

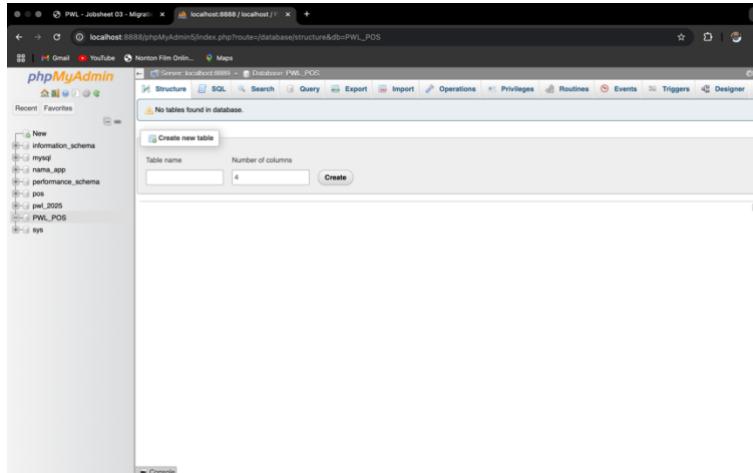
**Advanced Web Programming – Migration, Seeder, DB Façade,
Query Builder and Eloquent ORM**
Week 3



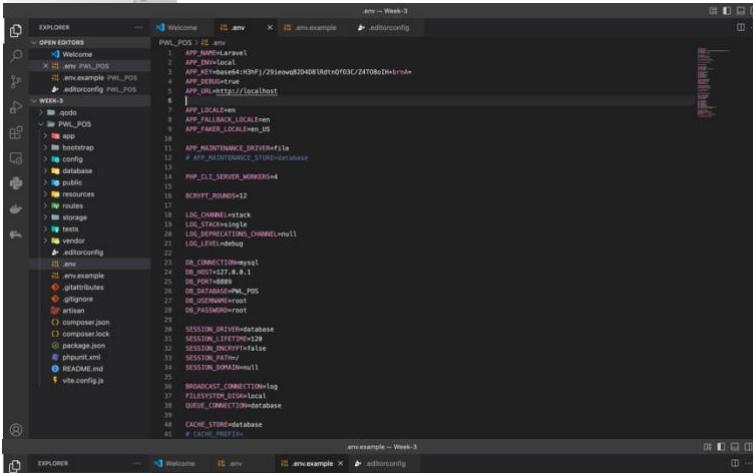
Baskoro Seno Aji
2341720063 / 05 / TI-2I
Github Link : [Github](#)
Polytechnic State of Malang

Pengaturan Database

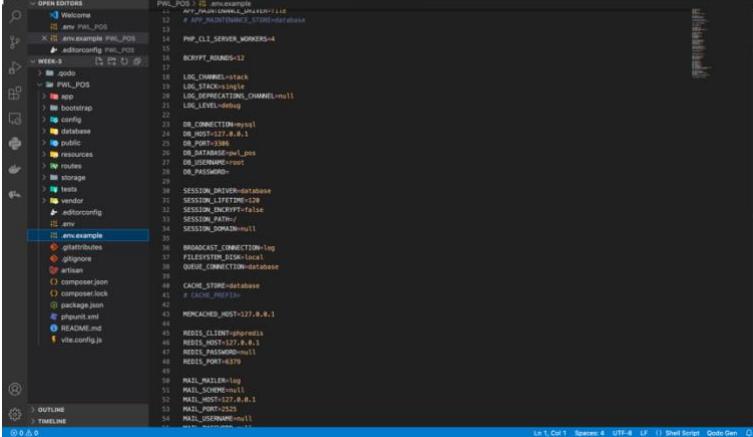
Praktikum 1 :



The screenshot shows the phpMyAdmin interface for the 'PWL_POS' database. A modal window titled 'Create new table' is open, prompting for a 'Table name' (left empty) and 'Number of columns' (set to 4). Below the table creation dialog, the database structure is visible, showing various schemas like 'information_schema', 'mysql', 'name_app', 'performance_schema', 'pwl_2025', and 'PWL_POS'. The 'sys' schema is also present.



The screenshot shows a code editor displaying the contents of the '.env.example' file. The file contains environment variables for a Laravel application, including database settings (DB_CONNECTION=mysql, DB_HOST=127.0.0.1, DB_PORT=3306, DB_DATABASE=pwa_pos, DB_USERNAME=root, DB_PASSWORD=root), session settings (SESSION_DRIVER=file, SESSION_LIFETIME=120, SESSION_REFRESH_EACH_REQUEST=true), log levels (LOG_CHANNEL=stack, LOG_LEVEL=debug), and other system configurations (APP_NAME=laravel, APP_ENV=local, APP_KEY=base64:K0Mj/9iew0lD0408U6txyf03C/2AT0bDHe+rate, APP_DEBUG=true, APP_URL=http://localhost:8000). It also includes broadcast connection settings (BROADCAST_CONNECTION=log, BROADCAST_DRIVER=log) and queue connection settings (QUEUE_CONNECTION=database).



The screenshot shows a code editor displaying the contents of the '.env' file. This file is identical to the '.env.example' file, containing the same environment variables for the Laravel application, including database, session, log, and broadcast settings.

- A. This section we're creating the databases first on the PhpMyAdmin then implement the database into the PWL_POS folder where it already installed by the Laravel and recheck it into the VSCode by specifically see the DB PORT, APP KEY, DB DATABASE, and change the DB PASSWORD.

MIGRATION

Praktikum 2.1 – Pembuatan File Migrasi Tanpa Relasi

PWL_POS > database > migrations > 2025_02_25_045735_create_m_level_table.php > class > up > Closure

```
1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 Qodo Gen: Options | Test this class
8 return new class extends Migration
9 {
10     /**
11      * Run the migrations.
12      */
13     public function up(): void
14     {
15         Schema::create('m_level', function (Blueprint $table): void {
16             $table->id(column: 'level_id');
17             $table->String(column: 'level_kode', length: 10)->unique();
18             $table->String(column: 'level_nama', length: 100);
19             $table->timestamps();
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      */
26     public function down(): void
27     {
28         Schema::dropIfExists('m_level');
29     }
30 }
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  zsh

● macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan migrate
  INFO Running migrations.
    2025_02_25_045735_create_m_level_table .....
○ macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % █
```

A. in this section we're implementing the **MIGRATIONS** for table who doesn't have the foreign key and only have the primary key or doesn't have any relations with the other table where we are using two main syntax sentences to create it, we use the :

- php artisan make:migration create_**name**_table --create=**name** (create the migrations)
- php artisan migrate (to run the migrations)

this MIGRATIONS feature do really help us a lot because of the simplicity that help to manage the database management thru the code and it's already connected to the PhpMyAdmin DB

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

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh - PWL_POS
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan migrate
INFO Running migrations.
2025_02_25_045735_create_m_level_table .....
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:migration create_m_kategori_table --create=m_kategori
INFO Migration [database/migrations/2025_02_25_050317_create_m_kategori_table.php] created successfully.
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

```

```

2025_02_25_050317_create_m_kategori_table.php - Week-3
PWL_POS > database > migrations > 2025_02_25_050317_create_m_kategori_table.php > class > up > Closure
... 2025_02_25_050317_create_m_kategori_table.php x D> B> I> C> E> F> X>
EXPLORER ... .000002_create_jobs_table.php 2025_02_25_045735_create_m_level_table.php 2025_02_25_050317_create_m_kategori_table.php x D> B> I> C> E> F> X>
OPEN EDITORS
  Welcome
  .env PWL_POS
  .env.example PWL_POS
  0001_01_01_000002_create_jobs_t...
  2025_02_25_045735_create_m_leve...
  2025_02_25_050317_create_m_kat...
  .editorconfig PWL_POS
WEEK-3
  .odo
  PWL_POS
  app
  bootstrap
  config
  database
    factories
    migrations
      0001_01_01_000000_create_use...
      0001_01_01_000001_create_cac...
      0001_01_01_000002_create_job...
      2025_02_25_045735_create_m...
      2025_02_25_050317_create_m_k...
  seeders
  .gitignore
  public
  resources
  routes
  storage
  tests
  vendor
    .editorconfig
    .env
    .env.example
    .gitattributes
    .gitignore
  OUTLINE
  TIMELINE
  0 0
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
macbookpro[Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan migrate
INFO: Running migrations.
2025_02_25_050317_create_m_kategori_table ...
macbookpro[Baskoro-Seno-Aji-Macbook-Pro PWL_POS % 36.40ms DONE
Ln 16, Col 28 Spaces: 4 UTF-8 LF () PHP 8.4.1: 8.2 Qodo Gen

```

- B. by implementing the same syntax sentences we've created the m_kategori and insert all the criteria that the table have.

Praktikum 2.2 – Pembuatan File Migrasi Dengan Relasi

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
zsh - PWL_POS
macbookpro[Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:migration create_m_user_table --create=m_user
INFO: Migration [database/migrations/2025_02_25_051030_create_m_user_table.php] created successfully.
macbookpro[Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

```

The screenshot displays a developer's environment with multiple windows open:

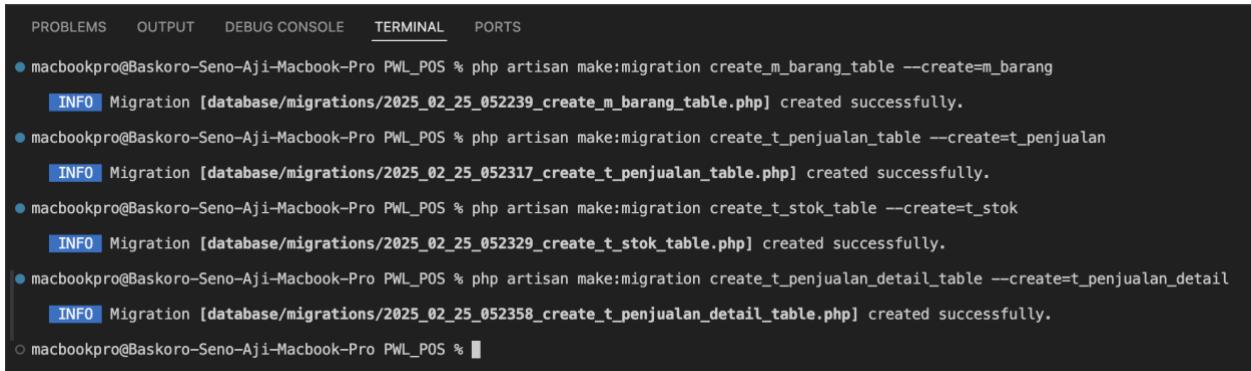
- Code Editor:** Shows a PHP migration file for the 'm_level' table. The code defines a migration class that runs migrations, creates the 'm_level' table with columns for id, level_id, username, nama, password, and timestamps, and adds a foreign key constraint linking 'level_id' to 'm_level.level_id'.
- Terminal:** A zsh shell window showing the command `php artisan migrate` being run, with the output indicating "Nothing to migrate".
- Browser:** A browser window showing the MySQL 'migrations' table via phpMyAdmin. The table contains six rows, each representing a migration step with its ID, migration name, and batch number (all in batch 1).

ID	Migration	Batch
1	20001_01_01_000000_create_users_table	1
2	20001_01_01_000001_create_cache_table	1
3	3 20001_01_01_000002_create_jobs_table	1
4	4 2025_02_25_045735_create_m_level_table	1
5	5 2025_02_25_050317_create_m_kategori_table	1
6	6 2025_02_25_051030_create_m_user_table	1

- C. In this section we're implementing the **MIGRATIONS** for the table that have the foreign key of the other table that implement the primary key and have the relations with the

other table, everything is the same but things that make it different only the criteria or code that implement the foreign key

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



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

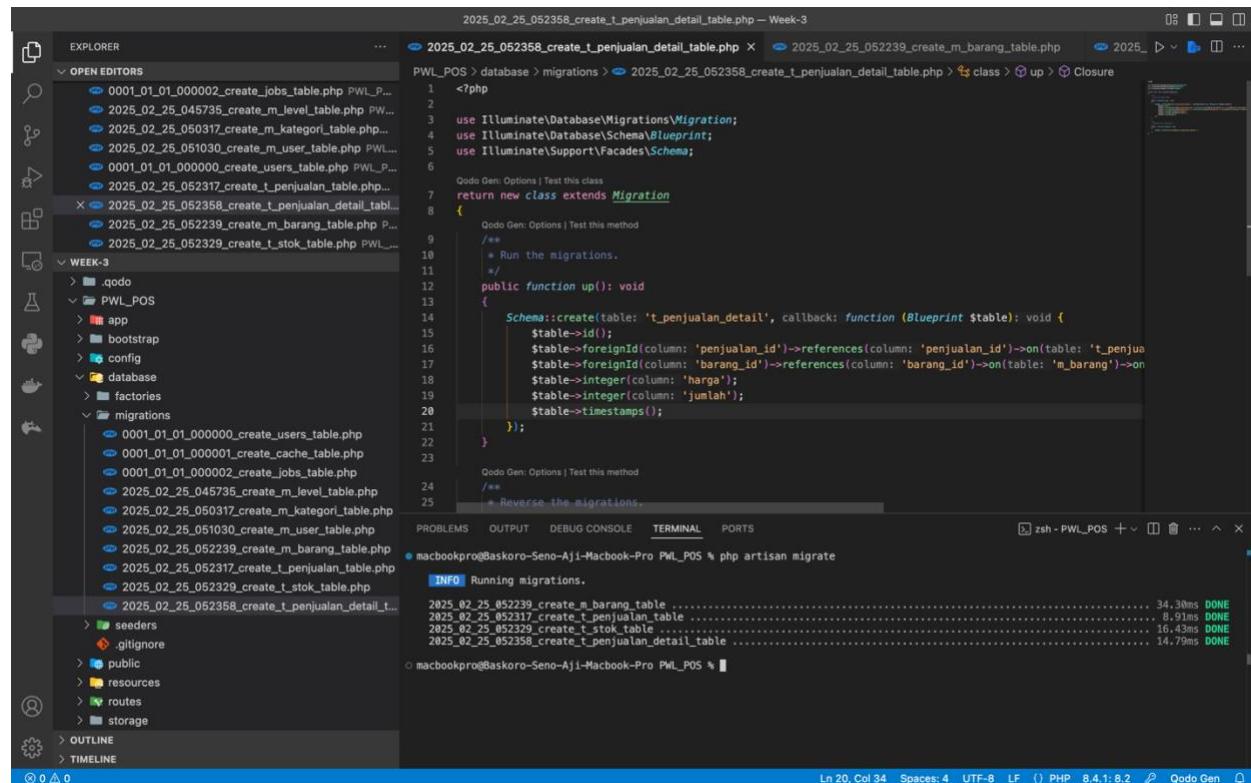
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:migration create_m_barang_table --create=m_barang
INFO | Migration [database/migrations/2025_02_25_052239_create_m_barang_table.php] created successfully.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:migration create_t_penjualan_table --create=t_penjualan
INFO | Migration [database/migrations/2025_02_25_052317_create_t_penjualan_table.php] created successfully.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:migration create_t_stok_table --create=t_stok
INFO | Migration [database/migrations/2025_02_25_052329_create_t_stok_table.php] created successfully.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:migration create_t_penjualan_detail_table --create=t_penjualan_detail
INFO | Migration [database/migrations/2025_02_25_052358_create_t_penjualan_detail_table.php] created successfully.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %
```



```
2025_02_25_052358_create_t_penjualan_detail_table.php - Week-3
2025_02_25_052358_create_t_penjualan_detail_table.php X 2025_02_25_052239_create_m_barang_table.php 2025_02_25_052317_create_t_penjualan_table.php ... Closure

PWL_POS > database > migrations > 2025_02_25_052358_create_t_penjualan_detail_table.php > class up > up > Closure

1 <?php
2
3 use Illuminate\Database\Migrations\Migration;
4 use Illuminate\Database\Schema\Blueprint;
5 use Illuminate\Support\Facades\Schema;
6
7 Qodo Gen: Options | Test this class
8 return new class extends Migration
9 {
10     /**
11      * Run the migrations.
12      */
13     public function up(): void
14     {
15         Schema::create('t_penjualan_detail', function (Blueprint $table) {
16             $table->id();
17             $table->foreignId('penjualan_id')->references('penjualan_id')->on('table: t_penjualan');
18             $table->foreignId('barang_id')->references('barang_id')->on('table: m_barang')->on('table: t_stok');
19             $table->integer('harga');
20             $table->integer('jumlah');
21             $table->timestamps();
22         });
23     }
24     /**
25      * Reverse the migrations.
26     */
27 }
```

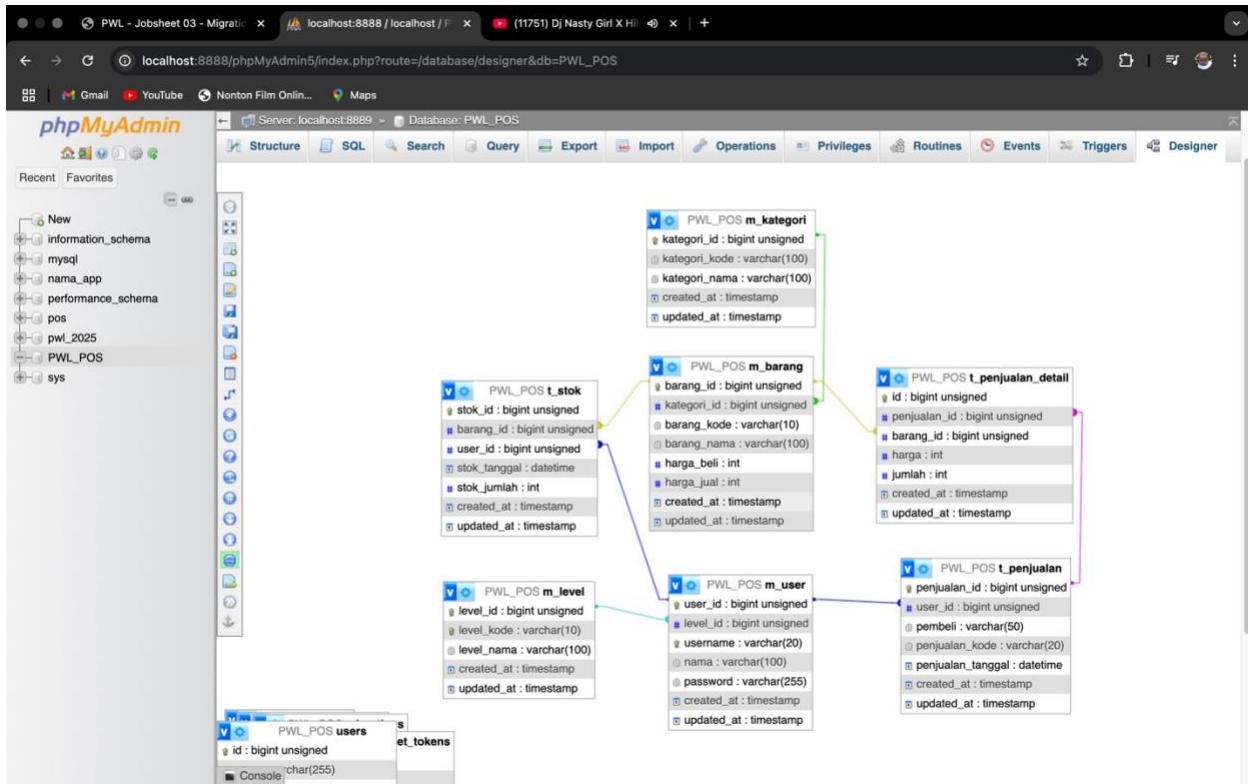
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan migrate
INFO | Running migrations.

2025_02_25_052239_create_m_barang_table ..... 34.30ms DONE
2025_02_25_052317_create_t_penjualan_table ..... 8.91ms DONE
2025_02_25_052329_create_t_stok_table ..... 16.43ms DONE
2025_02_25_052358_create_t_penjualan_detail_table ..... 14.79ms DONE
```

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

Ln 20, Col 34 Spaces: 4 UTF-8 LF PHP 8.4.1: 8.2 Qodo Gen



- D. This part we're doing the same but for an instance we're also checking it on the PhpMyAdmin Designer page that every table that have the relations with the main table or table master are connected well, this designer picture have proven that we're implementing it well.

SEEDER

Praktikum 3 – Membuat File Seeder

- Seeder on **M_LEVEL**

LevelSeeder.php — Week-3

```

LevelSeeder.php 2 > 0001_01_01_000000_create_users_table.php > 2025_02_25_052317_create_t_penjualan_table.php > 2025_02_25_052358_create_t_penjualan_detail_table.php > run

1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7
8 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class
9 class LevelSeeder extends Seeder
{
    Qodo Gen: Options | Test this method
10 /**
11 * Run the database seeds.
12 */
13 references | 0 overrides | Qodo Gen: Options | Test this method
14 public function run(): void
15 {
    $data = [
        ['level_id' => 1, 'level_kode' => 'ADM', 'level_nama' => 'Administrator'],
        ['level_id' => 2, 'level_kode' => 'MNG', 'level_nama' => 'Manager'],
        ['level_id' => 3, 'level_kode' => 'STF', 'level_nama' => 'Staff/Kasir']
    ];
18 DB::table('m_level')->insert($data);
19
20 }
21
22 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:seeder LevelSeeder

INFO Seeder [database/seeders/LevelSeeder.php] created successfully.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

LN 20, Col 45 Spaces: 4 UTF-8 LF () PHP 8.4.1: 8.2 Qodo Gen

LevelSeeder.php — Week-3

```

LevelSeeder.php X 0001_01_01_000000_create_users_table.php 2025_02_25_052317_create_t_penjualan_table.php 2025_02_25_052358_create_t_penjualan_detail_table.php > run()

1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class
10 class LevelSeeder extends Seeder
{
    Qodo Gen: Options | Test this method
11 /**
12 * Run the database seeds.
13 */
14 references | 0 overrides | Qodo Gen: Options | Test this method
15 public function run(): void
16 {
    $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']
    ];
19 DB::table('table: m_level')->insert(values: $data);
20
21 }
22
23 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan db:seed --class=LevelSeeder

INFO Seeding database.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

LN 21, Col 11 Spaces: 4 UTF-8 LF () PHP 8.4.1: 8.2 Qodo Gen

The screenshot shows the phpMyAdmin interface for the 'PWL_POS' database. The 'm_level' table is selected. The table structure is as follows:

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	3	STF	Staff/Kasir	NULL	NULL

A. In this section we're implementing **SEEDER** feature to inserting the data of every table. in this particular section we insert the data for the m_level by adding into three type of user, to add an use the seeder we're using particular syntaxes that are :

- php artisan make:seeder <nama-class-seeder> (create seeder)
- php artisan db:seed --class=<nama-class-seeder> (to send the seeder data to the database)

- Seeder on **M_USER**

PWL - Jobsheet 03 - Migrasi x localhost:8888 / localhost / P x (11752) DJ APT VERSI TH x +

To exit full screen, press and hold esc

localhost:8888/phpMyAdmin5/index.php?route=/sql&db=PWL

Server: localhost:8889 Database: PWL_POS Table: m_user

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

Showing rows 0 - 2 (total, Query took 0.0001 seconds.)

SELECT * FROM `m_user`

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

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
<input type="checkbox"/>	1	1	admin	administrator	\$2y\$12\$0btvva7JC/WcizwX0EeWcCOElkYbgQBkAsilN8Pyxz7...	NULL	NULL
<input type="checkbox"/>	2	2	manager	Manager	\$2y\$12\$Vq7a03vzHizDzwEIYr0je3F9SNOR6FpA.Z/QzdaAh...	NULL	NULL
<input type="checkbox"/>	3	3	staff	Staff/Kasir	\$2y\$12\$J18AmUwG9.3oJHaR.bkAeL7uO1OKAbL0wlCCWUJmH...	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

Query results operations

Print Copy to clipboard Export Display chart Create view

Console

UserSeeder.php — Week-3

```

UserSeeder.php
10 class UserSeeder extends Seeder
11 {
12     public function run(): void
13     {
14         $data = [
15             [
16                 'password' => Hash::make(value: '12345'), //class untuk mengenkripsi/hash password
17             [
18                 'user_id' => 2,
19                 'level_id' => 2,
20                 'username' => 'manager',
21                 'nama' => 'Manager',
22                 'password' => Hash::make(value: '12345')
23             ],
24             [
25                 'user_id' => 3,
26                 'level_id' => 3,
27                 'username' => 'staff',
28                 'nama' => 'Staff/Kasir',
29                 'password' => Hash::make(value: '12345')
30             ],
31             [
32                 'user_id' => 4,
33                 'level_id' => 4,
34                 'username' => 'admin',
35                 'nama' => 'Administrator',
36                 'password' => Hash::make(value: '12345')
37             ],
38         ];
39         DB::table(table: 'm_user')->insert(values: $data);
40     }
41 }
42
43

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

- macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:seeder UserSeeder

INFO Seeder [database/seeds/UserSeeder.php] created successfully.
- macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan db:seed --class=UserSeeder

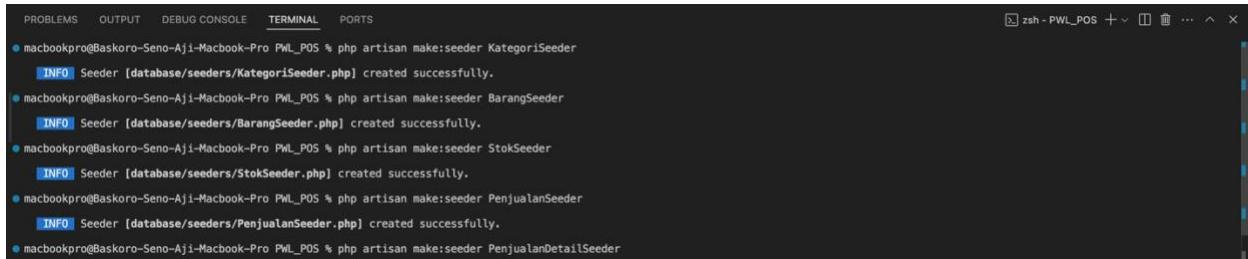
INFO Seeding database.
- macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

Ln 40, Col 44 Spaces: 4 UTF-8 LF () PHP 8.4.1: 8.2 Qode Gen

B. Implementing the same syntaxes to create the user by referring to the equivalent category to be insert on the databases

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



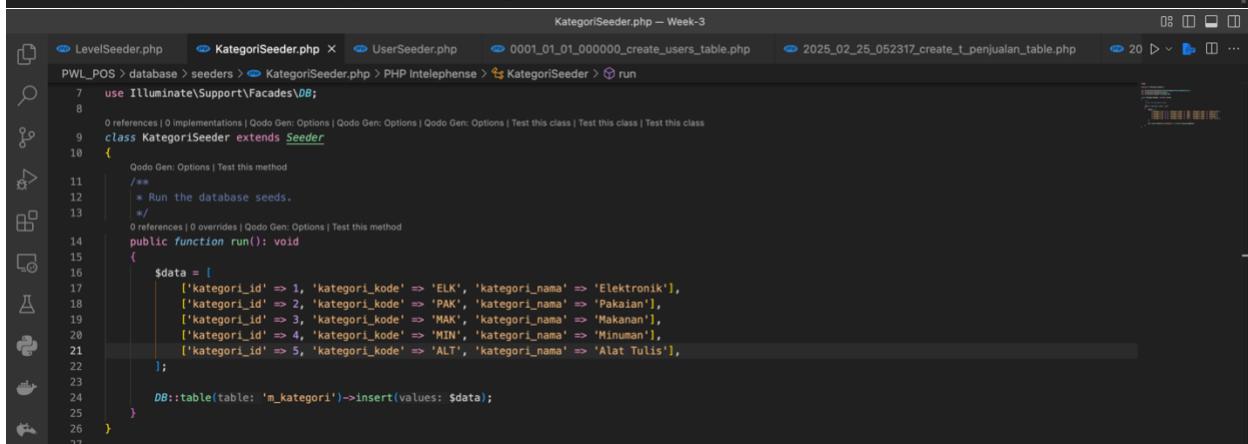
```
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:seeder KategoriSeeder
INFO Seeder [database/seeders/KategoriSeeder.php] created successfully.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:seeder BarangSeeder
INFO Seeder [database/seeders/BarangSeeder.php] created successfully.

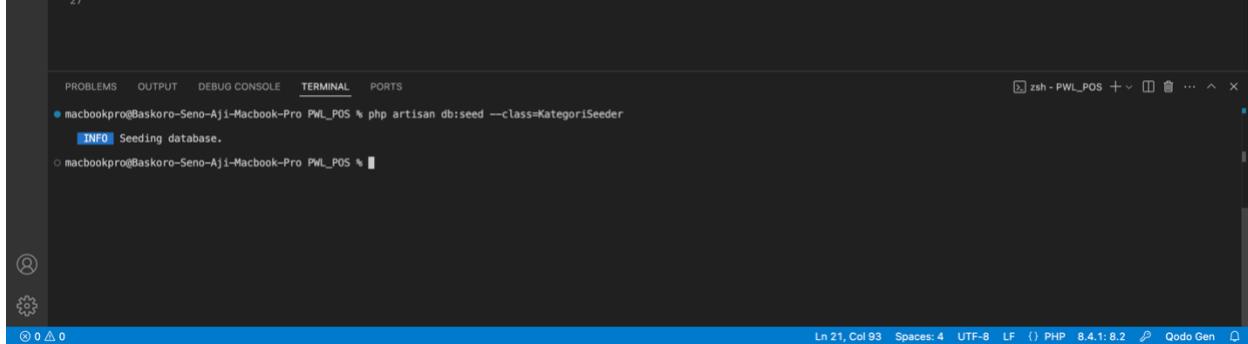
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:seeder StokSeeder
INFO Seeder [database/seeders/StokSeeder.php] created successfully.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:seeder PenjualanSeeder
INFO Seeder [database/seeders/PenjualanSeeder.php] created successfully.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:seeder PenjualanDetailSeeder
```



```
use Illuminate\Support\Facades\DB;
use Illuminate\Database\Seeder;
class KategoriSeeder extends Seeder
{
    public function run(): void
    {
        $data = [
            ['kategori_id' => 1, 'kategori_kode' => 'ELK', 'kategori_nama' => 'Elektronik'],
            ['kategori_id' => 2, 'kategori_kode' => 'PAK', 'kategori_nama' => 'Pakaian'],
            ['kategori_id' => 3, 'kategori_kode' => 'MAK', 'kategori_nama' => 'Makanan'],
            ['kategori_id' => 4, 'kategori_kode' => 'MIN', 'kategori_nama' => 'Minuman'],
            ['kategori_id' => 5, 'kategori_kode' => 'ALT', 'kategori_nama' => 'Alat Tulis'],
        ];
        DB::table('m_kategori')->insert($data);
    }
}
```



```
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan db:seed --class=KategoriSeeder
INFO Seeding database.
```

Ln 21, Col 93 Spaces: 4 UTF-8 LF () PHP 8.4.1:8.2 Qodo Gen

BarangSeeder.php — Week-3

```

class BarangSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            ['kategori_id' => 1, 'barang_kode' => 'ELE001', 'barang_nama' => 'Laptop', 'harga_beli' => 5000000, 'harga_jual' => 7000000],
            ['kategori_id' => 1, 'barang_kode' => 'ELE002', 'barang_nama' => 'Smartphone', 'harga_beli' => 3000000, 'harga_jual' => 4500000],
            ['kategori_id' => 2, 'barang_kode' => 'PAK001', 'barang_nama' => 'Baju', 'harga_beli' => 10000, 'harga_jual' => 15000],
            ['kategori_id' => 2, 'barang_kode' => 'PAK002', 'barang_nama' => 'Celana', 'harga_beli' => 120000, 'harga_jual' => 180000],
            ['kategori_id' => 3, 'barang_kode' => 'MAK001', 'barang_nama' => 'Roti', 'harga_beli' => 5000, 'harga_jual' => 7500]
        ];

        DB::table('m_barang')->insert($data);
    }
}

```

StokSeeder.php — Week-3

```

class StokSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            ['stok_id' => 1, 'barang_id' => 1, 'user_id' => 1, 'stok_tanggal' => now(), 'stok_jumlah' => 100],
            ['stok_id' => 2, 'barang_id' => 2, 'user_id' => 2, 'stok_tanggal' => now(), 'stok_jumlah' => 50],
            ['stok_id' => 3, 'barang_id' => 3, 'user_id' => 3, 'stok_tanggal' => now(), 'stok_jumlah' => 200],
        ];

        DB::table('t_stok')->insert($data);
    }
}

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan db:seed --class=BarangSeeder

INFO Seeding database.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan db:seed --class=StokSeeder

INFO Seeding database.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

PenjualanSeeder.php – Week-3

```
namespace Database\Seeders;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;

class PenjualanSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            ['penjualan_id' => 1, 'user_id' => 1, 'pembeli' => 'Pembeli 1', 'penjualan_kode' => 'TRX001', 'jumlah' => 1],
            ['penjualan_id' => 2, 'user_id' => 2, 'pembeli' => 'Pembeli 2', 'penjualan_kode' => 'TRX002', 'jumlah' => 1],
            ['penjualan_id' => 3, 'user_id' => 3, 'pembeli' => 'Pembeli 3', 'penjualan_kode' => 'TRX003', 'jumlah' => 1],
            ['penjualan_id' => 4, 'user_id' => 1, 'pembeli' => 'Pembeli 4', 'penjualan_kode' => 'TRX004', 'jumlah' => 1],
            ['penjualan_id' => 5, 'user_id' => 2, 'pembeli' => 'Pembeli 5', 'penjualan_kode' => 'TRX005', 'jumlah' => 1]
        ];

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan db:seed --class=PenjualanSeeder

INFO Seeding database.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

Ln 21, Col 141 Spaces: 4 UTF-8 LF (I) PHP 8.4.1: 8.2 Qodo Gen

PenjualanDetailSeeder.php – Week-3

```
class PenjualanDetailSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        $data = [
            ['detail_id' => 1, 'penjualan_id' => 1, 'barang_id' => 1, 'harga' => 5500000, 'jumlah' => 1],
            ['detail_id' => 2, 'penjualan_id' => 1, 'barang_id' => 2, 'harga' => 3500000, 'jumlah' => 1],
            ['detail_id' => 3, 'penjualan_id' => 1, 'barang_id' => 3, 'harga' => 150000, 'jumlah' => 1],
            ['detail_id' => 4, 'penjualan_id' => 2, 'barang_id' => 4, 'harga' => 170000, 'jumlah' => 2],
            ['detail_id' => 5, 'penjualan_id' => 2, 'barang_id' => 5, 'harga' => 10000, 'jumlah' => 3],
            ['detail_id' => 6, 'penjualan_id' => 2, 'barang_id' => 1, 'harga' => 5500000, 'jumlah' => 1],
            ['detail_id' => 7, 'penjualan_id' => 3, 'barang_id' => 2, 'harga' => 3500000, 'jumlah' => 2],
            ['detail_id' => 8, 'penjualan_id' => 3, 'barang_id' => 3, 'harga' => 150000, 'jumlah' => 2],
            ['detail_id' => 9, 'penjualan_id' => 3, 'barang_id' => 4, 'harga' => 170000, 'jumlah' => 1],
            ['detail_id' => 10, 'penjualan_id' => 4, 'barang_id' => 5, 'harga' => 10000, 'jumlah' => 4],
            ['detail_id' => 11, 'penjualan_id' => 4, 'barang_id' => 1, 'harga' => 5500000, 'jumlah' => 1],
            ['detail_id' => 12, 'penjualan_id' => 4, 'barang_id' => 2, 'harga' => 3500000, 'jumlah' => 1],
            ['detail_id' => 13, 'penjualan_id' => 5, 'barang_id' => 3, 'harga' => 150000, 'jumlah' => 2],
            ['detail_id' => 14, 'penjualan_id' => 5, 'barang_id' => 4, 'harga' => 170000, 'jumlah' => 2],
            ['detail_id' => 15, 'penjualan_id' => 5, 'barang_id' => 5, 'harga' => 10000, 'jumlah' => 3],
        ];

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan db:seed --class=PenjualanDetailSeeder

INFO Seeding database.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %

Ln 21, Col 141 Spaces: 4 UTF-8 LF (I) PHP 8.4.1: 8.2 Qodo Gen

- C. This section task we need to implement the fill the data or do the seeder with the criteria above, in this section I've manage to do it all by implement or fill it in every column sections so It could be detect by the database

DB FAÇADE

DB Façade is a feature that use the query using RAW QUERY or code the full SQL syntax, there are several type of DB Façade that use in this practicum, that are :

a. DB::select()

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

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

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

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

b. DB::insert()

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

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

c. DB::update()

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

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

d. DB::delete()

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

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

Practicum 4 - Implementasi DB Façade

- DB Façade Insert :

web.php - Week-3

```
PWL_POS > routes > web.php > ...
1  <?php
2
3  use App\Http\Controllers\LevelController;
4  use Illuminate\Support\Facades\Route;
5
6  Qodo Gen: Options | Test this function
7  Route::get('/', action: function () { View {
8      return view('welcome');
9  });
10 Route::get(uri: '/level', action: [LevelController::class, 'index']);
11 |
```

TERMINAL

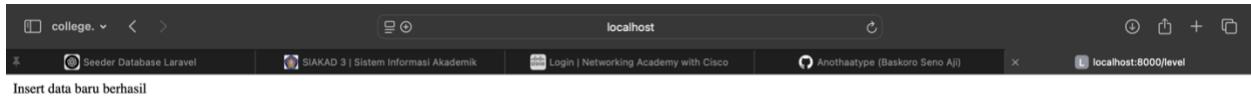
```
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:controller LevelController
INFO Controller [app/Http/Controllers/LevelController.php] created successfully.
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %
```

LevelController.php

```
PWL_POS > app > Http > Controllers > LevelController.php > PHP Intelephense > LevelController > index
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  2 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class | Test this class
9  class LevelController extends Controller
10 {
11     1 reference | 0 overrides | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this method | Test this method
12     public function index(): string
13     {
14         DB::insert(query: 'insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)');
15         return 'Insert data baru berhasil';
16     }
17 }
```

TERMINAL

```
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:controller LevelController
INFO Controller [app/Http/Controllers/LevelController.php] created successfully.
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %
```



	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	4	CUS	Pelanggan	2025-02-25 07:14:54	NULL

- A. In this section we do some several steps to implement the DB Façade statement, first we need to create the controller by implementing php artisan make:controller (NameController) where in this section are focusing for the M_LEVEL table, second we need to set and connect the route to the controller. Third we need to insert the data to the level controller. Fourth we need to test and launch it in the web by accessing the /level on localhost, then we can see the output of the inserting data also the changes that occur in the PhpMyAdmin

- DB Façade Update



The screenshot shows a code editor and a browser window. The code editor displays PHP code for a controller named LevelController.php. The browser window shows a successful database update message.

```
1 p
2
3 space App\Http\Controllers;
4
5 Illuminate\Http\Request;
6 Illuminate\Support\Facades\DB;
7
8 s LevelController extends Controller
9
10 public function index(): string
11 {
12     //DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS
13     //return 'Insert data baru berhasil';
14
15     $row = DB::update(query: 'update m_level set level_nama = ? where level_kode = ?' , bindings
16     return "...Update data berhasil. Jumlah data yang diupdate: ' . $row. ' baris';|
17 }
18
19
```

localhost

Update data berhasil. Jumlah data yang diupdate: 1baris

The screenshot shows the phpMyAdmin interface for the 'm_level' table. The table has columns: level_id, level_kode, level_nama, created_at, and updated_at. The data is as follows:

level_id	level_kode	level_nama	created_at	updated_at
1	ADM	Administrator	NULL	NULL
2	MNG	Manager	NULL	NULL
3	STF	Staff/Kasir	NULL	NULL
4	CUS	Customer	2025-02-25 07:14:54	NULL

- A. In this section we're implementing the DB Façade for updating the data inside the table where we add CUS in the new row

- DB Façade Delete

```

LevelController.php — Week-3
Screenshot Screenshot
Welcome UserController.php LevelController.php X
PWL_POS > app > Http > Controllers > LevelController.php > PHP Symbols > LevelController
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 2 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this method
9 class LevelController extends Controller
9 {
10     1 reference | 0 overrides | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this method | Test this method
11     public function index(): string
11     {
12         //DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
13         //return 'Insert data baru berhasil';
14
15         // Facade untuk Update Data
16         //$row = DB::update('update m_level set level_nama = ? where level_kode = ? ', ['Customer', 'CUS']);
17         //return 'Update data berhasil. Jumlah data yang diupdate: ' . $row. 'baris';
18
19         // Facade untuk Delete data
20         $row = DB::delete('delete from m_level where level_kode = ? ', ['CUS']);
21         return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row. 'baris';
22
23         //Facade untuk menampilkan data
24         //$data = DB::select('select * from m_level');
25         //return view('level', ['data' => $data]);
26     }
27 }

```

Ln 21, Col 9 Spaces: 4 UTF-8 LF (PHP 8.4.1: 8.2 Qodo Gen

The screenshot shows the phpMyAdmin interface for the 'm_level' table. The table has four rows with the following data:

	level_id	level_kode	level_nama	created_at	updated_at
<input type="checkbox"/>	1	ADM	Administrator	NULL	NULL
<input type="checkbox"/>	2	MNG	Manager	NULL	NULL
<input type="checkbox"/>	3	STF	Staff/Kasir	NULL	NULL
<input type="checkbox"/>	4	CUS	Customer	2025-02-25 07:14:54	NULL

Below the table, there are buttons for 'Check all', 'With selected:', and various operations like Edit, Copy, Delete, Export, Print, Copy to clipboard, Export, Display chart, and Create view.

At the bottom of the page, a message says: "Delete data berhasil. Jumlah data yang dihapus: 1baris".

- B. In this section we try to implement the delete db façade by deleting the data that we've update in db faced update

- DB Façade Select

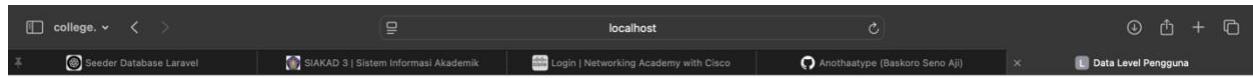
The screenshot shows a code editor with two tabs open: `LevelController.php` and `level.blade.php`.

LevelController.php (Top Tab):

```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class LevelController extends Controller
9 {
10     // Facade untuk Insert Data
11     //DB::insert('insert into m_level(level_kode, level_nama, created_at) values(?, ?, ?)', ['CUS', 'Pelanggan', now()]);
12     //return 'Insert data baru berhasil';
13
14     // Facade untuk Update Data
15     //DB::update('update m_level set level_nama = ? where level_kode = ?', ['Customer', 'CUS']);
16     //return 'Update data berhasil. Jumlah data yang diupdate: ' . $row. 'baris';
17
18     // Facade untuk Delete data
19     //DB::delete('delete from m_level where level_kode = ?', ['CUS']);
20     //return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row. 'baris';
21
22     //Facade untuk menampilkan data
23     $data = DB::select('select * from m_level');
24     return view('level', ['data' => $data]);
25 }
26
27 }
28 }
```

level.blade.php (Bottom Tab):

```
1 <html>
2     <head>
3         <title>Data Level Pengguna</title>
4     </head>
5     <body>
6         <h1>Data Level Pengguna</h1>
7         <table border="1" cellpadding="2" cellspacing="0">
8             <tr>
9                 <th>ID</th>
10                <th>Kode Level</th>
11                <th>Nama Level</th>
12            </tr>
13            @foreach ($data as $d)
14            <tr>
15                <td>{{ $d->level_id }}</td>
16                <td>{{ $d->level_kode }}</td>
17                <td>{{ $d->level_nama }}</td>
18            </tr>
19            @endforeach
20        </table>
21    </body>
22 </html>
```



Data Level Pengguna

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

Showing rows 0 - 2 (3 total, Query took 0.0001 seconds.)

```
SELECT * FROM `m_level`
```

Profile [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

level_id	level_kode	level_nama	created_at	updated_at
1	ADM	Administrator	NULL	NULL
2	MNG	Manager	NULL	NULL
3	STF	Staff/Kasir	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

Query results operations

Print Copy to clipboard Export Display chart Create view

Console

- C. In this section we try to show or select all the data in the m_level table by creating the html page or view in the Laravel, this section manage a new step where we need to connect the blade.php into the controller and route.

Query Builder

Query Builder is a feature provided by Laravel for performing CRUD (Create, Retrieve/Read, Update, Delete) operations on a database. Unlike raw queries in DB Facade, which require us to write SQL commands manually, Query Builder allows access to these SQL commands through methods. This means that instead of writing SQL commands directly, we only need to call the available methods in Query Builder.

- a. Perintah untuk *insert* data dengan method `insert()`

```
DB::table('m_kategori')->insert(['kategori_kode' => 'SMP', 'kategori_nama' => 'Smartphone']);
```

Query yang dihasilkan dari kode di atas adalah

```
insert into m_kategori(kategori_kode, kategori_nama) values('SMP', 'Smartphone');
```

- b. Perintah untuk *update* data dengan method `where()` dan `update()`

```
DB::table('m_kategori')->where('kategori_id', 1)->update(['kategori_nama' => 'Makanan Ringan']);
```

Query yang dihasilkan dari kode di atas adalah

```
update m_kategori set kategori_nama = 'Makanan Ringan' where kategori_id = 1;
```

- c. Perintah untuk *delete* data dengan method `where()` dan `delete()`

```
DB::table('m_kategori')->where('kategori_id', 9)->delete();
```

Query yang dihasilkan dari kode di atas adalah

```
delete from m_kategori where kategori_id = 9;
```

- d. Perintah untuk ambil data

<i>Method Query Builder</i>	<i>Query yang dihasilkan</i>
<code>DB::table('m_kategori')->get();</code>	<code>select * from m_kategori</code>
<code>DB::table('m_kategori')->where('kategori_id', 1)->get();</code>	<code>select * from m_kategori where kategori_id = 1;</code>
<code>DB::table('m_kategori')->select('kategori_kode')->where('kategori_id', 1)->get();</code>	<code>select kategori_kode from m_kategori where kategori_id = 1;</code>

Praktikum 5 – Implementasi Query Builder

- Query Builder Insert

The screenshot shows a code editor interface with the following details:

- File:** KategoriController.php — Week-3
- Code Content (Controller):**

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

- Terminal Output:**

```
macbookpro@Baskoro-Seno-Aji-MacBook-Pro PWL_POS % php artisan make:controller KategoriController
INFO Controller [app/Http/Controllers/KategoriController.php] created successfully.
```

- Code Below (routes/web.php):**

```
1 <?php
2
3 use App\Http\Controllers\KategoriController;
4 use App\Http\Controllers\LevelController;
5 use Illuminate\Support\Facades\Route;
6
7 Qodo Gen: Options | Test this function
8 Route::get(uri: '/', action: function (): View {
9     return view(view: 'welcome');
10 });
11 Route::get(uri: '/level', action: [LevelController::class, 'index']);
12 Route::get(uri: '/kategori', action: [KategoriController::class, 'index']);
```

```
PWL_POS > app > Http > Controllers > KategoriController.php > PHP Symbols > KategoriController.php
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
Qodo Gen: Options | Test this class
7
2 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class
8  class KategoriController extends Controller
9  {
1 reference | 0 overrides | Qodo Gen: Options | Qodo Gen: Options | Test this method | Test this method
10 public function index(): string
11 {
12     $data = [
13         'kategori_kode' => 'SNK',
14         'kategori_nama' => 'Snack/Makanan Ringan',
15         'created_at' => now()
16     ];
17
18     DB::table('m_kategori')->insert(values: $data);
19     return 'Insert data baru berhasil';
20 }
21 }
22 }
```

- A. in this terms we insert all the data in the \$data variable but to call or use it we use the index function where the already connect the route into the KategoriController so in that terms when we use /kategori we can see the output of the return where it is “Insert data baru berhasil”

- Query Builder Update

>Welcome web.php KategoriController.php ×

PWL_POS > app > Http > Controllers > KategoriController.php > PHP Intelephense > KategoriController

```

1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6  use Illuminate\Support\Facades\DB;
7
8  class KategoriController extends Controller
9
10
11     public function index(): string
12     {
13         $data = [
14             'kategori_kode' => 'SNK',
15             'kategori_nama' => 'Snack/Makanan Ringan',
16             'created_at' => now()
17         ];
18
19         // Digunakan untuk menambah data
20         DB::table('m_kategori')->insert($data);
21         return 'Insert data baru berhasil';
22
23         $row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
24         return 'Update data berhasil. Jumlah data yang diupdate: ' . $row. 'baris';
25     }
26

```

localhost:8888/phpMyAdmin5/index.php?route=/sql&db=PWL_POS&table=m_kategori

Showing rows 0 - 6 (7 total, Query took 0.00004 seconds.)

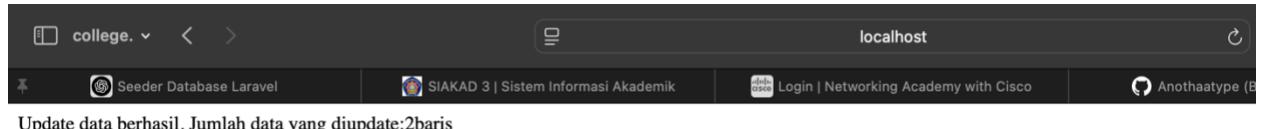
	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
<input type="checkbox"/>	1	ELK	Elektronik	NULL	NULL
<input type="checkbox"/>	2	PAK	Pakaian	NULL	NULL
<input type="checkbox"/>	3	MAK	Makanan	NULL	NULL
<input type="checkbox"/>	4	MIN	Minuman	NULL	NULL
<input type="checkbox"/>	5	ALT	Alat Tulis	NULL	NULL
<input type="checkbox"/>	6	SNK	Camilan	2025-02-25 07:49:42	NULL
<input type="checkbox"/>	7	SNK	Camilan	2025-02-25 07:49:46	NULL

With selected: Edit Copy Delete Export

Query results operations

Print Copy to clipboard Export Display chart Create view

Console



- B. In this section we're implementing the update terms for the Query Builder where we can see that the output are updating 2 row, we add the camilan 2 times.

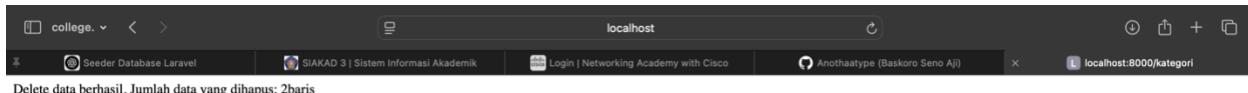
- Query Builder Delete

```

Welcome web.php KategoriController.php ×
PWL_POS > app > Http > Controllers > KategoriController.php > PHP Symbols > KategoriController
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 2 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class | Test this class
9 class KategoriController extends Controller
{
    1 reference | 0 overrides | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this method | Test this method
10
11     public function index(): string
12     {
13         /*$data = [
14             'kategori_kode' => 'SNK',
15             'kategori_nama' => 'Snack/Makanan Ringan',
16             'created_at' => now()
17         ];
18
19         // Digunakan untuk menambah data
20         DB::table('m_kategori')->insert($data);
21         return 'Insert data baru berhasil';
22
23         // Digunakan untuk mengupdate data
24         //$row = DB::table('m_kategori')->where('kategori_kode', 'SNK')->update(['kategori_nama' => 'Camilan']);
25         //return 'Update data berhasil. Jumlah data yang diupdate: ' . $row. 'baris';
26
27         // Digunakan untuk menghapus data
28         $row = DB::table('m_kategori')->where(column: 'kategori_kode', operator: 'SNK')->delete();
29         return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row. 'baris';
30     }
31

```

	kategori_id	kategori_kode	kategori_nama	created_at	updated_at
1	ELK	Elektronik	NULL	NULL	
2	PAK	Pakaiian	NULL	NULL	
3	MAK	Makanan	NULL	NULL	
4	MIN	Minuman	NULL	NULL	
5	ALT	Alat Tulis	NULL	NULL	



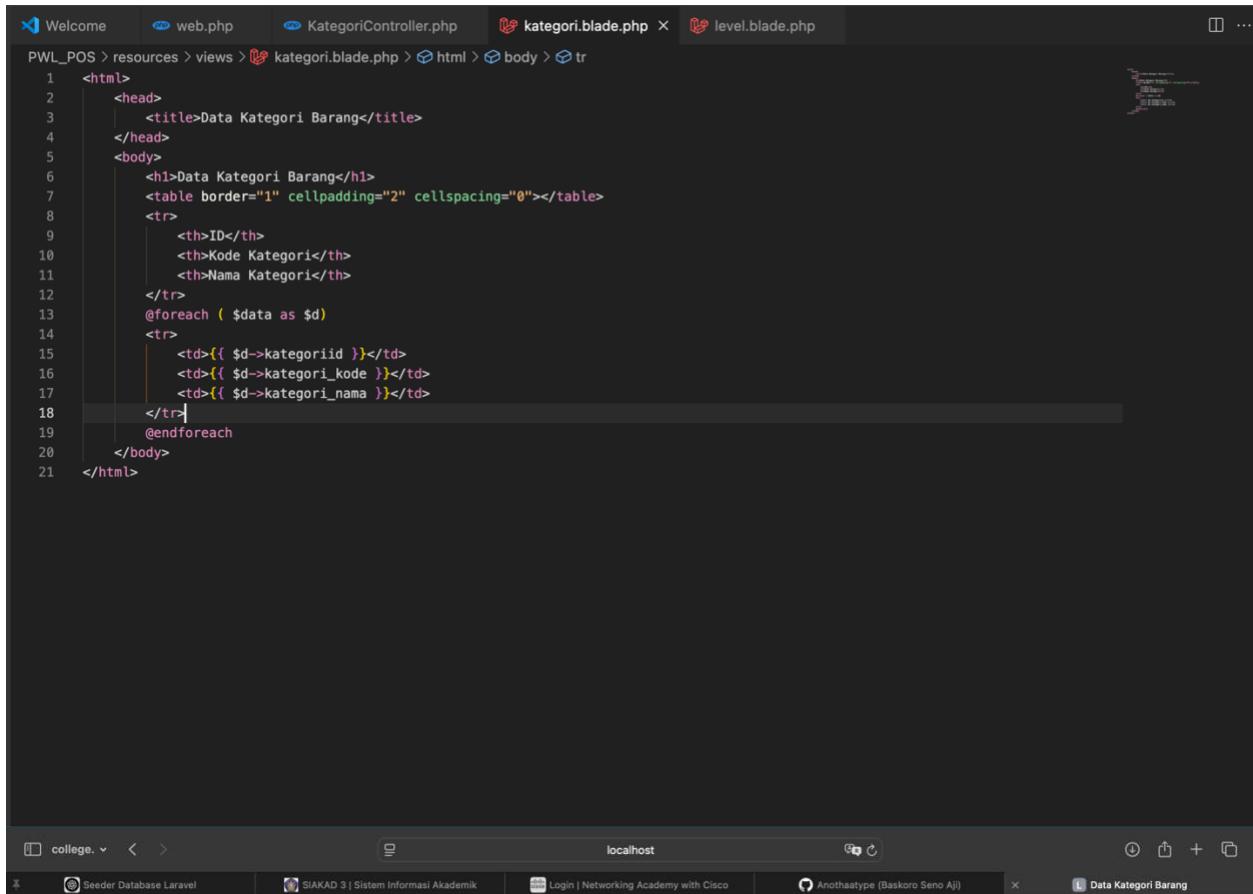
- C. In this section we can see clear that we're implementing the delete feature, we look that the output has erased the 2 camilan rows.

- Query Builder Select

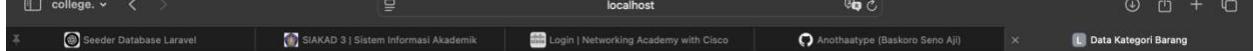
```
KategoriController.php – Week-3
Welcome web.php KategoriController.php level.blade.php
PWL_POS > app > Http > Controllers > KategoriController.php > PHP > KategoriController > index()

1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\DB;
7
8 class KategoriController extends Controller
9 {
10     public function index(): View
11     {
12         $data = [
13             'kategori_kode' => 'SNK',
14             'kategori_nama' => 'Snack/Makanan Ringan',
15             'created_at' => now()
16         ];
17
18         // Digunakan untuk menambah data
19         DB::table('m_kategori')->insert($data);
20         return 'Insert data baru berhasil!';
21
22         // Digunakan untuk mengupdate data
23         //$row = DB::table('m_kategori')->where('kategori.kode', 'SNK')->update(['kategori.nama' => 'Camilan']);
24         //return 'Update data berhasil. Jumlah data yang diupdate: ' . $row. ' baris';
25
26         // Digunakan untuk menghapus data
27         //$row = DB::table('m_kategori')->where('kategori.kode', 'SNK')->delete();
28         //return 'Delete data berhasil. Jumlah data yang dihapus: ' . $row. ' baris';
29
30         //Digunakan untuk menampilkan semua data yang ada di dalam tabel kategori
31         $data = DB::table('m_kategori')->get();
32         return view('kategori', data: [$data]);
33     }
34 }

Ln 32, Col 40 Spaces: 4 UTF-8 LF PHP 8.4.1: 8.2 Qodo Gen
```



```
1 <html>
2   <head>
3     <title>Data Kategori Barang</title>
4   </head>
5   <body>
6     <h1>Data Kategori Barang</h1>
7     <table border="1" cellpadding="2" cellspacing="0"></table>
8     <tr>
9       <th>ID</th>
10      <th>Kode Kategori</th>
11      <th>Nama Kategori</th>
12    </tr>
13    @foreach ($data as $d)
14      <tr>
15        <td>{{ $d->kategoriid }}</td>
16        <td>{{ $d->kategori_kode }}</td>
17        <td>{{ $d->kategori_nama }}</td>
18      </tr>
19    @endforeach
20  </body>
21 </html>
```



Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	ELK	Elektronik
2	PAK	Pakaian
3	MAK	Makanan
4	MIN	Minuman
5	ALT	Alat Tulis

D. In this section we try to implement the select feature from the **QueryBuilder** by showing it into the table with the html or php in the view terms or file that we already connect to the controller and route, in this section it shown all the data that m_kategori have.

Eloquent ORM

Eloquent ORM (Object-Relational Mapping) in Laravel is a feature that makes working with databases easier by allowing you to interact with database tables using PHP objects instead of writing raw SQL queries.

INFO

Eloquent ORM memerlukan Model untuk proses konversi data pada tabel menjadi object. Object inilah yang nantinya akan kita akses dari dalam controller. Oleh karena itu **membuat Model pada Laravel berarti menggunakan Eloquent ORM**. Silahkan cek disini

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

Perintah untuk membuat model adalah sebagai berikut

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

Praktikum 6 – Implementasi Eloquent Model

- First Model Controller

The screenshot shows a terminal window with the following content:

```
PWL_POS > app > Models > UserModel.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class UserModel extends Model
8  {
9  }
10 }

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:model UserModel
[INFO] Model [app/Models/UserModel.php] created successfully.

macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS %
```

The terminal shows the command `php artisan make:model UserModel` being run, which creates the `UserModel.php` file in the `app/Models` directory. The file contains the basic structure of an Eloquent Model.

Screenshot 1: user.blade.php — Week-3

```

<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>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>

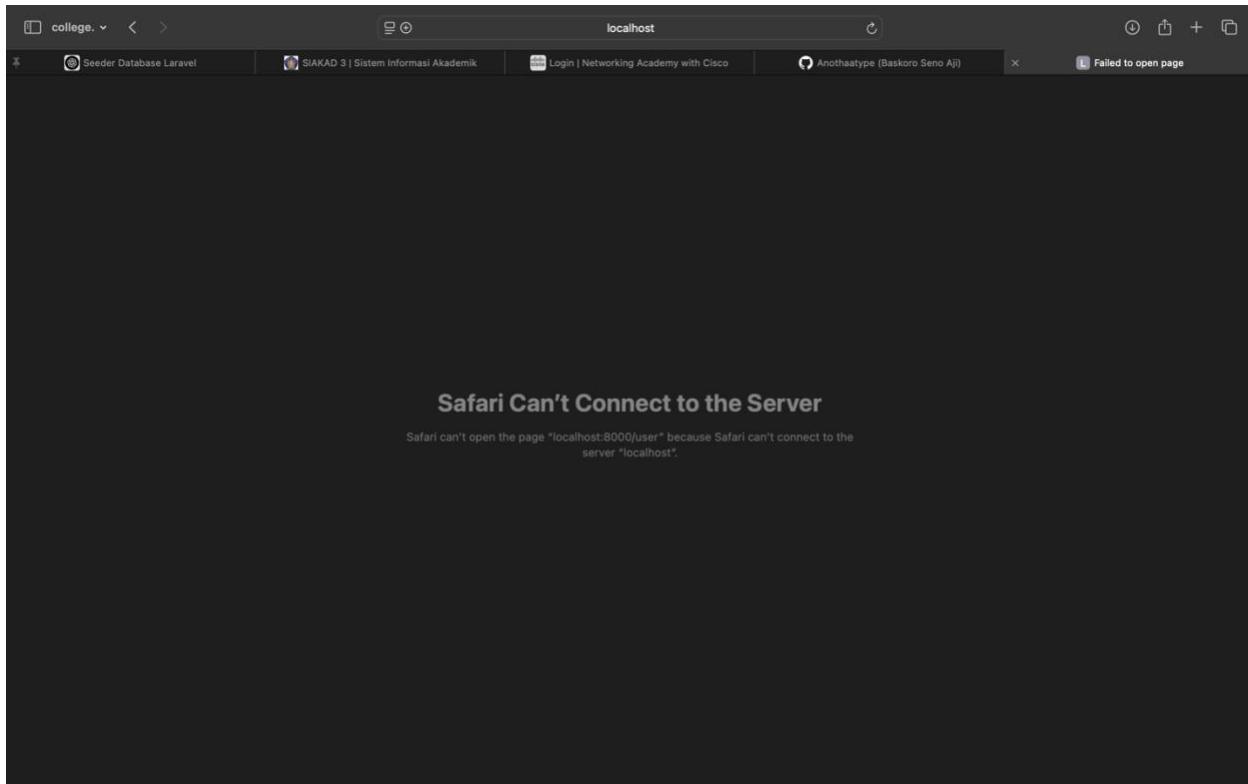
```

Screenshot 2: UserController.php — Week-3

```

<?php
namespace App\Http\Controllers;
use App\Models\UserModel;
use Illuminate\Http\Request;
class UserController extends Controller
{
    public function index(): View
    {
        //Coba akses Model UserModel
        $user = UserModel::all(); //ambil semua data dari table m_user
        return view ('user', ['data' => $user]);
    }
}

```



- A. When we try to implement the Eloquent model we need to create the route, model (as an object) and also the view where we can use it as an output, but for the first model we can't implement it because the model is in the protected areas and we can't attain or connect into it

- Second Model Controller

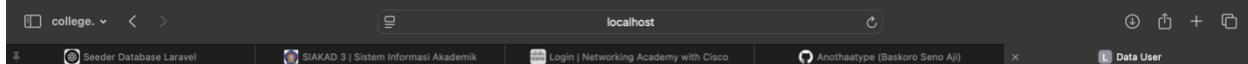
A screenshot of a browser window titled "Data User". The table has four columns: ID, Username, Nama, and ID Level Pengguna. The data is as follows:

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

- B. In the second terms of model controller we are implementing The insert() method that used, which directly inserts the data without triggering Eloquent model events.

- Third Model Controller

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



Data User

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

- C. In the third terms of model controller we are implementing The Update() method where we modifying the existing data by using an object

Penutup

1. Pada **Praktikum 1 - Tahap 5**, apakah fungsi dari `APP_KEY` pada *file setting .env* Laravel?
2. Pada **Praktikum 1**, bagaimana kita men-*generate* nilai untuk `APP_KEY`?
3. Pada **Praktikum 2.1 - Tahap 1**, secara *default* Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut?
4. Secara *default*, file migrasi terdapat kode `$table->timestamps();`, apa tujuan/*output* dari fungsi tersebut?
5. Pada File Migrasi, terdapat fungsi `$table->id();` Tipe data apa yang dihasilkan dari fungsi tersebut?
6. Apa bedanya hasil migrasi pada table `m_level`, antara menggunakan `$table->id();` dengan menggunakan `$table->id('level_id');` ?
7. Pada migration, Fungsi `->unique()` digunakan untuk apa?
8. Pada **Praktikum 2.2 - Tahap 2**, kenapa kolom `level_id` pada tabel `m_user` menggunakan `$table->unsignedBigInteger('level_id')`, sedangkan kolom `level_id` pada tabel `m_level` menggunakan `$table->id('level_id')` ?
9. Pada **Praktikum 3 - Tahap 6**, apa tujuan dari Class `Hash`? dan apa maksud dari kode program `Hash::make('1234');`?
10. Pada **Praktikum 4 - Tahap 3/5/7**, pada *query builder* terdapat tanda tanya (?), apa kegunaan dari tanda tanya (?) tersebut?
11. Pada **Praktikum 6 - Tahap 3**, apa tujuan penulisan kode `protected $table = 'm_user';` dan `protected $primaryKey = 'user_id';` ?
12. Menurut kalian, lebih mudah menggunakan mana dalam melakukan operasi CRUD ke database (*DB Façade / Query Builder / Eloquent ORM*) ? jelaskan

1. Ensuring the security of the application by making sure that encrypted data can only be decrypted by the app with the correct key
2. Php artisan key:generate (this statement will automatically generate the AppKey into our .env)

3. By default Laravel have 2 files of migration and The purpose of migration files is to manage the database schema using code, allowing easy creation, updating, or deletion of database structures without writing raw SQL queries.
4. Used for tracking changes in the database records
5. It creates a Primary Key where it uses BIGINT data type and auto increment
6. The `$table->id();` function creates a primary key column named id by default, using the BIGINT data type with auto-increment enabled. On the other hand, `$table->id('level_id');` creates a primary key column with the custom name level_id, but it still retains the BIGINT AUTO_INCREMENT properties. The key difference lies in the name of the primary key column.
7. The `->unique()` function in Laravel migrations is used to enforce uniqueness on a column. This means that no two records in the table can have the same value for that column. For example, using `$table->string('email')->unique();` ensures that email addresses in the table remain unique, preventing duplicate entries.
8. In the m_level table, the `$table->id('level_id');` function is used to create a primary key column with the name level_id, which is an auto-incrementing BIGINT field. Meanwhile, in the m_user table, the `$table->unsignedBigInteger('level_id');` function is used to define a foreign key column. This foreign key references the level_id primary key from the m_level table. Using unsignedBigInteger ensures that the foreign key matches the data type of the referenced primary key.
9. The Hash class in Laravel is used for password encryption and hashing. The code `Hash::make('1234');` generates a secure hashed version of the string '1234' using a hashing algorithm. This approach ensures that passwords are stored securely in the database, making it difficult for attackers to retrieve the original password.
10. The question mark (?) in Laravel's query builder is used for parameter binding. It acts as a placeholder for values that will be safely inserted into the query. This technique helps prevent SQL injection attacks by ensuring that user input is properly escaped before being executed. For example, in `DB::table('users')->where('id', '?')->get();`, the question mark will be replaced with the actual ID value safely.
11. The line `protected $table = 'm_user';` is used in an Eloquent model to specify the name of the database table associated with the model. This is necessary when the table name does not follow Laravel's default naming convention. Similarly, `protected $primaryKey = 'user_id';` is

used to define a custom primary key column when the default `id` column is not used. These configurations help Laravel correctly map the model to the corresponding database table and primary key.

12. among the three options, Eloquent ORM is generally the easiest for performing CRUD (Create, Read, Update, Delete) operations in Laravel. Eloquent allows developers to interact with the database using object-oriented methods instead of writing raw SQL queries. It simplifies database operations by providing built-in functions for handling relationships, validations, and queries.