete	4	CUS	Customer	2024-03-10 09:02:40	NULL

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



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

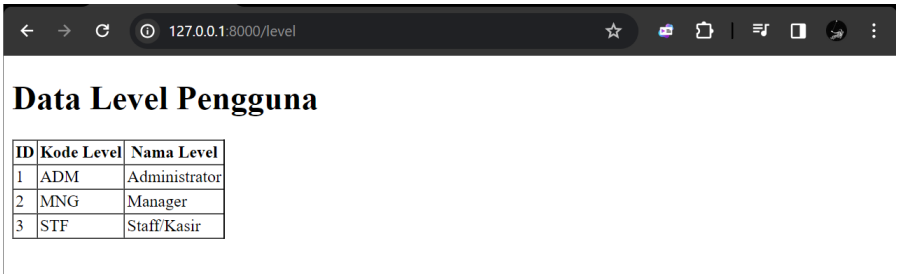
```php
class LevelController extends Controller
{
    public function index()
    {
        // DB::insert('insert into m_level (level_kode, level_nama, created_at) values
        //     'CUS', 'Pelanggan', now()
        // ]);

        // $row = DB::update('update m_level set level_nama = ? where level_kode = ?',
        // return 'Update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris

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

        $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

resources > views > level.blade.php > html > body > table

```php
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <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



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

E. Implementasi Query Builder

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

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS        powershell

PS D:\Polinema\Semester 4\PWL\pwl_pos> php artisan make:controller KategoriController

  INFO  Controller [D:\Polinema\Semester 4\PWL\pwl_pos\app\Http\Controllers\KategoriController.php] cre
ated successfully.

PS D:\Polinema\Semester 4\PWL\pwl_pos>
```

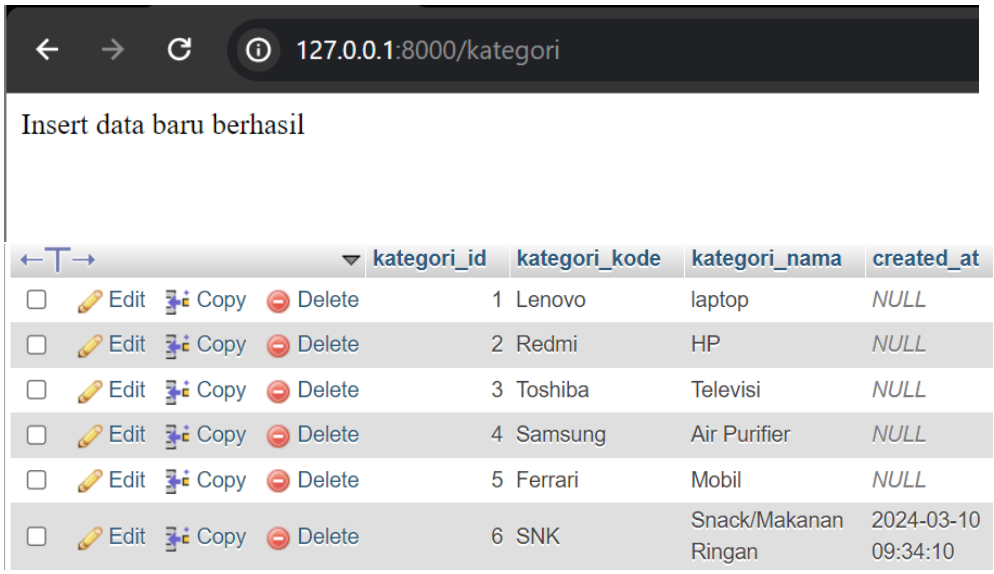
2. Kita modifikasi dulu untuk routing-nya, ada di PWL_POS/routes/web.php

```
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

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

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
Edit Copy Delete	1	Lenovo	laptop	NULL
Edit Copy Delete	2	Redmi	HP	NULL
Edit Copy Delete	3	Toshiba	Televisi	NULL
Edit Copy Delete	4	Samsung	Air Purifier	NULL
Edit Copy Delete	5	Ferrari	Mobil	NULL
Edit Copy Delete	6	SNK	Snack/Makanan Ringan	2024-03-10 09:34:10

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

```
9   class KategoriController extends Controller
10  {
11      public function index()
12      {
13          // $data = [
14          //     'kategori_kode' => 'SNK',
15          //     'kategori_nama' => 'Snack/Makanan Ringan',
16          //     'created_at' => now()
17          // ];
18          // DB::table('m_kategori')->insert($data);
19          // return 'Insert data baru berhasil';
20
21          $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kateg
22          return 'update data berhasil. Jumlah data yang diupdate: ' . $row . ' baris';
23      }
24  }
25
```

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	Lenovo	laptop	NULL	NULL
Edit Copy Delete	2	Redmi	HP	NULL	NULL
Edit Copy Delete	3	Toshiba	Televisi	NULL	NULL
Edit Copy Delete	4	Samsung	Air Purifier	NULL	NULL
Edit Copy Delete	5	Ferrari	Mobil	NULL	NULL
Edit Copy Delete	6	SNK	Camilan	2024-03-10 09:34:10	NULL

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

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

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

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

8. 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

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

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

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

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

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

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

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

11. Silahkan dicoba pada browser dan amati apa yang terjadi

127.0.0.1:8000/kategori

Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	Lenovo	laptop
2	Redmi	HP
3	Toshiba	Televisi
4	Samsung	Air Purifier
5	Ferrari	Mobil

F. Implementasi Eloquent ORM

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

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLENS        powershell

PS D:\Polinema\Semester 4\PWL\pwl_pos> php artisan make:model UserModel

  INFO  Model [D:\Polinema\Semester 4\PWL\pwl_pos\app\Models\UserModel.php] created successfully.

PS D:\Polinema\Semester 4\PWL\pwl_pos>
```

2. Kita buka file UserModel.php dan modifikasi seperti berikut

app > Models > UserModel.php > PHP Intelephense > UserModel

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class UserModel extends Model
{
    use HasFactory;

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

```

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

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

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

```

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

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

```php
<?php

namespace App\Http\Controllers;

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

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

```

5. Kemudian kita buat view user.blade.php

resources > views > user.blade.php > html > body

```php
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <meta http-equiv="X-UA-Compatible" content="ie=edge">
        <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>
```

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

127.0.0.1:8000/user

Data User

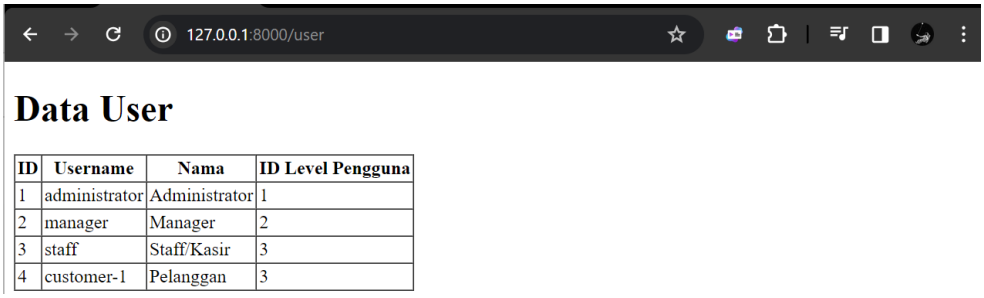
ID	Username	Nama	ID Level Pengguna
1	administrator	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3

7. Setelah itu, kita modifikasi lagi file UserController

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

```php
<?php

namespace App\Http\Controllers;

use App\Models\UserModel;
use Illuminate\Http\Request;
use Illuminate\Routing\Controller;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function index()
    {
        $data = [
            'level_id' => 3,
            'username' => 'customer-1',
            'nama' => 'Pelanggan',
            'password' => Hash::make('12345')
        ];
        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }
}
```

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

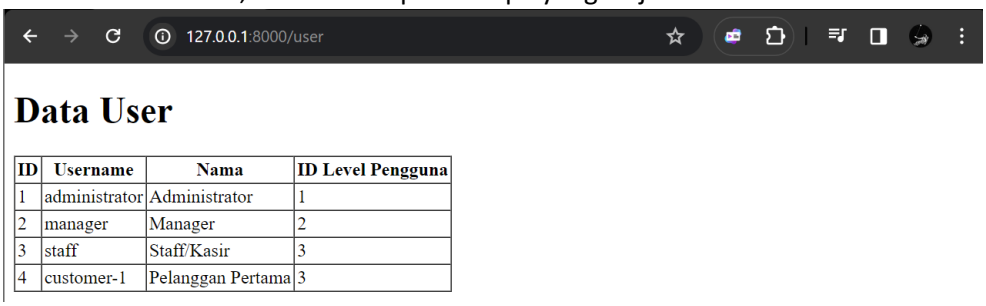


ID	Username	Nama	ID Level Pengguna
1	administrator	Administrator	1
2	manager	Manager	2
3	staff	Staff/Kasir	3
4	customer-1	Pelanggan	3

9. Kita modifikasi lagi file UserController menjadi seperti berikut

```php
class UserController extends Controller
{
    public function index()
    {
        // $data = [
        //     'level_id' => 3,
        //     'username' => 'customer-1',
        //     'nama' => 'Pelanggan',
        //     'password' => Hash::make('12345')
        // ];
        // UserModel::insert($data);

        $data = [
            'nama' => 'Pelanggan Pertama',
        ];
        UserModel::where('username', 'customer-1')->update($data);

        $user = UserModel::all();
        return view('user', ['data' => $user]);
    }
}
```

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



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

Penutup - Pertanyaan

1. **PadaPraktikum 1- Tahap 5, apakah fungsi dari APP_KEY pada file setting .env Laravel?**
   Jawab:
   Berfungsi sebagai kunci enkripsi yang digunakan oleh aplikasi Laravel untuk mengamankan data yang disimpan.

2. **PadaPraktikum 1, bagaimana kita men-generate nilai untuk APP_KEY?**
   Jawab:
   Menggunakan php artisan

3. **PadaPraktikum 2.1- Tahap 1, secara default Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?**
   Jawab:
   Terdapat 4 file migrasi yaitu:
   - Create_user_table
   - Create_password_reset_tokens_table
   - Create_failed_jobs_table
   - Create_personal_access_token_table

4. **Secara default, file migrasi terdapat kode $table->timestamps();, apa tujuan/output dari fungsi tersebut?**
   Jawab:
   Kode $table->timestamps() pada Laravel digunakan untuk menambahkan dua kolom ke dalam tabel yaitu created_at & updated_at.

5. **PadaFile Migrasi, terdapat fungsi $table->id(); Tipe data apa yang dihasilkan dari fungsi tersebut?**
   Jawab:
   Fungsi $table->id(); pada file migrasi digunakan untuk membuat kolom dengan tipe data UNSIGNED BIG INTEGERAUTO_INCREMENTPRIMARYKEY.

6. **Apabedanya hasil migrasi pada table m_level, antara menggunakan $table->id(); dengan menggunakan $table->id('level_id'); ?**
   Jawab:
   jika menggunakan $table->id() maka kolom akan secara otomatis dinamai menjadi id
   jika menggunakan $table->id(‘level_id) maka kolom akan dinamai sesuai dengan argumen yang diberikan, yaitu level_id

7. **Pada migration, Fungsi->unique() digunakan untuk apa?**
   Jawab:
   Digunakan untuk menentukan atau mendeklarasikan unique constraint pada kolom tertentu

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->unsignedBigInteger('level_id') digunakan untuk membuat kolom foreign dari tabel level sedangkan kolom level_id pada tabel m_level menggunakan $tabel->id('level_id') karena itu untuk memberi nama primary key pada tabel m_level yaitu level_id

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 melakukan enkripsi password, berfungsi untuk meningkatkan keamanan dengan menyimpan password dalam bentuk yang tidak dapat dibaca atau hash.

10. **Pada Praktikum 4- Tahap 3/5/7, pada db facade terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?**
    Jawab:
    Tanda tanya merupakan parameter bind placeholder. Biasanya disebut sebagai prepared statement atau binding parameter.

11. **Pada Praktikum 6- Tahap 3, apa tujuan penulisan kode protected $table = 'm_user'; dan protected $primaryKey = 'user_id'; ?**
    Jawab:
    Bertujuan untuk menentukan nama table dan primary key yang terkait dengan model tersebut.

12. **Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (DB Façade / Query Builder / Eloquent ORM) ? Jelaskan!**
    Jawab:
    Lebih mudah menggunakan Eloquent ORM karena memudahkan pengelolaan relasi antar tabel. Ibarat DB FAÇADE adalah native dan Eloquent ORM adalah Laravel.