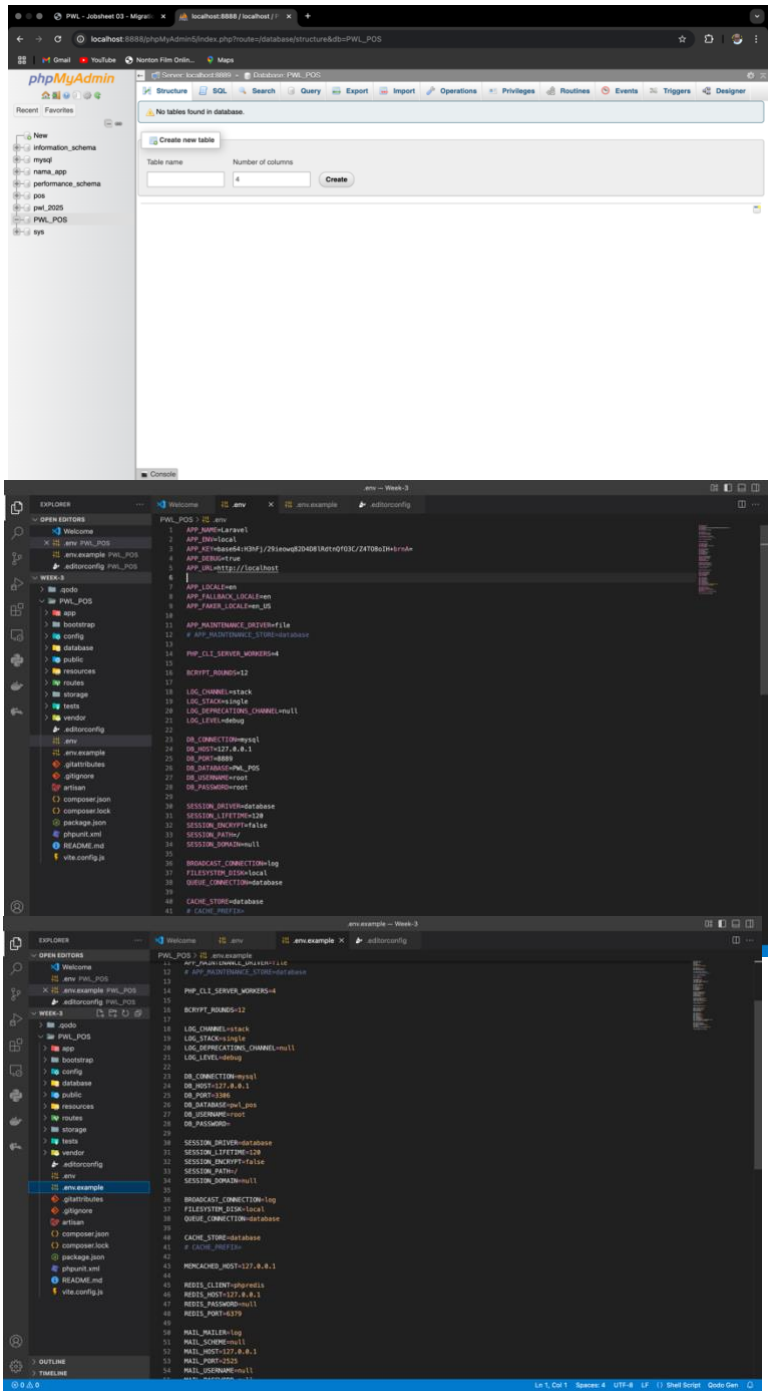
Advanced Web Programming – Migration, Seeder, DB Façade, Query Builder and Eloquent ORM
Week 3

Baskoro Seno Aji
2341720063 / 05 / TI-2I
Github Link : Github
Polytechnic State of Malang

Pengaturan Database

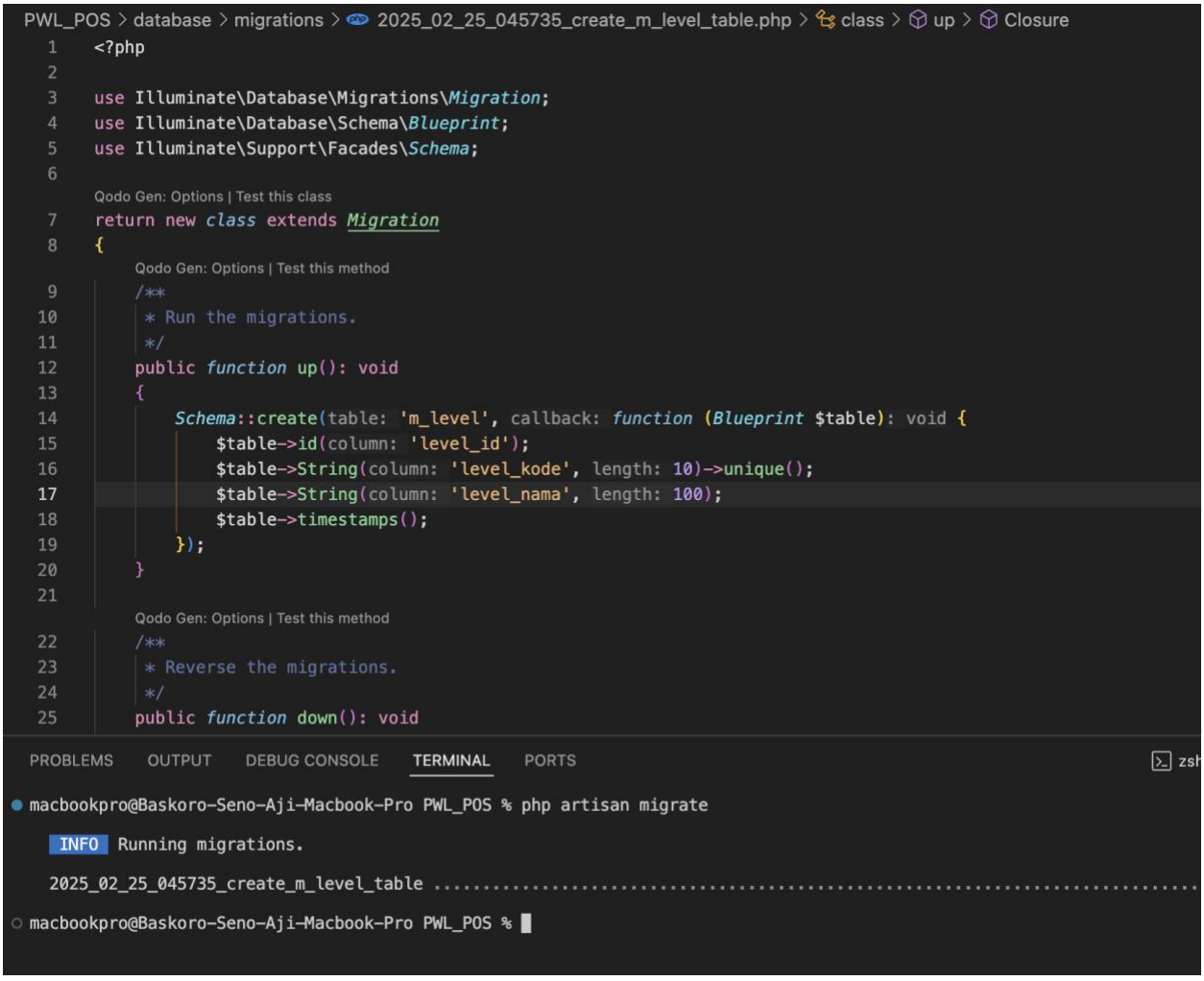
Praktikum 1 :

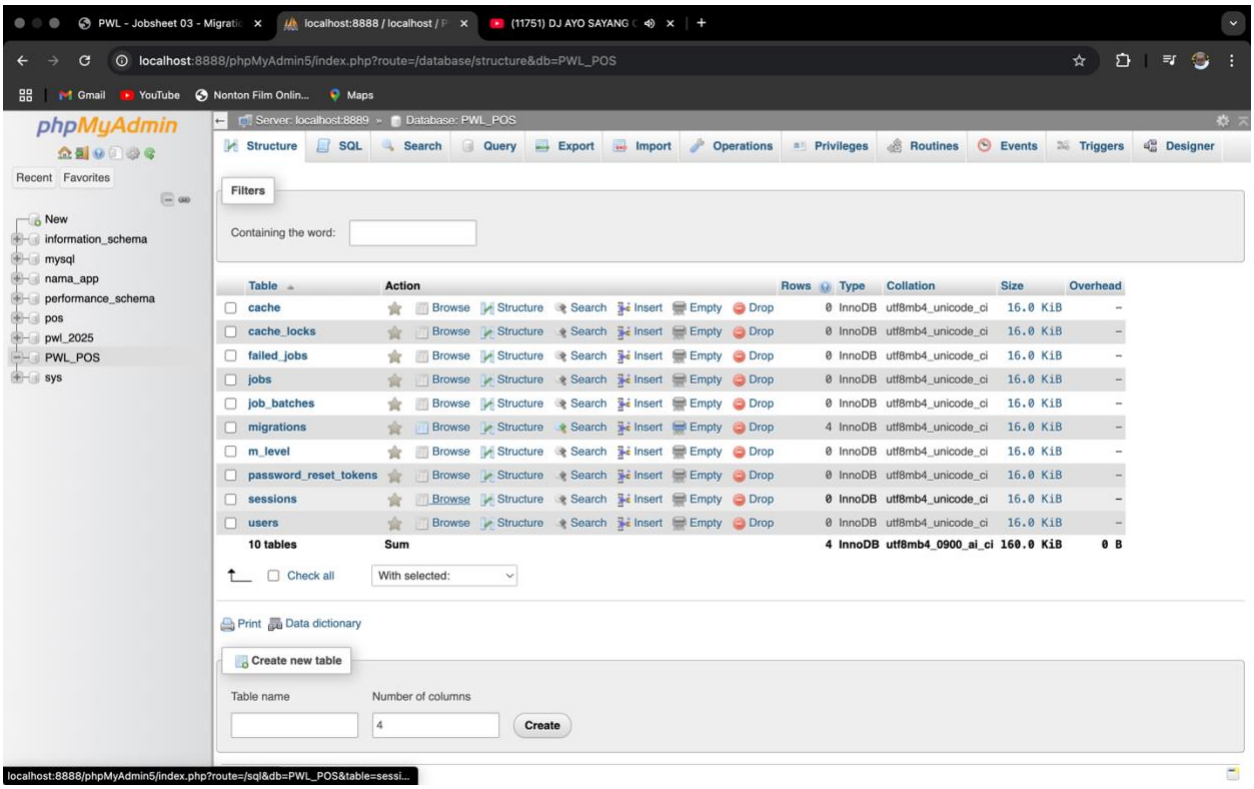


A. This section we're creating the databases first on the PhpMyAdmin then implement the database into the PWL_POS folder where it already installed by the Laravel and recheck it into the VSCode by specifically see the DB PORT, APP KEY, DB DATABASE, and change the DB PASSWORD.

MIGRATION

Praktikum 2.1 – Pembuatan File Migrasi Tanpa Relasi



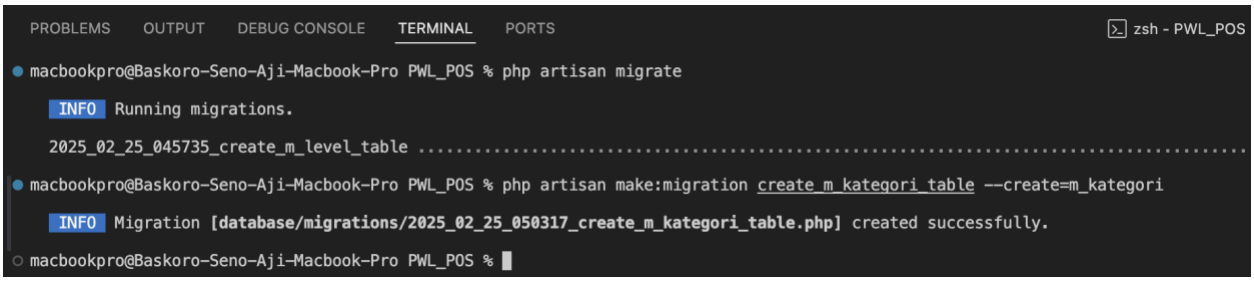


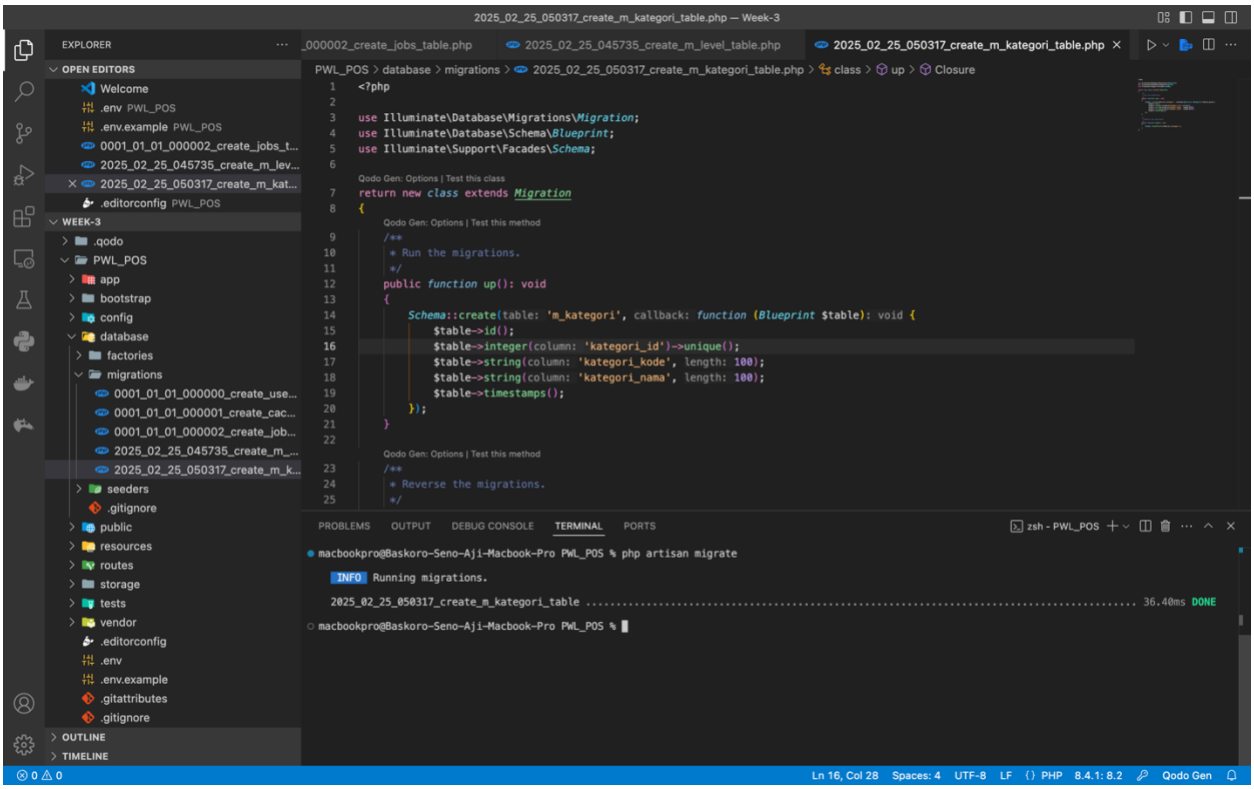
A. in this section we're implementing the **MIGRATIONS** for table who doesn't have the foreign key and only have the primary key or doesn't have any relations with the other table where we are using two main syntax sentences to create it, we use the :

- 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

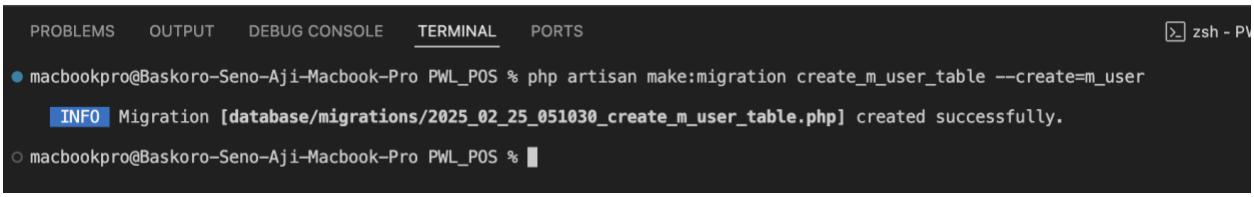
8. Buat table database dengan migration untuk table m_kategori yang sama-sama tidak memiliki foreign key

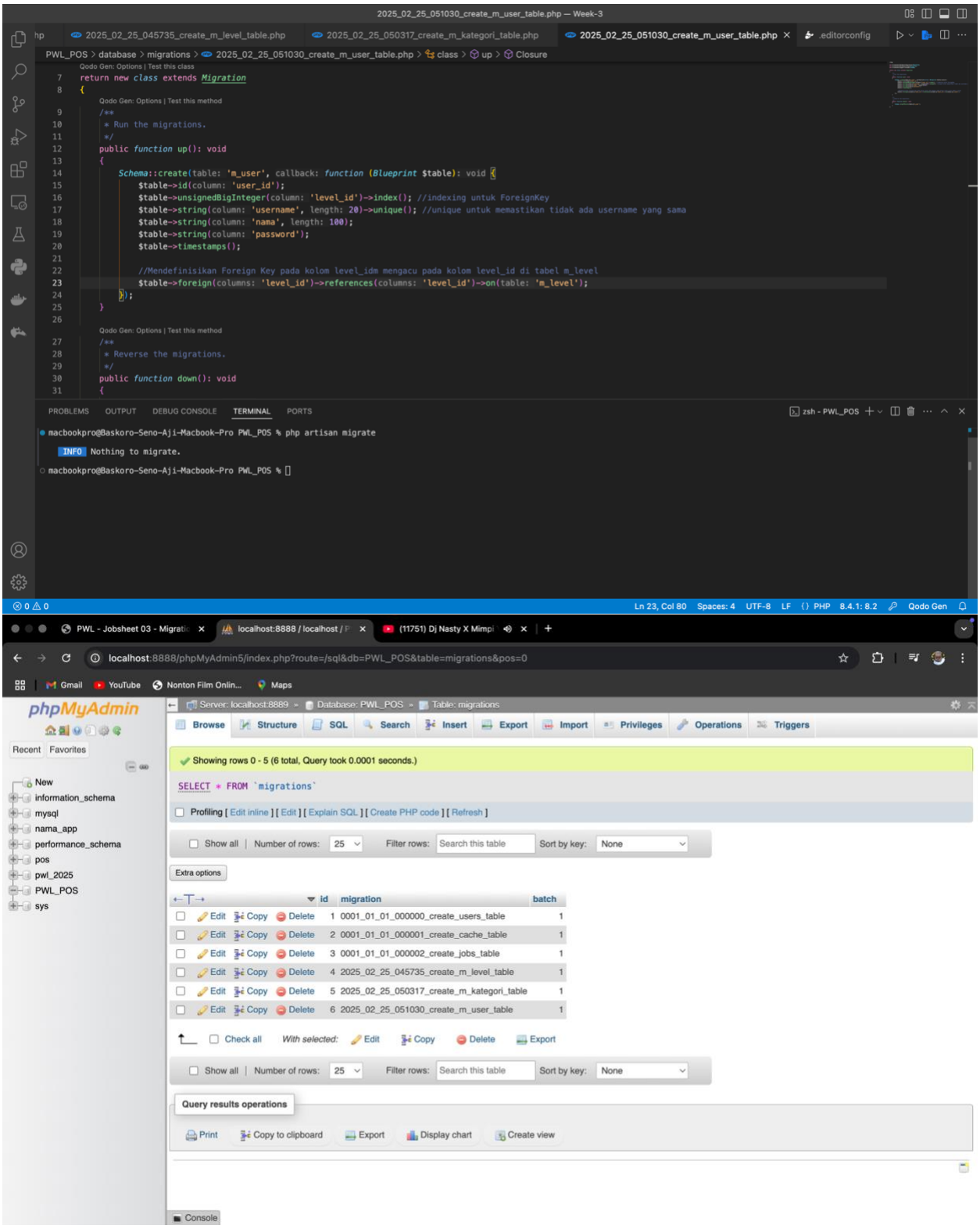




B. by implementing the same syntax sentences we've created the m_kategori and insert all the criteria that the table have.

Praktikum 2.2 – Pembuatan File Migrasi Dengan Relasi



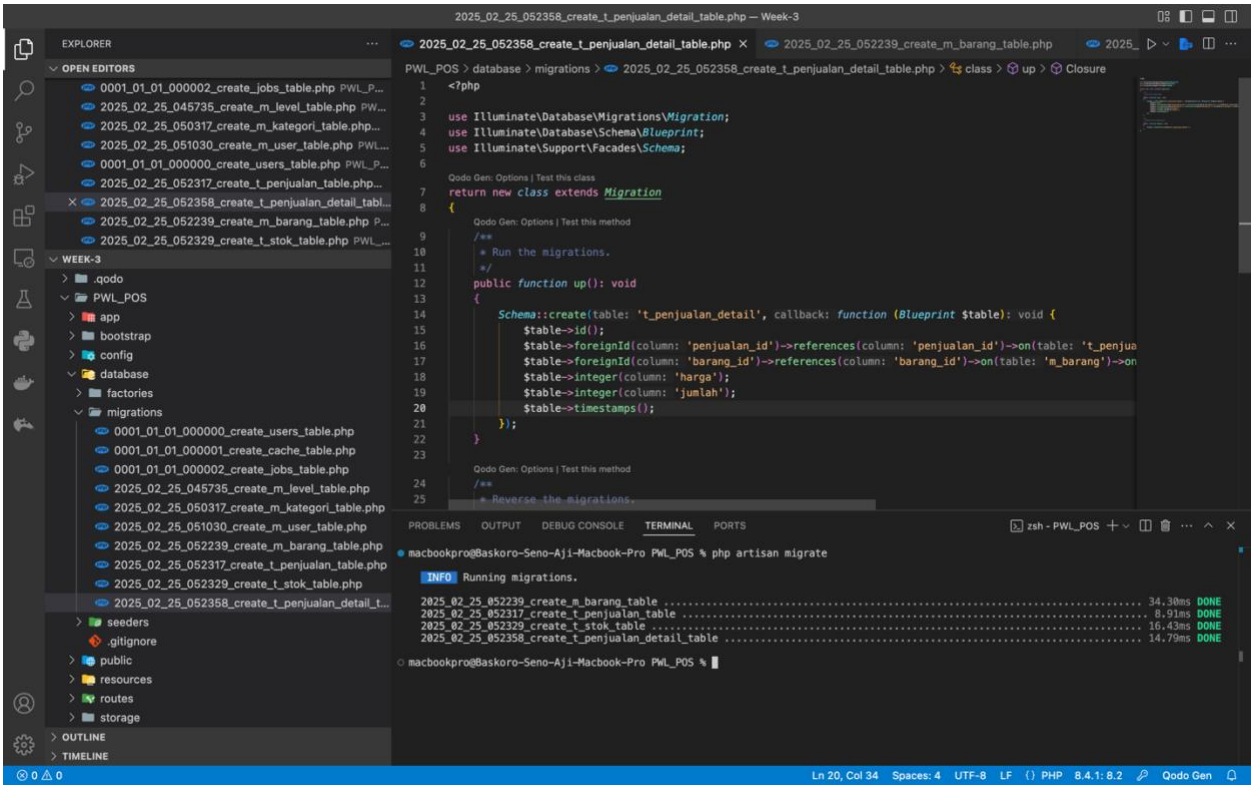


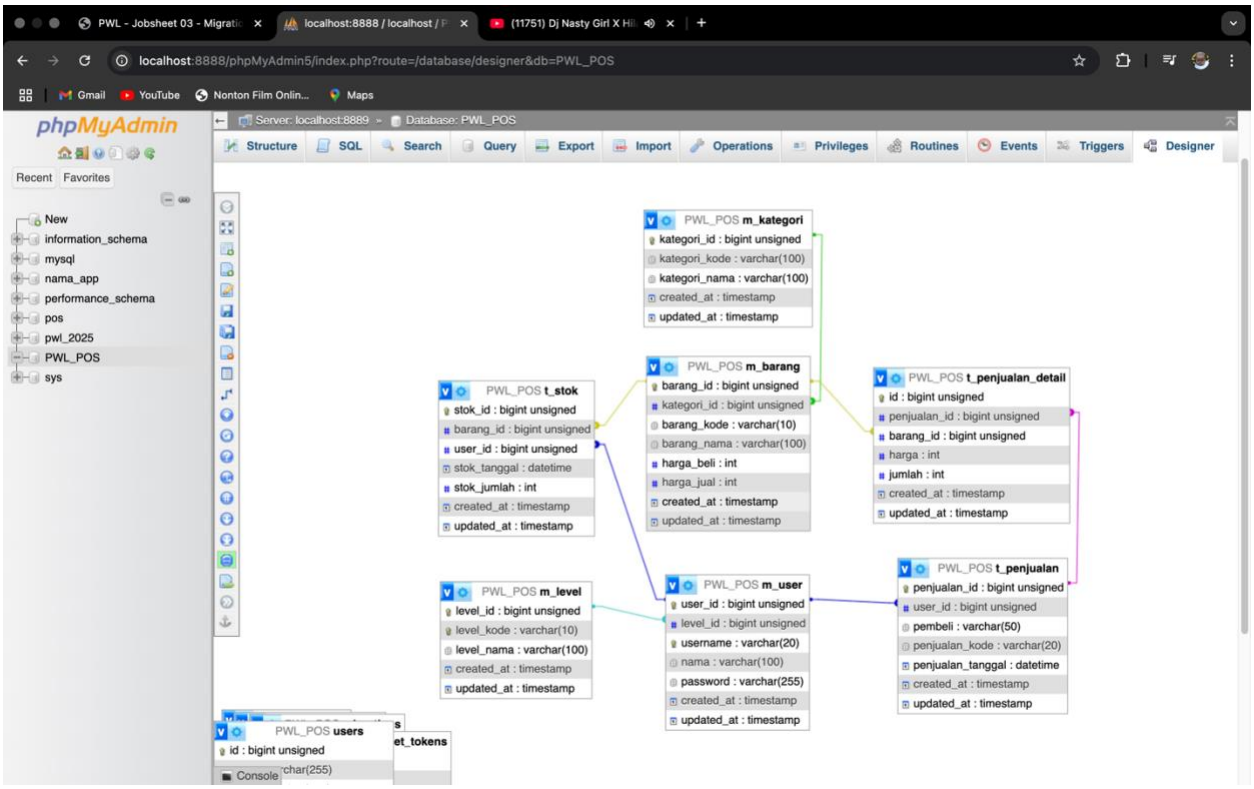
C. In this section we're implementing the **MIGRATIONS** for the table that have the foreign key of the other table that implement the primary key and have the relations with the

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 %
```



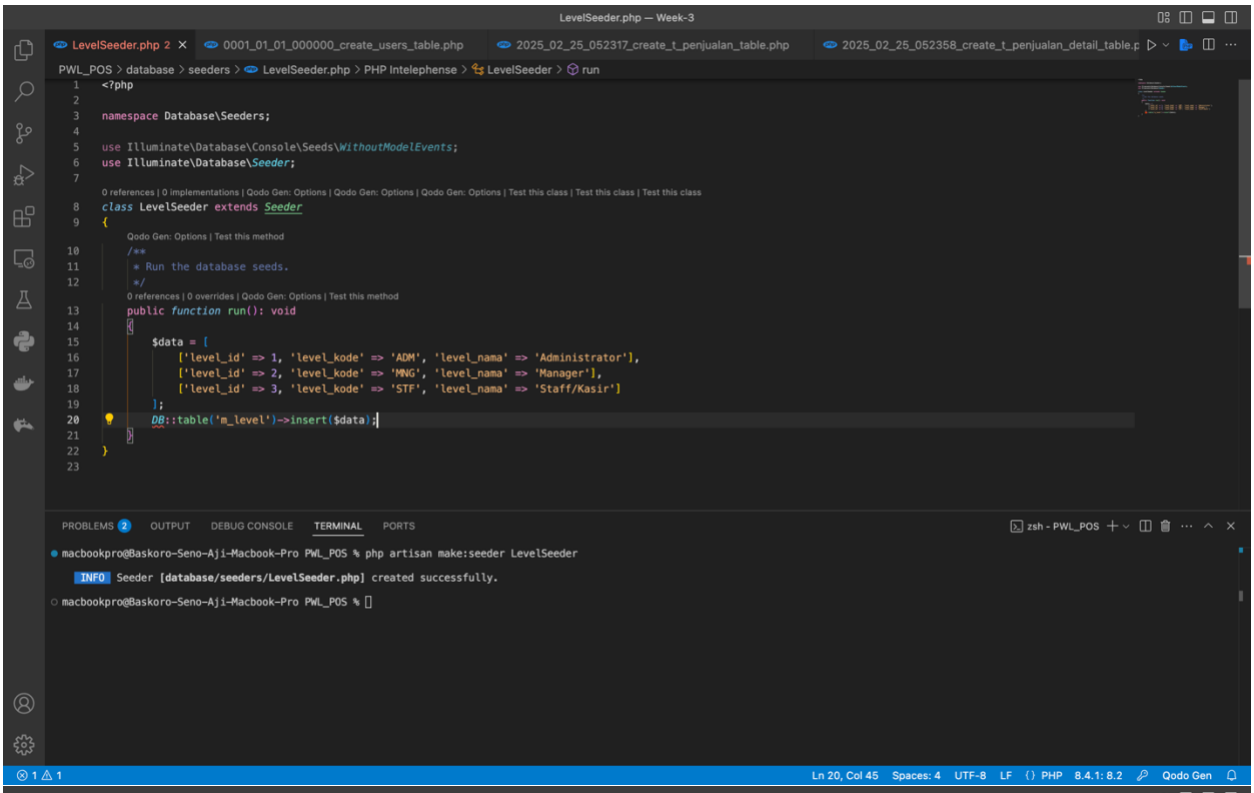


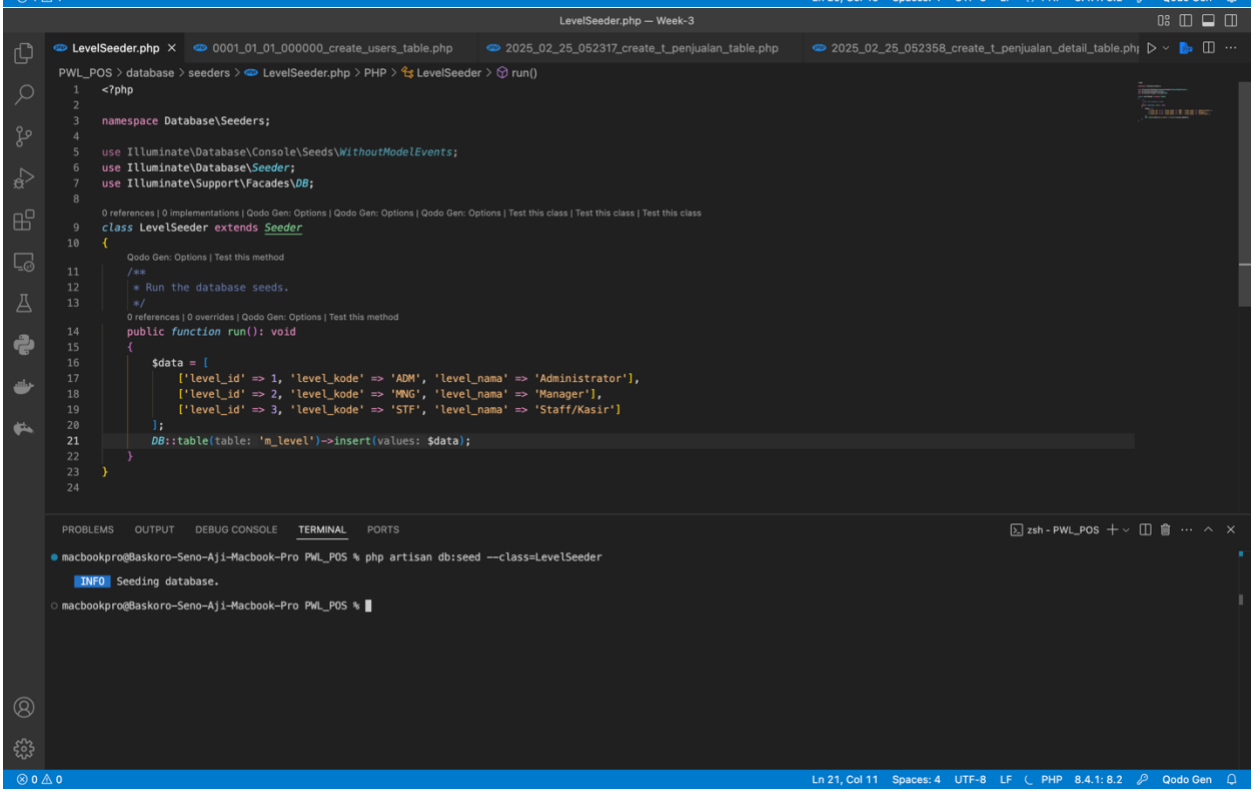
D. This part we're doing the same but for an instance we're also checking it on the PhpMyAdmin Designer page that every table that have the relations with the main table or table master are connected well, this designer picture have proven that we're implementing it well.

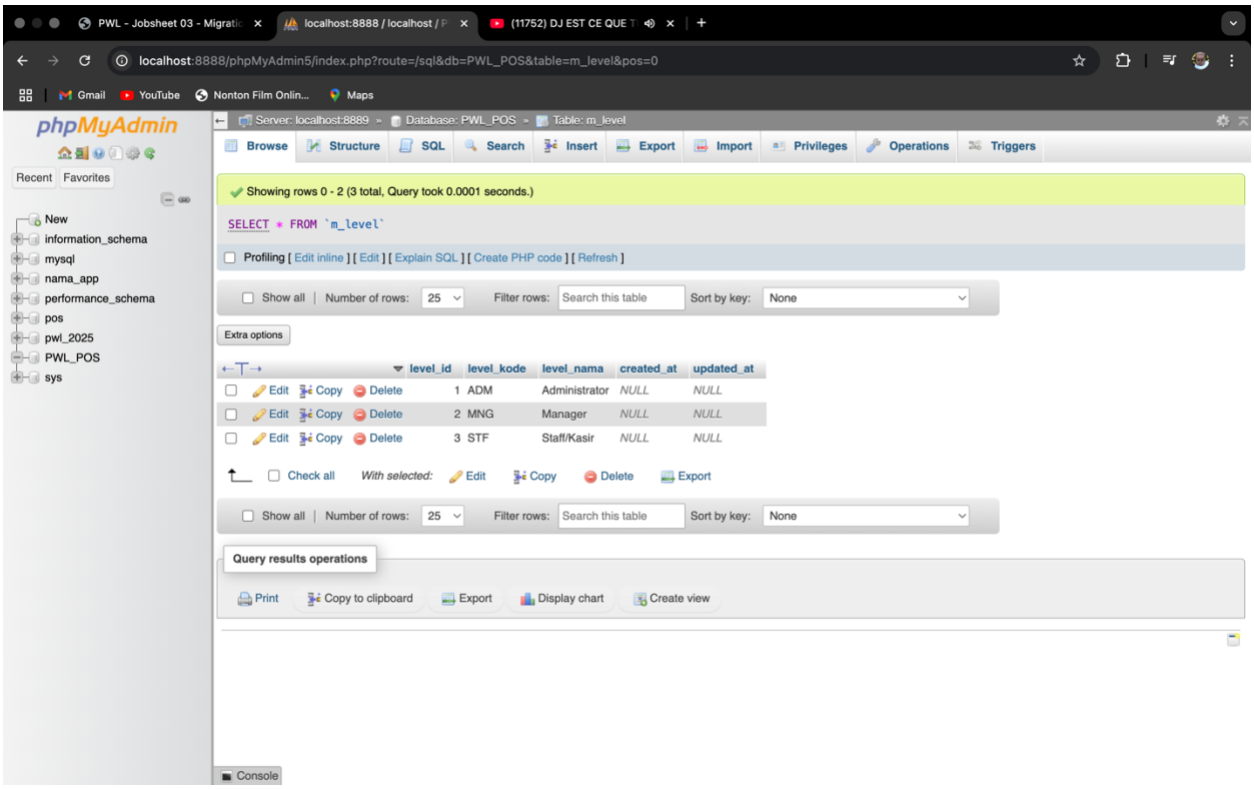
SEEDER

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.p
PWL_POS > database > seeders > LevelSeeder.php > PHP Intelephense > LevelSeeder > run
<?php
namespace Database\Seeders;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
0 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class | Test this class
class LevelSeeder extends Seeder
{
Qodo Gen: Options | Test this method
/**
* Run the database seeds.
*/
0 references | 0 overrides | Qodo Gen: Options | Test this method
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);
}
}
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh - PWL_POS
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 %
⊗ 1 ⚠ 1
Ln 20, Col 45
Spaces: 4
UTF-8
LF
{} PHP 8.4.1: 8.2
Qodo Gen

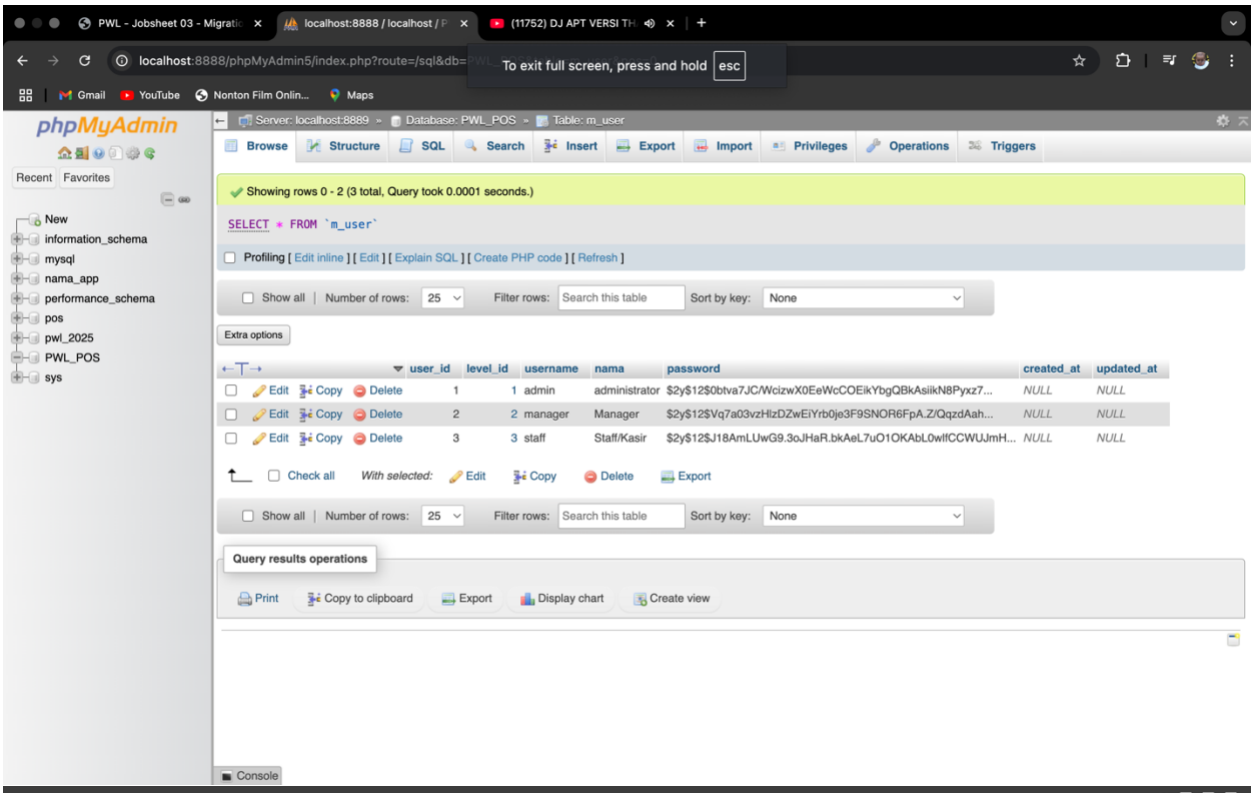

LevelSeeder.php — Week-3
LevelSeeder.php
0001_01_01_000000_create_users_table.php
2025_02_25_052317_create_t_penjualan_table.php
2025_02_25_052358_create_t_penjualan_detail_table.php
PWL_POS > database > seeders > LevelSeeder.php > PHP > LevelSeeder > run()
<?php
namespace Database\Seeders;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use Illuminate\Database\Seeder;
use Illuminate\Support\Facades\DB;
0 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class | Test this class
class LevelSeeder extends Seeder
{
Qodo Gen: Options | Test this method
/**
* Run the database seeds.
*/
0 references | 0 overrides | Qodo Gen: Options | Test this method
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(table: 'm_level')->insert(values: $data);
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh - PWL_POS
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 %
⊗ 0 ⚠ 0
Ln 21, Col 11
Spaces: 4
UTF-8
LF
PHP 8.4.1: 8.2
Qodo Gen

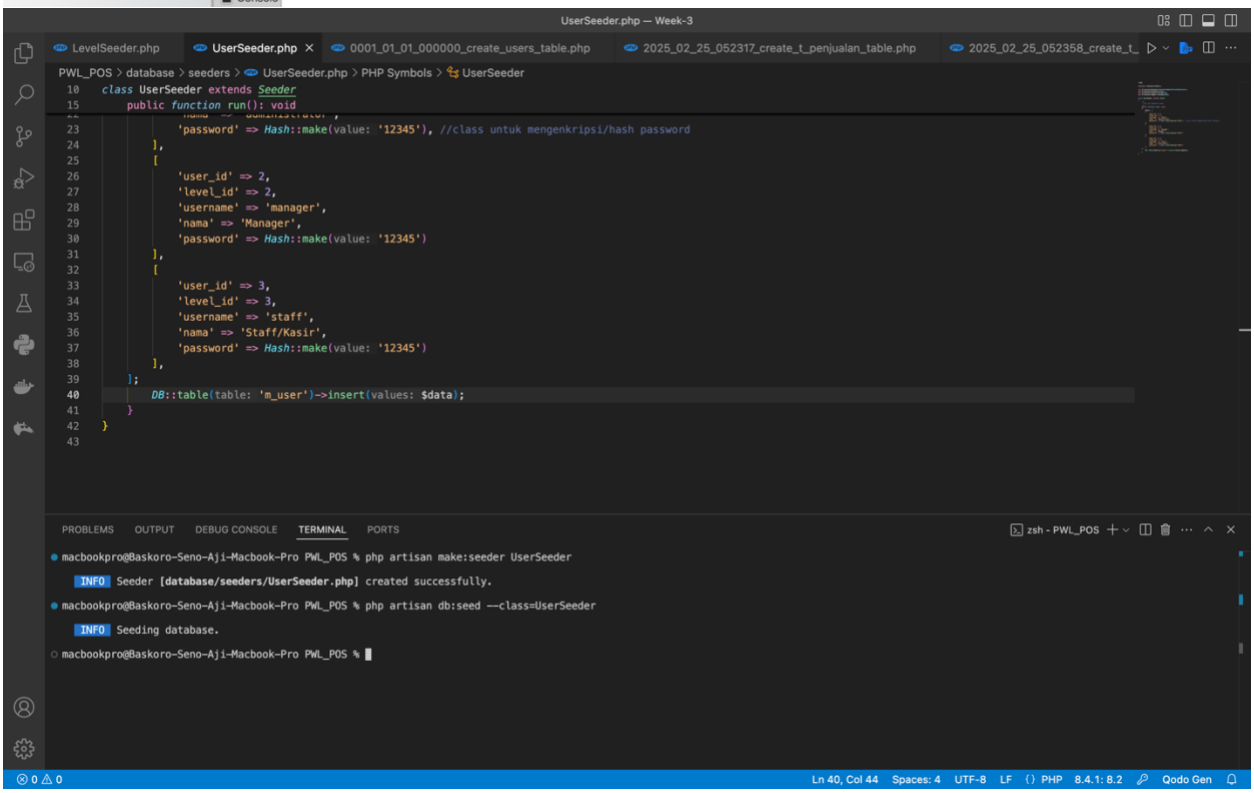


A. In this section we're implementing **SEEDER** feature to inserting the data of every table. in this particular section we insert the data for the m_level by adding into three type of user, to add an use the seeder we're using particular syntaxes that are :

- 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**

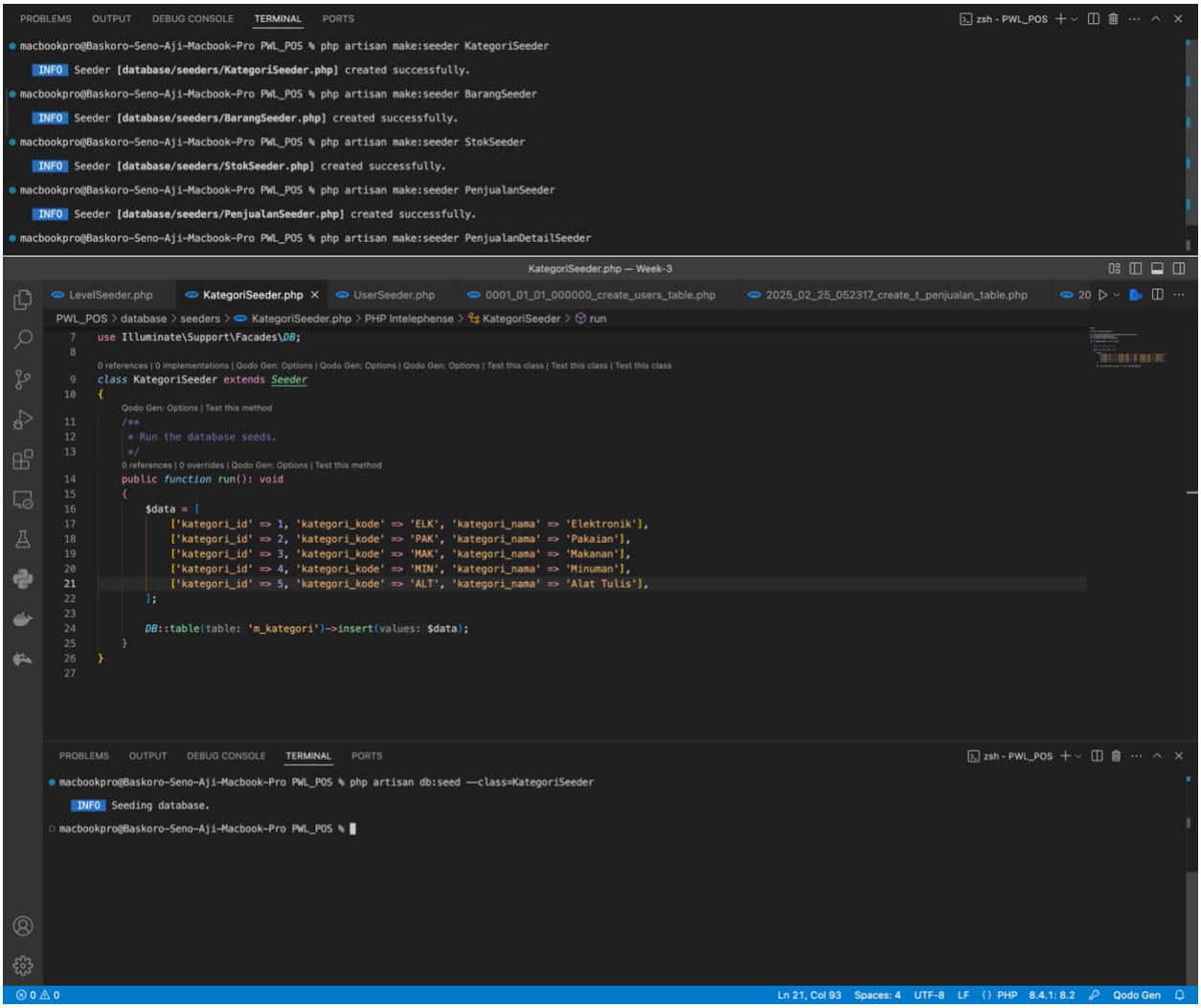

PWL - Jobsheet 03 - Migrati
localhost:8888 / localhost / P
(11752) DJ APT VERSI TH
localhost:8888/phpMyAdmin5/index.php?route=/sql&db=PWL_
To exit full screen, press and hold esc
Gmail
YouTube
Nonton Film Onlin...
Maps
phpMyAdmin
Recent
Favorites
New
information_schema
mysql
nama_app
performance_schema
pos
pwl_2025
PWL_POS
sys
Server: localhost:8889 » Database: PWL_POS » Table: m_user
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
Showing rows 0 - 2 (3 total, Query took 0.0001 seconds.)
SELECT * FROM `m_user`
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
user_id level_id username nama password created_at updated_at
Edit Copy Delete 1 1 admin administrator $2y$12$0btva7JC/WcizwX0EeWcCOEikYbgQBkAsiikN8Pyxz7... NULL NULL
Edit Copy Delete 2 2 manager Manager $2y$12$Vq7a03vzHlzDZwEiYrb0je3F9SNOR6FpA.Z/QqzdAah... NULL NULL
Edit Copy Delete 3 3 staff Staff/Kasir $2y$12$J18AmLUwG9.3oJHaR.bkAeL7uO1OKAbL0wIfCCWUJmH... 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

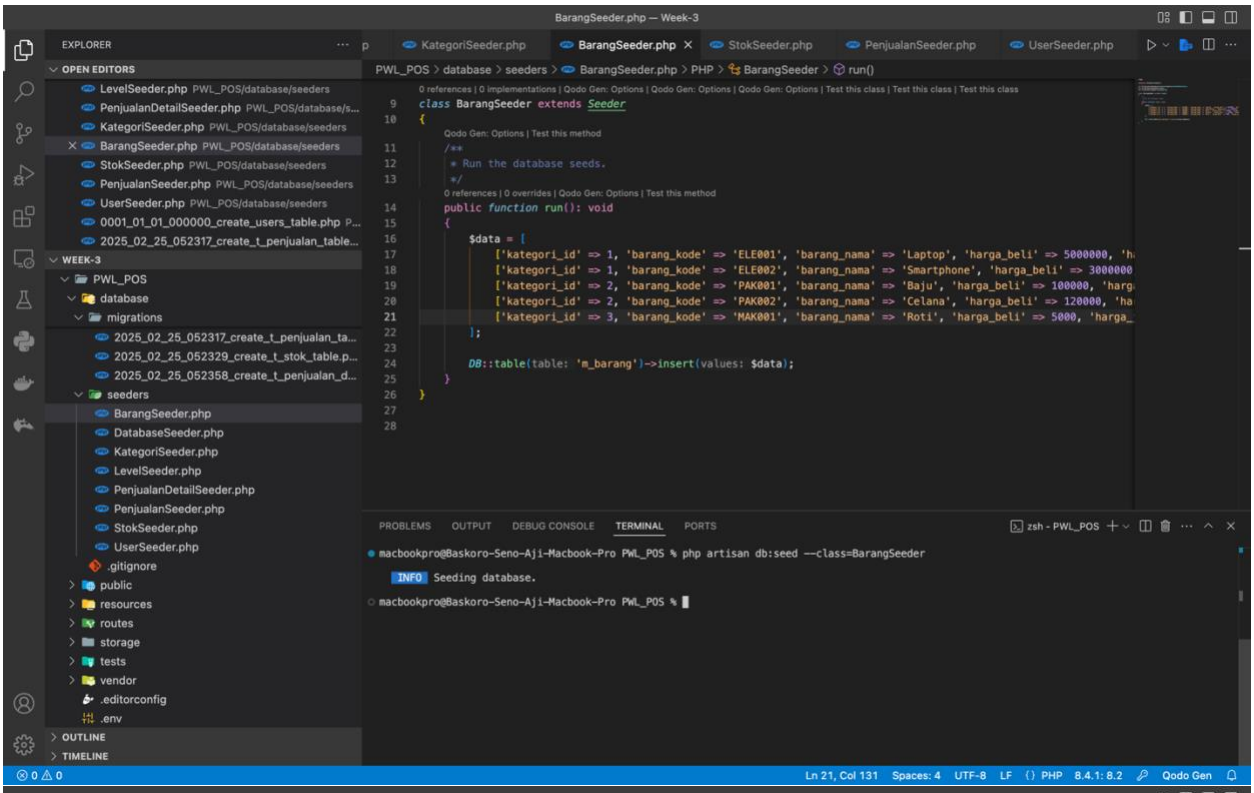

UserSeeder.php — Week-3
LevelSeeder.php
UserSeeder.php
0001_01_01_000000_create_users_table.php
2025_02_25_052317_create_t_penjualan_table.php
2025_02_25_052358_create_t
PWL_POS > database > seeders > UserSeeder.php > PHP Symbols > UserSeeder
class UserSeeder extends Seeder
public function run(): void
'password' => Hash::make(value: '12345'), //class untuk mengenkripsi/hash password
'user_id' => 2,
'level_id' => 2,
'username' => 'manager',
'nama' => 'Manager',
'password' => Hash::make(value: '12345')
'user_id' => 3,
'level_id' => 3,
'username' => 'staff',
'nama' => 'Staff/Kasir',
'password' => Hash::make(value: '12345')
DB::table(table: 'm_user')->insert(values: $data);
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
zsh - PWL_POS
macbookpro@Baskoro-Seno-Aji-Macbook-Pro PWL_POS % php artisan make:seeder UserSeeder
INFO Seeder [database/seeders/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 Qodo Gen

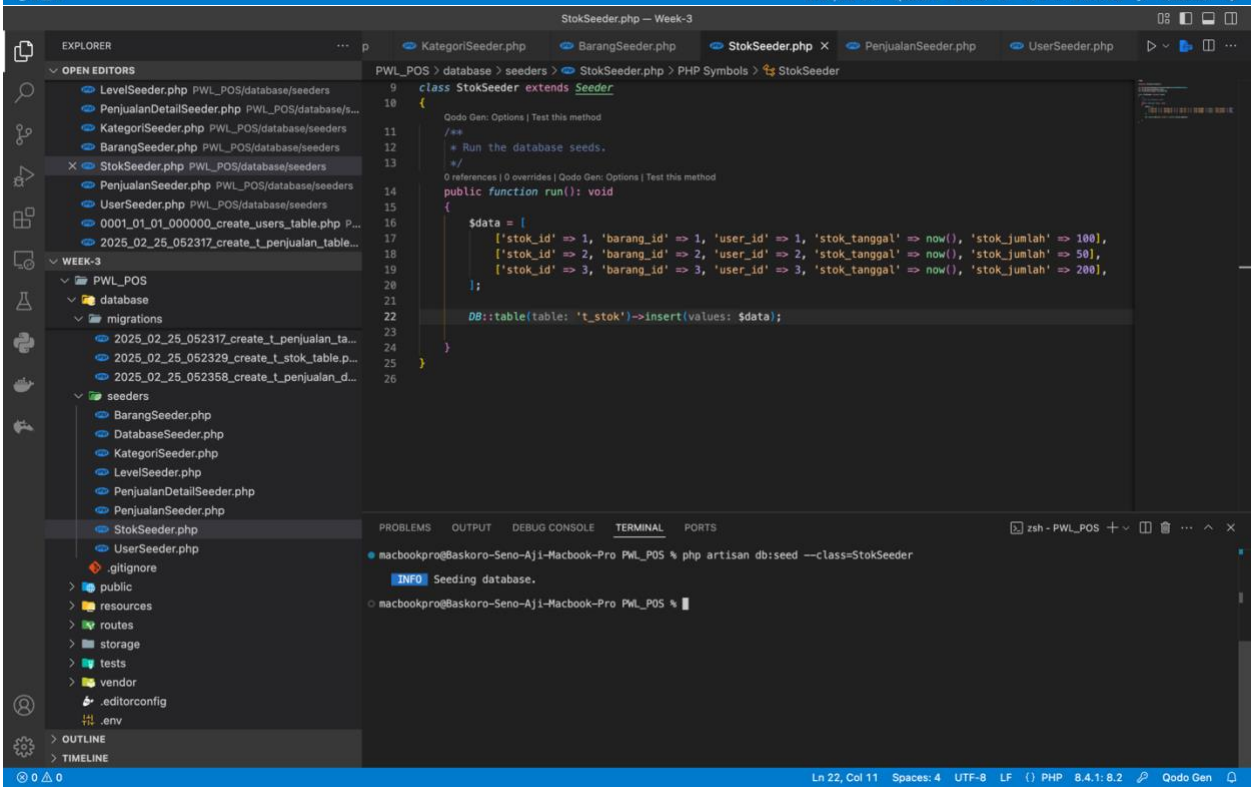
B. Implementing the same syntaxes to create the user by referring to the equivalent category to be insert on the databases

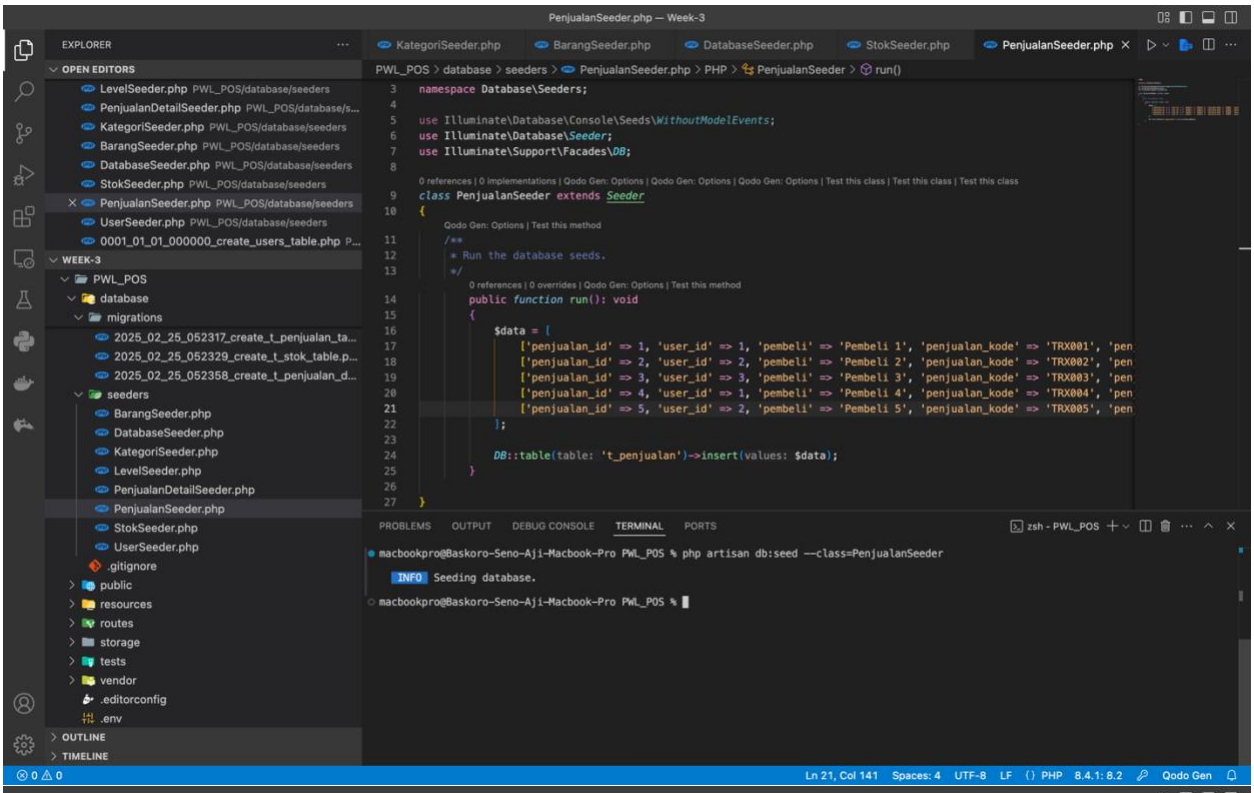
10. Sekarang coba kalian masukkan data seeder untuk table yang lain, dengan ketentuan seperti berikut

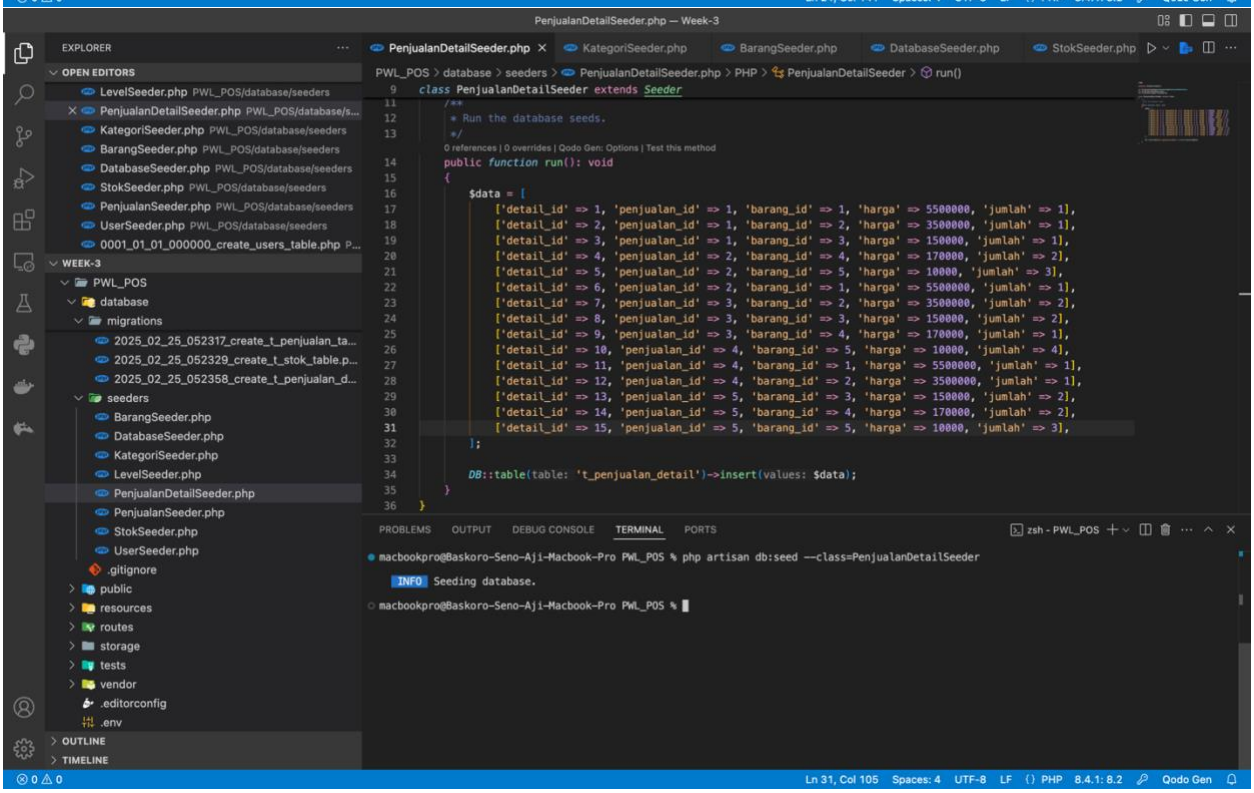
No	Nama Tabel	Jumlah Data	Keterangan
1	m_kategori	5	5 kategori barang
2	m_barang	10	10 barang yang berbeda
3	t_stok	10	Stok untuk 10 barang
4	t_penjualan	10	10 transaksi penjualan
5	t_penjualan_detail	30	3 barang untuk setiap transaksi penjualan




BarangSeeder.php — Week-3
EXPLORER
OPEN EDITORS
LevelSeeder.php PWL_POS/database/seeders
PenjualanDetailSeeder.php PWL_POS/database/s...
KategoriSeeder.php PWL_POS/database/seeders
BarangSeeder.php PWL_POS/database/seeders
StokSeeder.php PWL_POS/database/seeders
PenjualanSeeder.php PWL_POS/database/seeders
UserSeeder.php PWL_POS/database/seeders
0001_01_01_000000_create_users_table.php P...
2025_02_25_052317_create_t_penjualan_table...
WEEK-3
PWL_POS
database
migrations
2025_02_25_052317_create_t_penjualan_ta...
2025_02_25_052329_create_t_stok_table.p...
2025_02_25_052358_create_t_penjualan_d...
seeders
BarangSeeder.php
DatabaseSeeder.php
KategoriSeeder.php
LevelSeeder.php
PenjualanDetailSeeder.php
PenjualanSeeder.php
StokSeeder.php
UserSeeder.php
.gitignore
public
resources
routes
storage
tests
vendor
.editorconfig
.env
OUTLINE
TIMELINE
KategoriSeeder.php
BarangSeeder.php
StokSeeder.php
PenjualanSeeder.php
UserSeeder.php
PWL_POS > database > seeders > BarangSeeder.php > PHP > BarangSeeder > run()
0 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class | Test this class
class BarangSeeder extends Seeder
{
Qodo Gen: Options | Test this method
/**
* Run the database seeds.
*/
0 references | 0 overrides | Qodo Gen: Options | Test this method
public function run(): void
{
$data = [
['kategori_id' => 1, 'barang_kode' => 'ELE001', 'barang_nama' => 'Laptop', 'harga_beli' => 5000000, 'h
['kategori_id' => 1, 'barang_kode' => 'ELE002', 'barang_nama' => 'Smartphone', 'harga_beli' => 3000000
['kategori_id' => 2, 'barang_kode' => 'PAK001', 'barang_nama' => 'Baju', 'harga_beli' => 100000, 'harg
['kategori_id' => 2, 'barang_kode' => 'PAK002', 'barang_nama' => 'Celana', 'harga_beli' => 120000, 'ha
['kategori_id' => 3, 'barang_kode' => 'MAK001', 'barang_nama' => 'Roti', 'harga_beli' => 5000, 'harga_
];
DB::table(table: 'm_barang')->insert(values: $data);
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh - PWL_POS
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 %
0 0
Ln 21, Col 131
Spaces: 4
UTF-8
LF
PHP 8.4.1: 8.2
Qodo Gen


StokSeeder.php — Week-3
EXPLORER
OPEN EDITORS
LevelSeeder.php PWL_POS/database/seeders
PenjualanDetailSeeder.php PWL_POS/database/s...
KategoriSeeder.php PWL_POS/database/seeders
BarangSeeder.php PWL_POS/database/seeders
StokSeeder.php PWL_POS/database/seeders
PenjualanSeeder.php PWL_POS/database/seeders
UserSeeder.php PWL_POS/database/seeders
0001_01_01_000000_create_users_table.php P...
2025_02_25_052317_create_t_penjualan_table...
WEEK-3
PWL_POS
database
migrations
2025_02_25_052317_create_t_penjualan_ta...
2025_02_25_052329_create_t_stok_table.p...
2025_02_25_052358_create_t_penjualan_d...
seeders
BarangSeeder.php
DatabaseSeeder.php
KategoriSeeder.php
LevelSeeder.php
PenjualanDetailSeeder.php
PenjualanSeeder.php
StokSeeder.php
UserSeeder.php
.gitignore
public
resources
routes
storage
tests
vendor
.editorconfig
.env
OUTLINE
TIMELINE
KategoriSeeder.php
BarangSeeder.php
StokSeeder.php
PenjualanSeeder.php
UserSeeder.php
PWL_POS > database > seeders > StokSeeder.php > PHP Symbols > StokSeeder
class StokSeeder extends Seeder
{
Qodo Gen: Options | Test this method
/**
* Run the database seeds.
*/
0 references | 0 overrides | Qodo Gen: Options | Test this method
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(table: 't_stok')->insert(values: $data);
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
zsh - PWL_POS
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 %
0 0
Ln 22, Col 11
Spaces: 4
UTF-8
LF
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']);
```

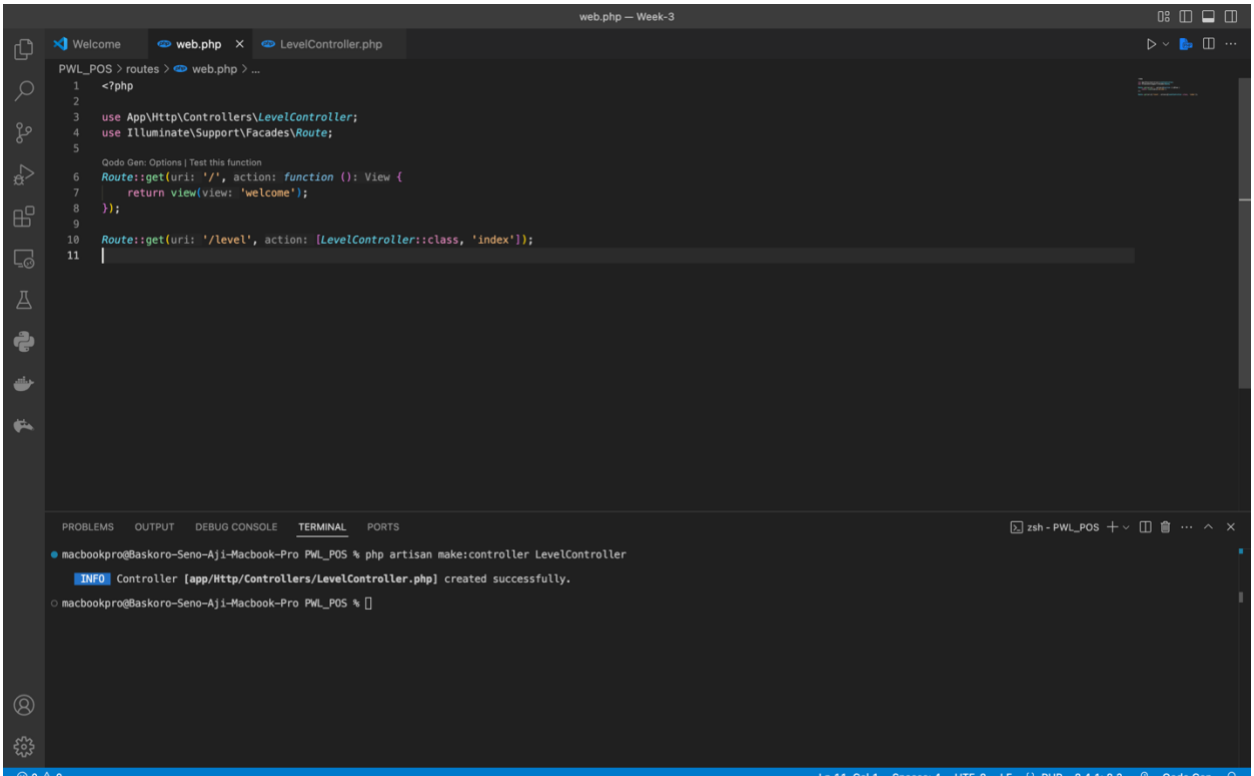
d. `DB::delete()`

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 :



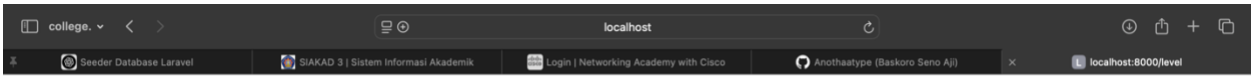
Welcome | web.php | LevelController.php

PWL_POS > app > Http > Controllers > LevelController.php > PHP Intelephense > LevelController > index

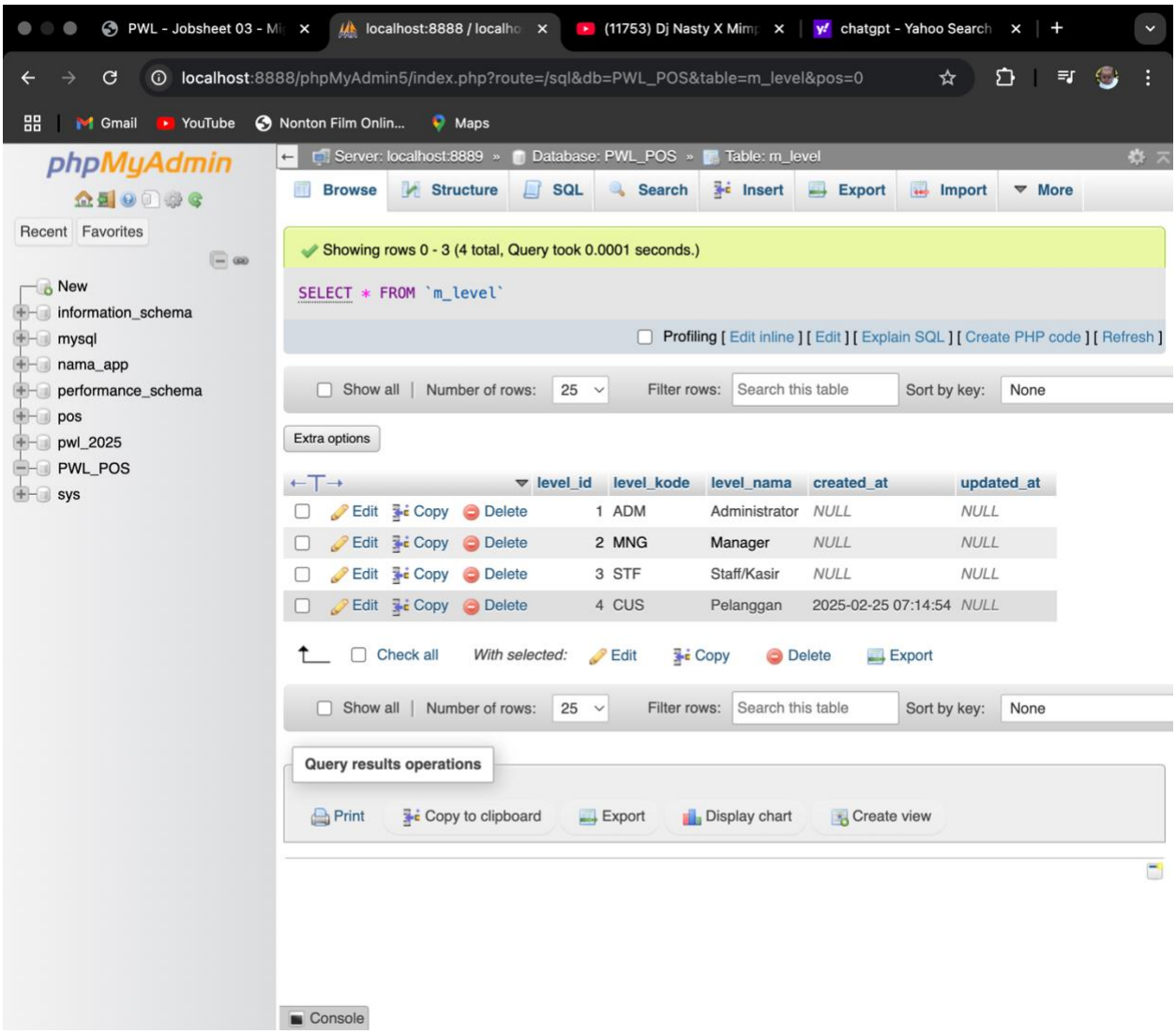
```php
<?php

namespace App\Http\Controllers;

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

2 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class | Test this class
class LevelController extends Controller
{
    1 reference | 0 overrides | Qodo Gen: Options | Qodo Gen: Options | Test this method | Test this method
    public function index(): string
    {
        DB::insert(query: 'insert into m_level(level_kode, level_nama, created_at) values(?, ?,

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS zsh - PWL_POS

```
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 %
```

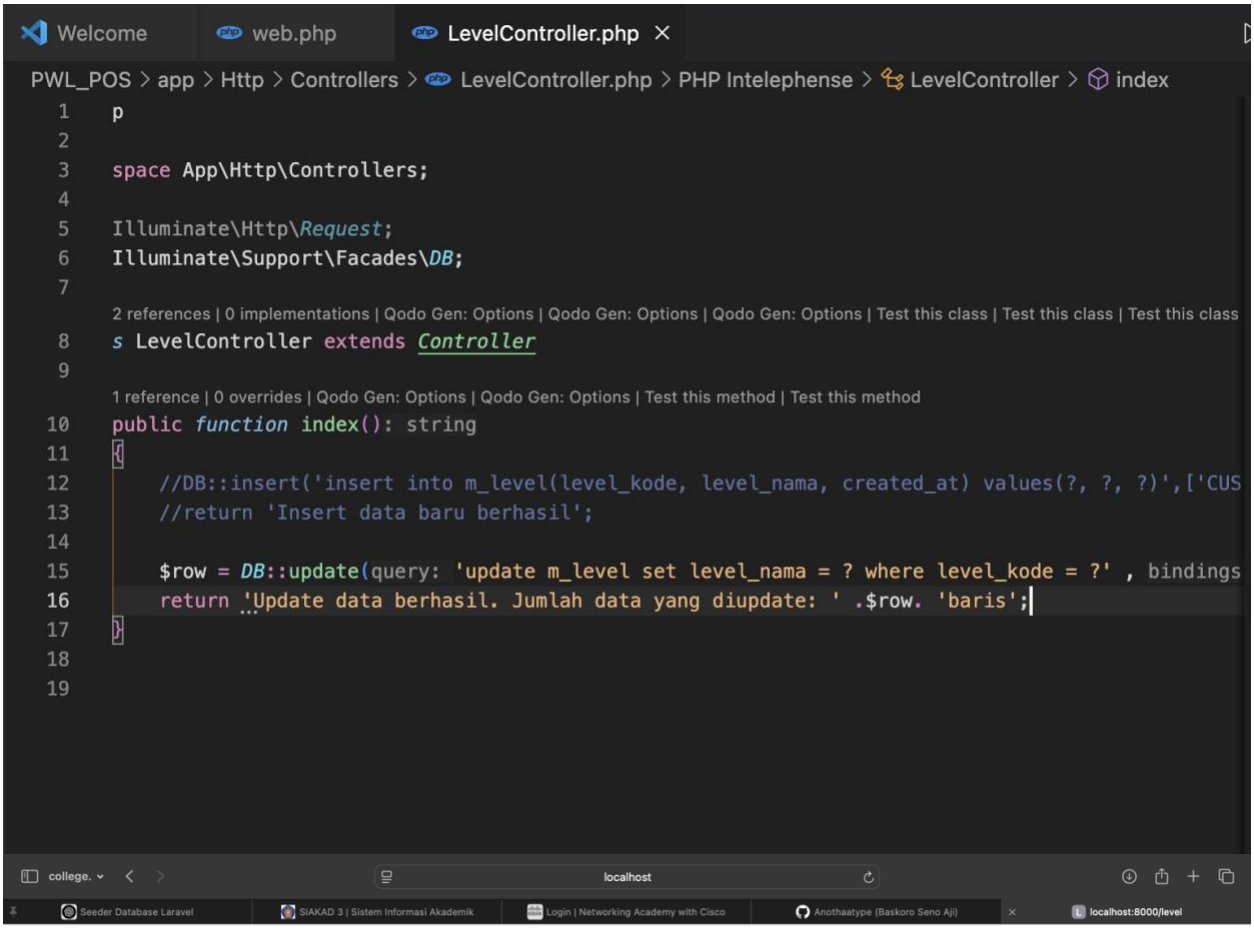


Insert data baru berhasil

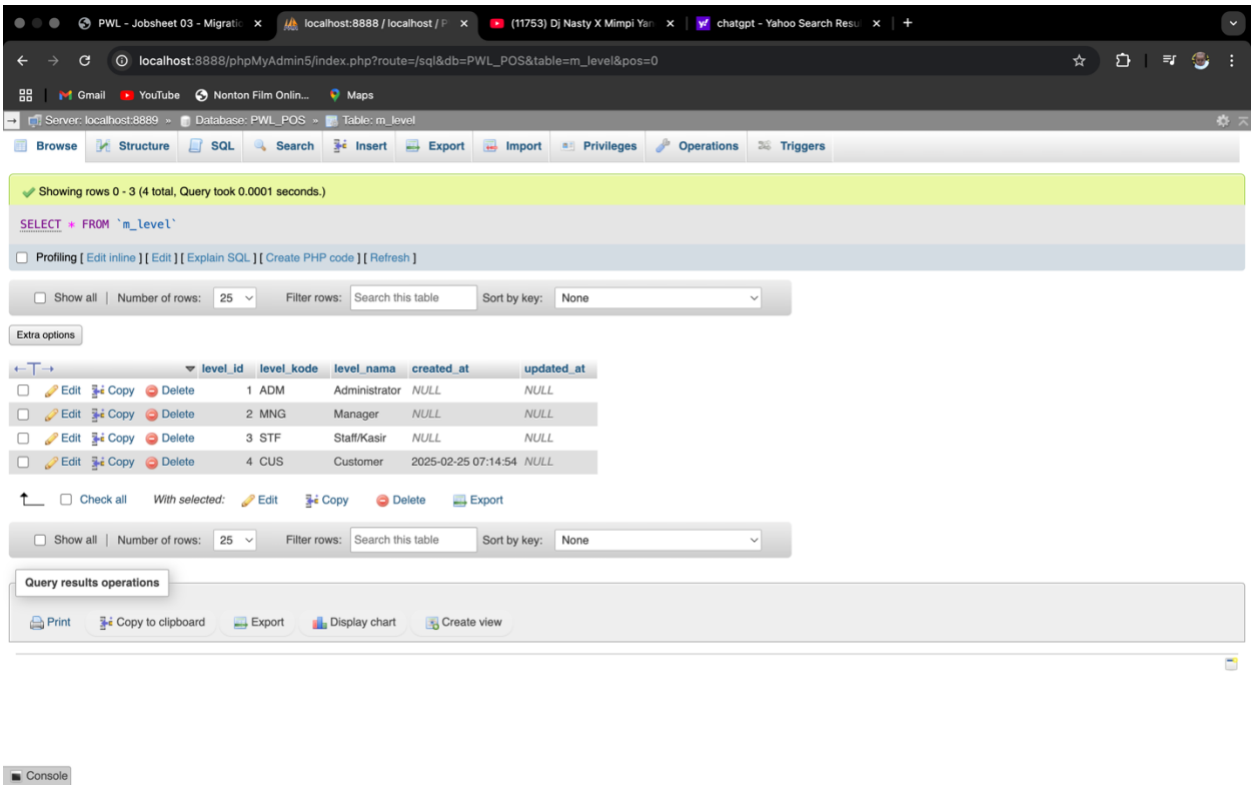


A. In this section we do some several steps to implement the DB Façade statement, first we need to create the controller by implementing php artisan make:controller (NameController) where in this section are focusing for the M_LEVEL table, second we need to set and connect the route to the controller. 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



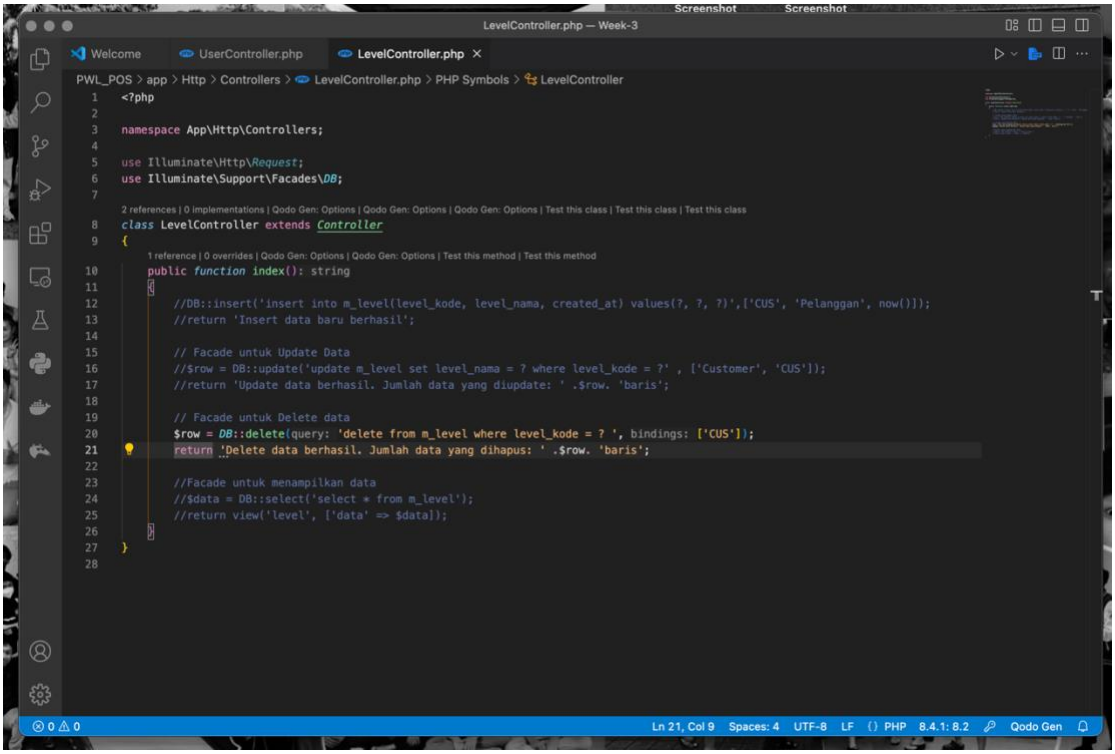
Update data berhasil. Jumlah data yang diupdate: 1baris

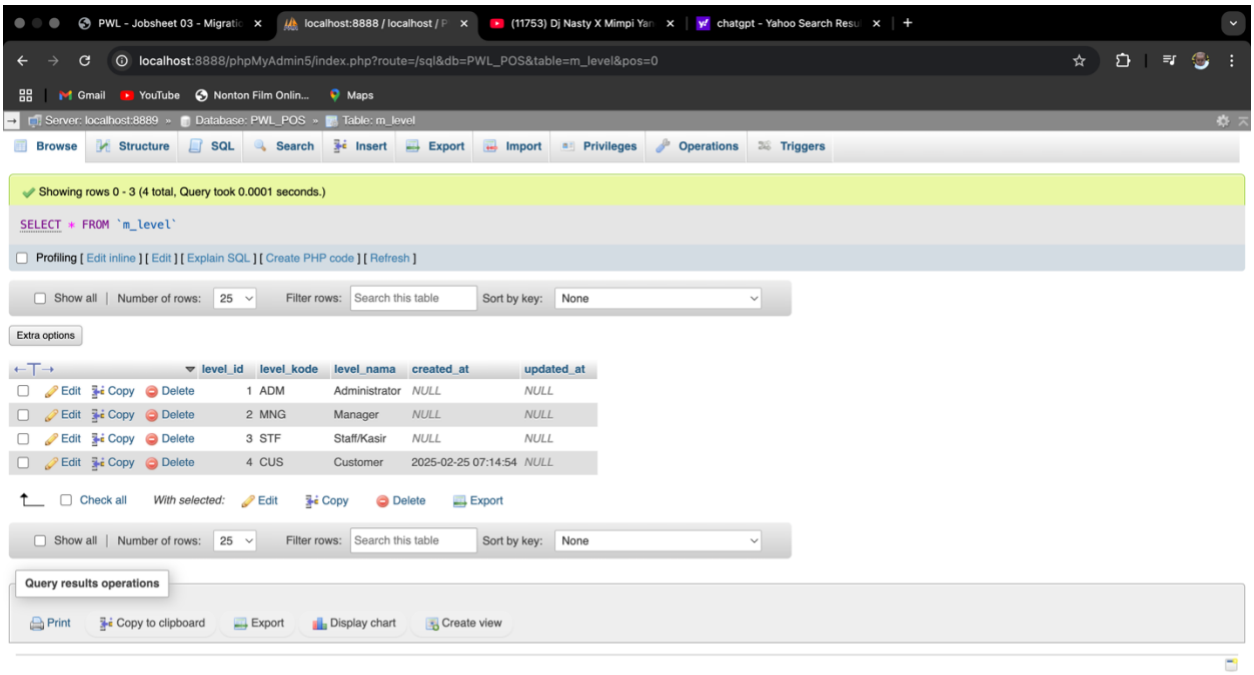


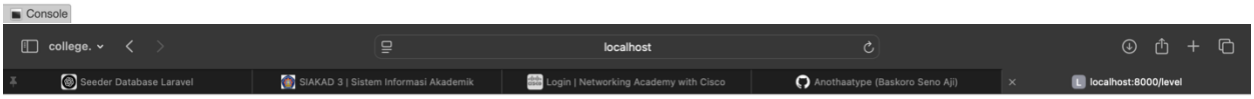
Console

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



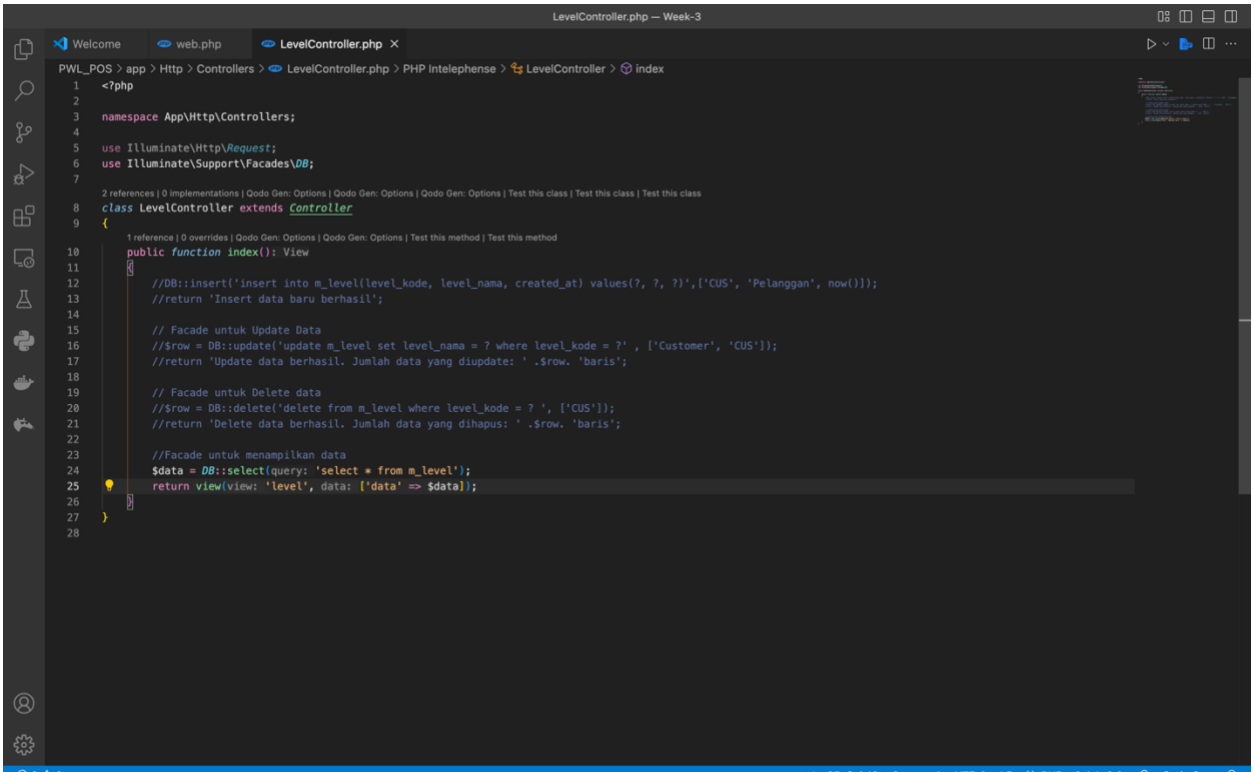


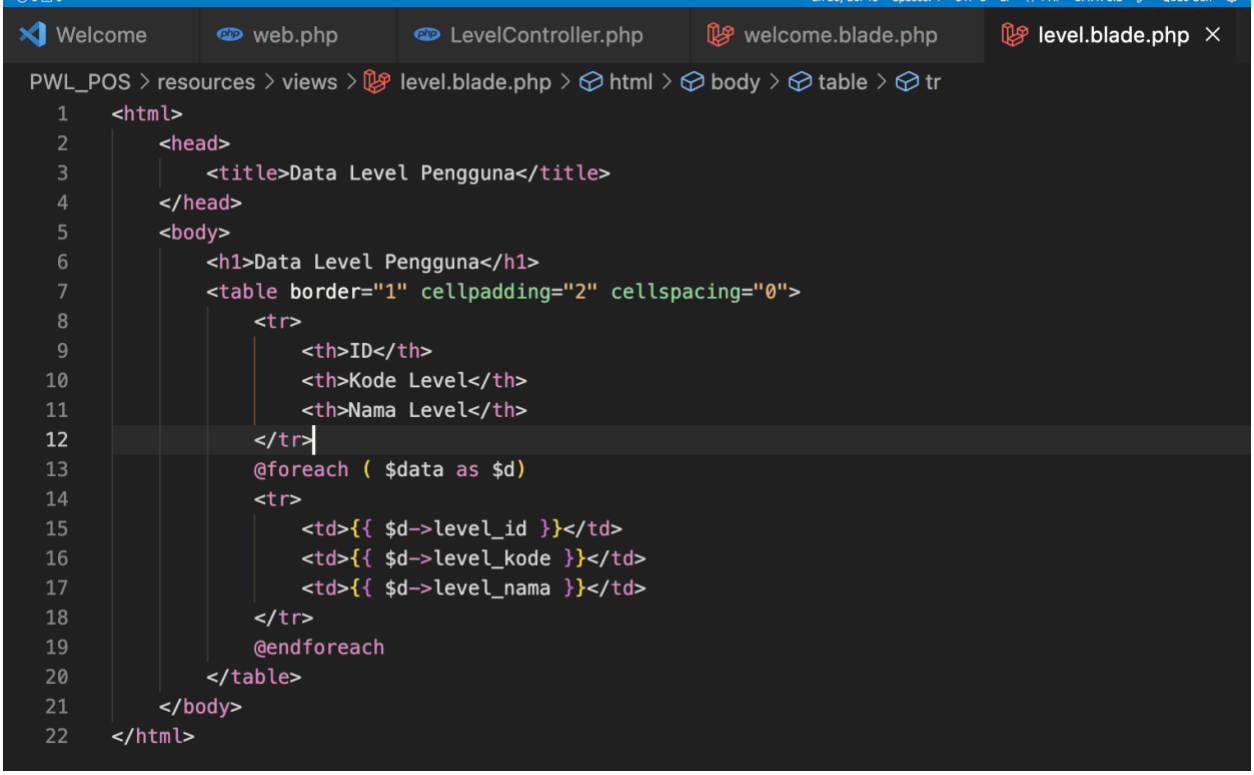


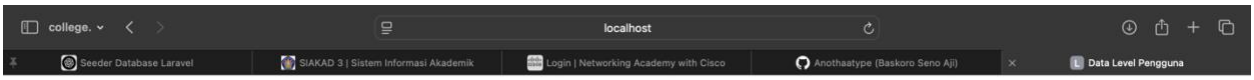
Delete data berhasil. Jumlah data yang dihapus: 1baris

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

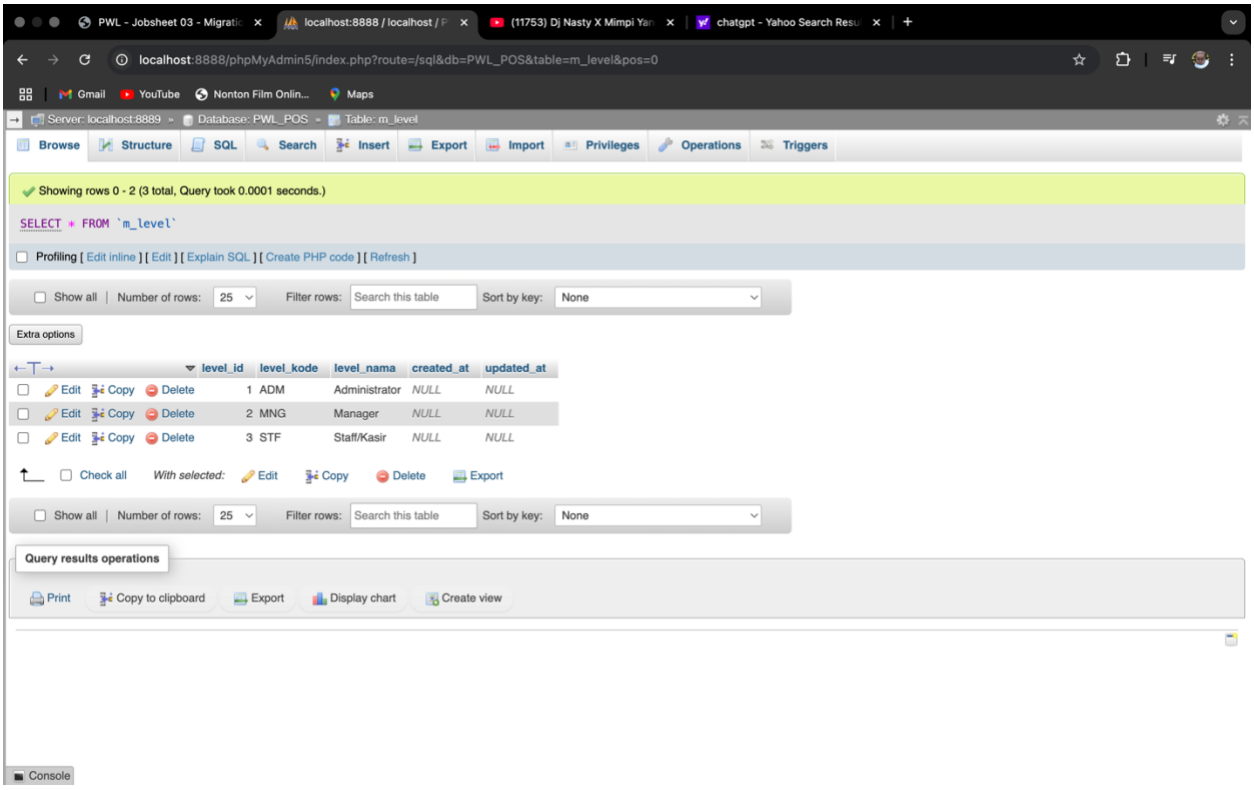






Data Level Pengguna

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



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;
```

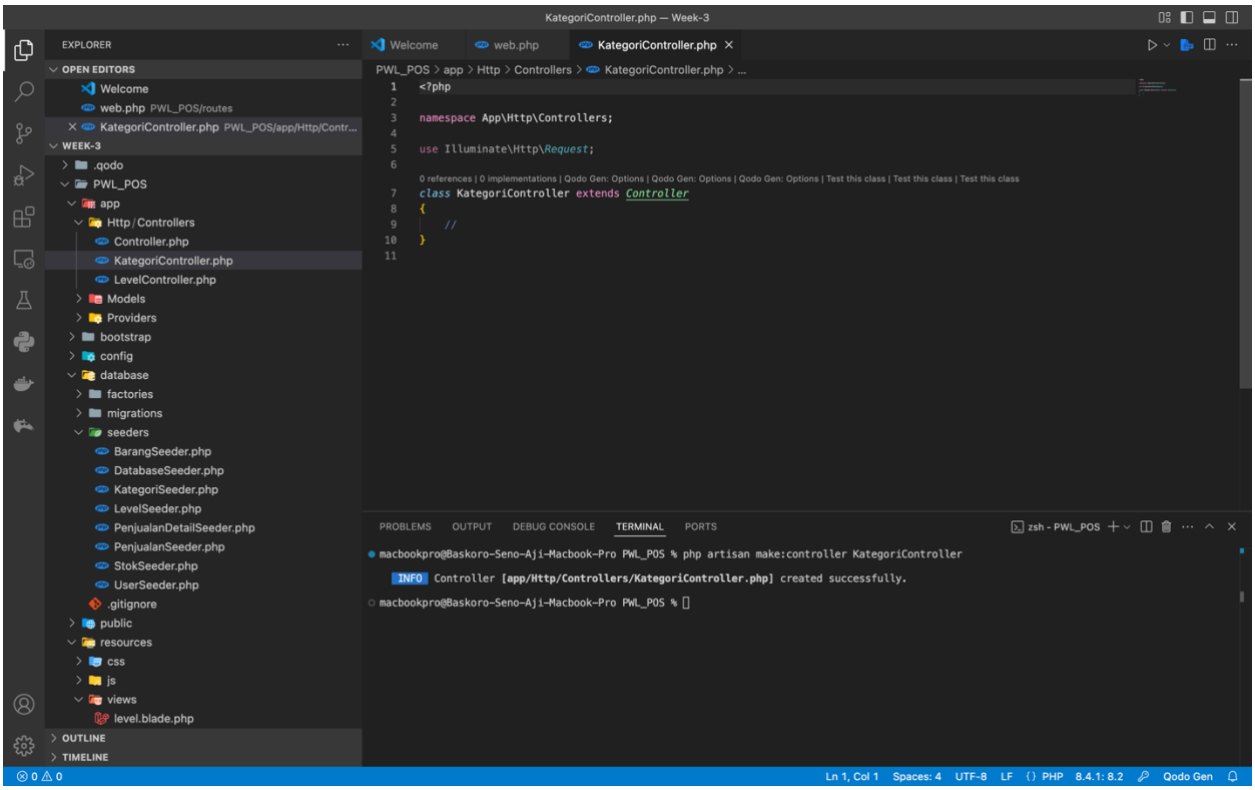
d. Perintah untuk ambil data

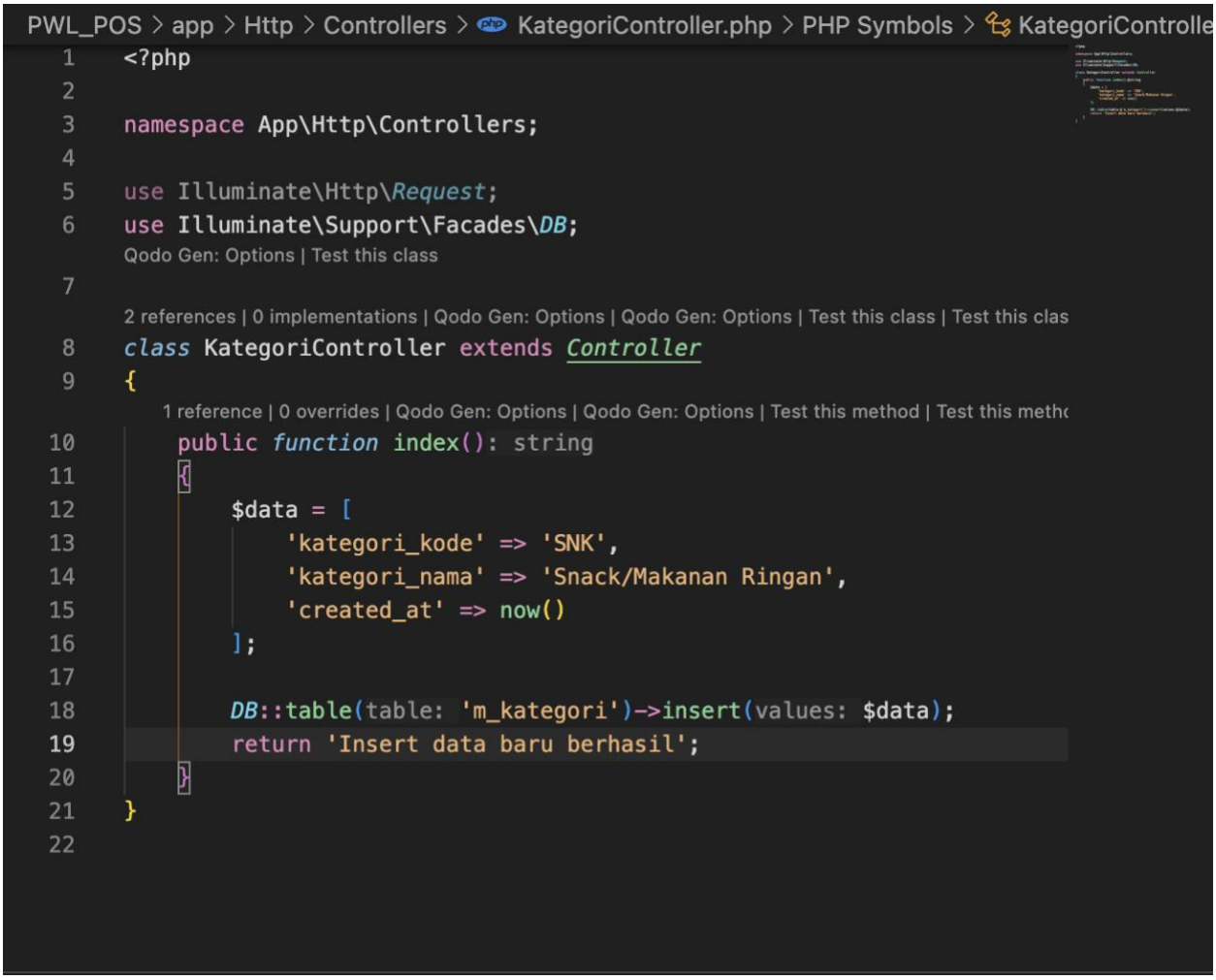
Method Query Builder	**Query yang dihasilkan**
`DB::table('m_kategori')->get();`	`select * from m_kategori`
`DB::table('m_kategori')` `->where('kategori_id', 1)->get();`	`select * from m_kategori where kategori_id = 1;`

`DB::table('m_kategori')`
`->select('kategori_kode')`
`->where('kategori_id', 1)->get();`
	`select kategori_kode from m_kategori where kategori_id = 1;`

Praktikum 5 – Implementasi Query Builder

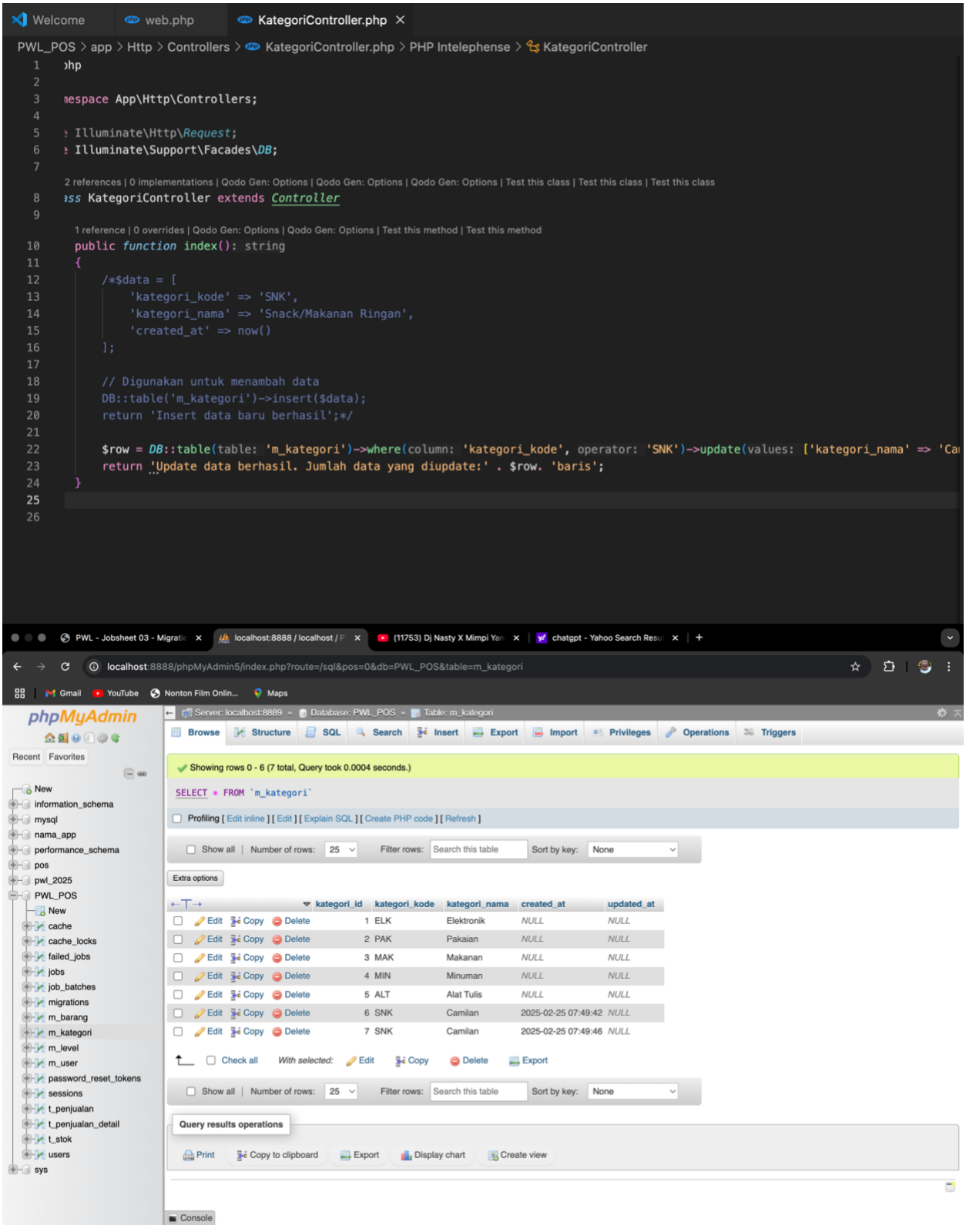
- Query Builder Insert

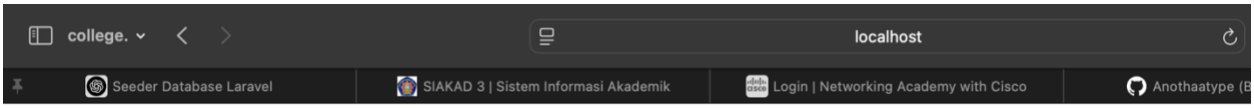




A. in this terms we insert all the data in the $data variable but to call or use it we use the index function where the already connect the route into the KategoriController so in that terms when we use /kategori we can see the output of the return where it is “Insert data baru berhasil”

- Query Builder Update

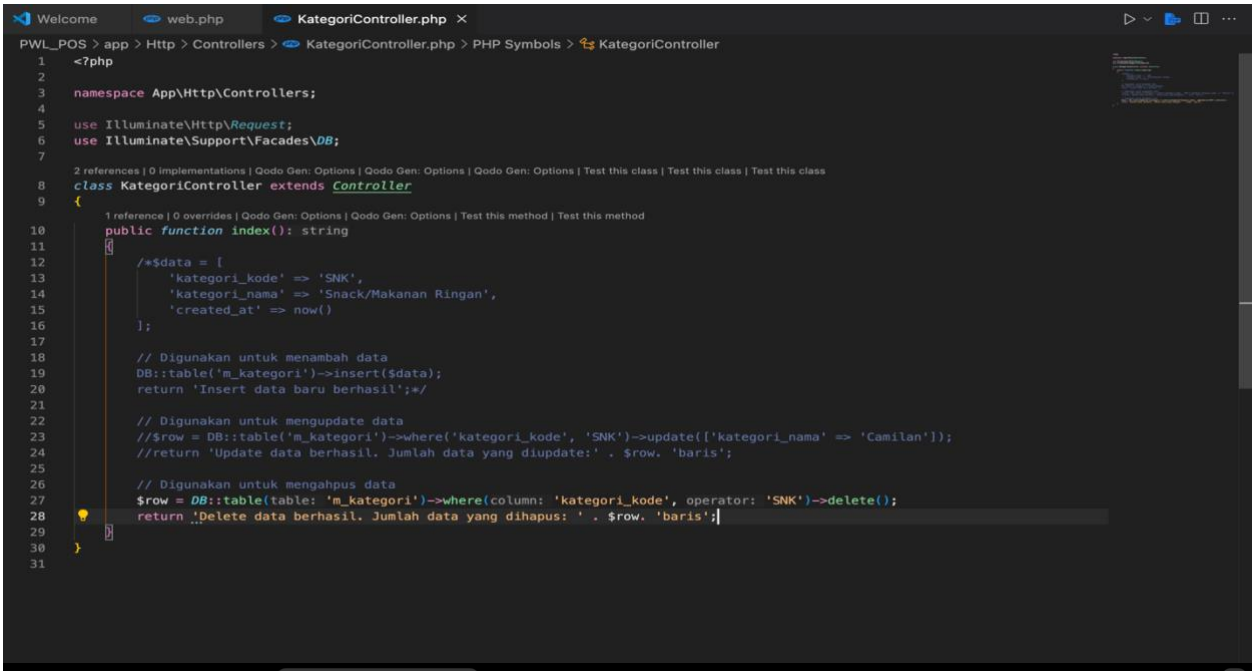

Welcome
web.php
KategoriController.php
PWL_POS > app > Http > Controllers > KategoriController.php > PHP Intelephense > KategoriController
2 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class | Test this class
1 reference | 0 overrides | Qodo Gen: Options | Qodo Gen: Options | Test this method | Test this method
public function index(): string
{
/*$data = [
'kategori_kode' => 'SNK',
'kategori_nama' => 'Snack/Makanan Ringan',
'created_at' => now()
];
// Digunakan untuk menambah data
DB::table('m_kategori')->insert($data);
return 'Insert data baru berhasil';*/
$row = DB::table(table: 'm_kategori')->where(column: 'kategori_kode', operator: 'SNK')->update(values: ['kategori_nama' => 'Ca
return 'Update data berhasil. Jumlah data yang diupdate:' . $row. 'baris';
}
localhost:8888/phpMyAdmin5/index.php?route=/sql&pos=0&db=PWL_POS&table=m_kategori
phpMyAdmin
Server: localhost:8889 » Database: PWL_POS » Table: m_kategori
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Triggers
Showing rows 0 - 6 (7 total, Query took 0.0004 seconds.)
SELECT * FROM `m_kategori`
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
kategori_id kategori_kode kategori_nama created_at updated_at
1 ELK Elektronik NULL NULL
2 PAK Pakaian NULL NULL
3 MAK Makanan NULL NULL
4 MIN Minuman NULL NULL
5 ALT Alat Tulis NULL NULL
6 SNK Camilan 2025-02-25 07:49:42 NULL
7 SNK Camilan 2025-02-25 07:49:46 NULL
Check all With selected: Edit Copy Delete Export
Query results operations
Print Copy to clipboard Export Display chart Create view
Console

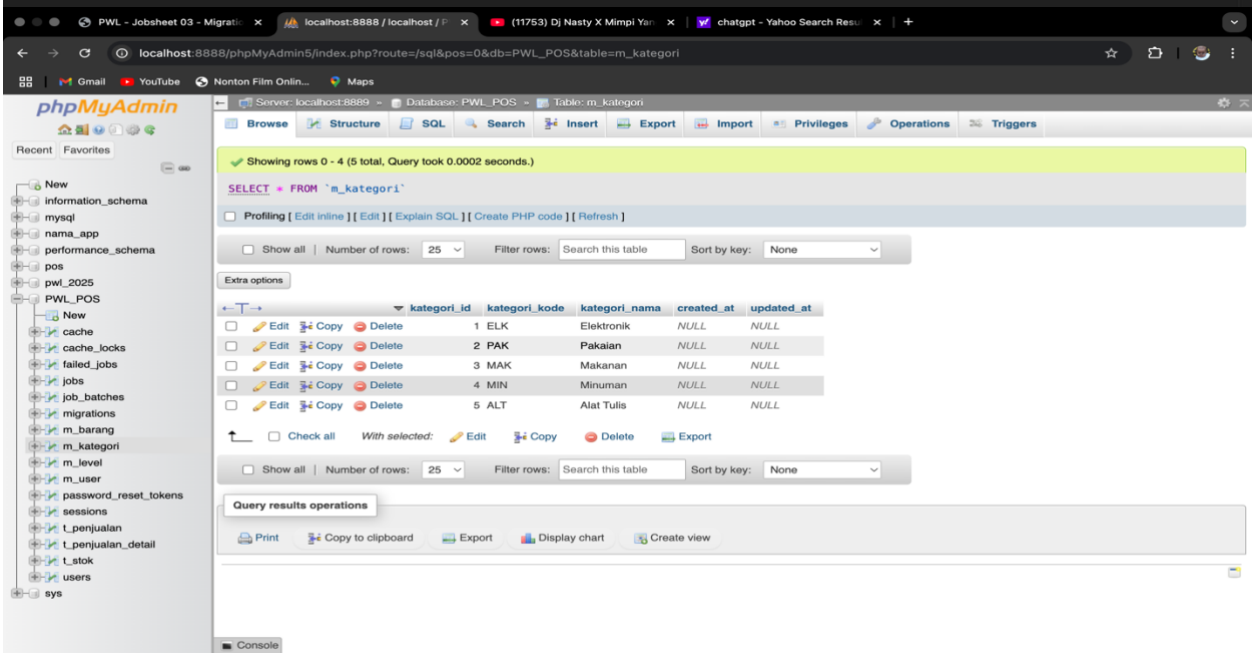


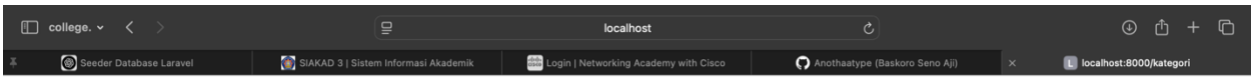
Update data berhasil. Jumlah data yang diupdate:2baris

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



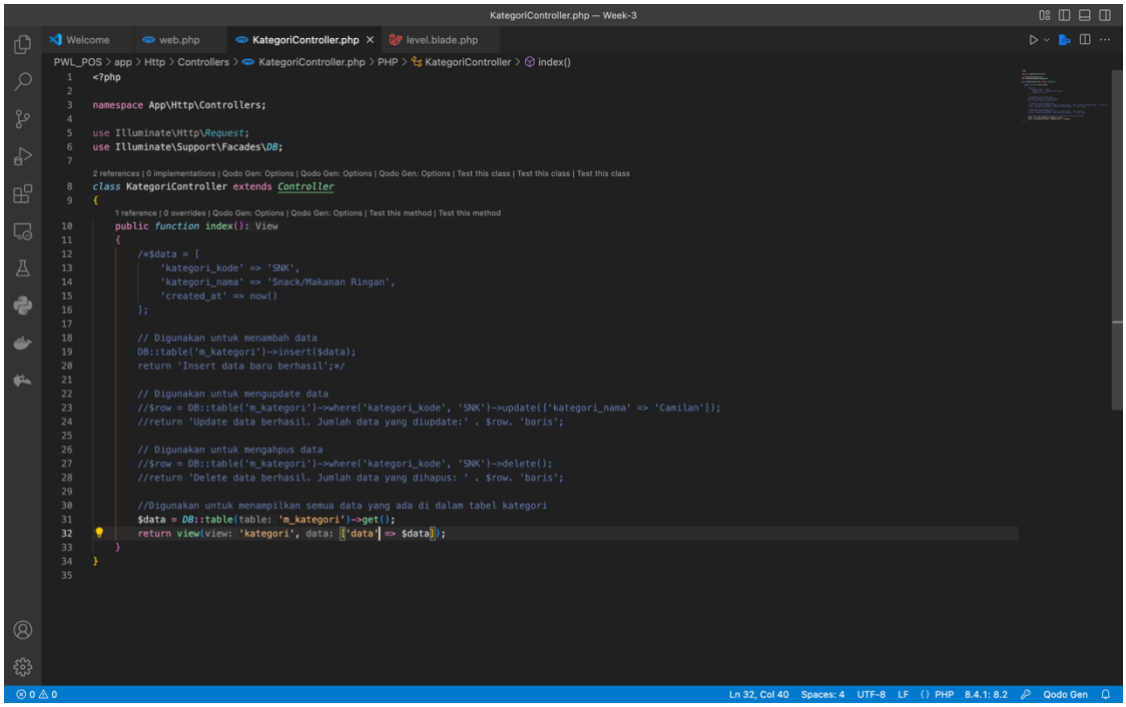


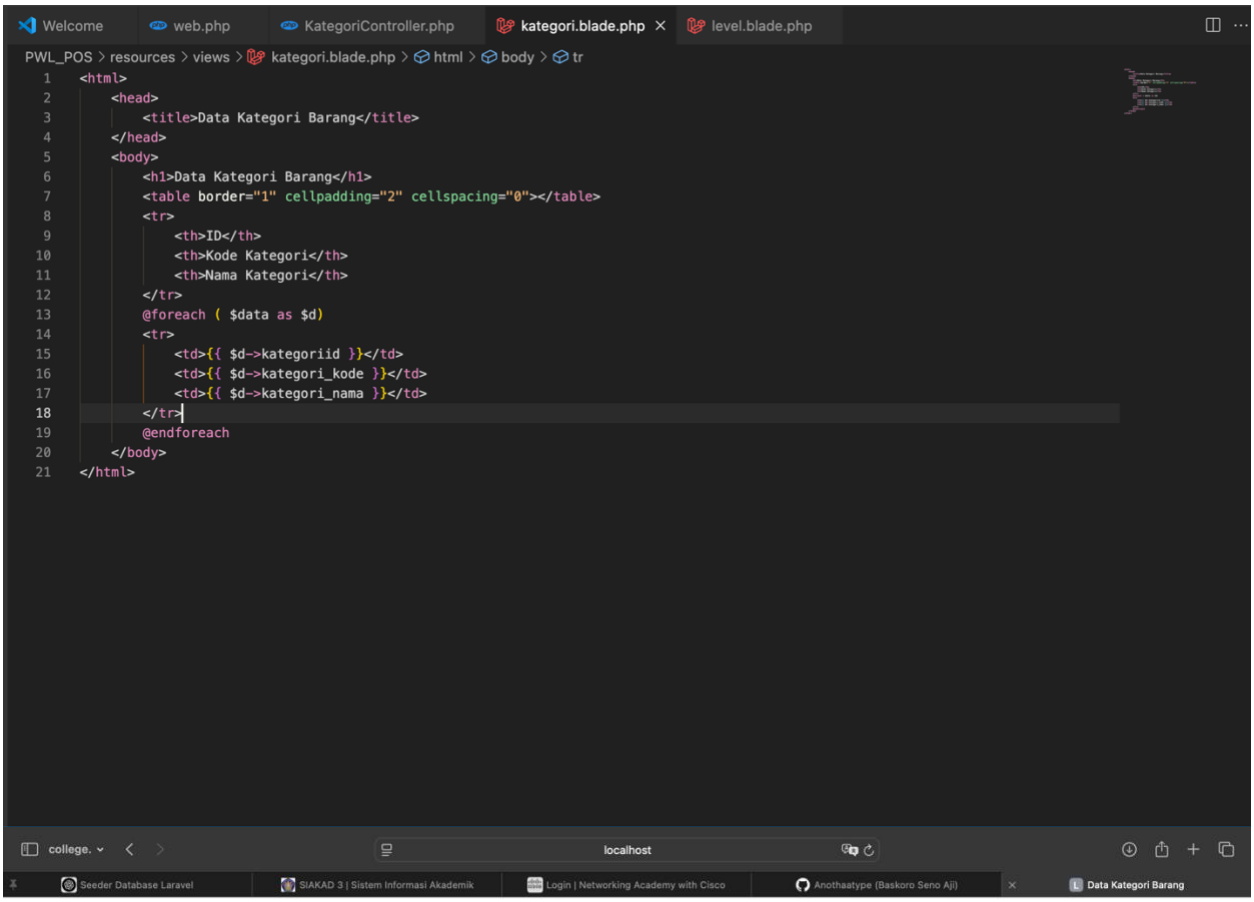


Delete data berhasil. Jumlah data yang dihapus: 2baris

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





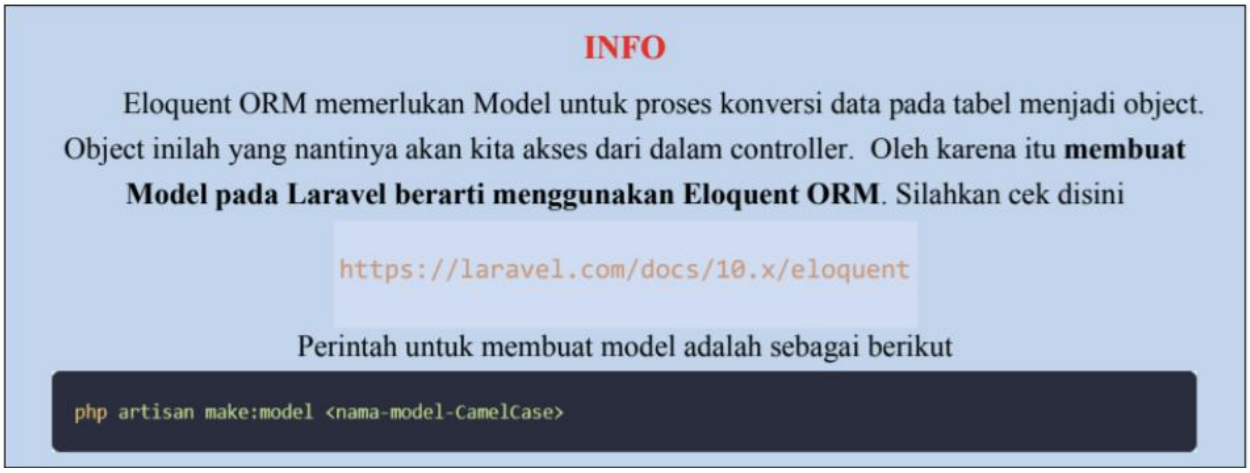
Data Kategori Barang

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 **Query Builder** 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.

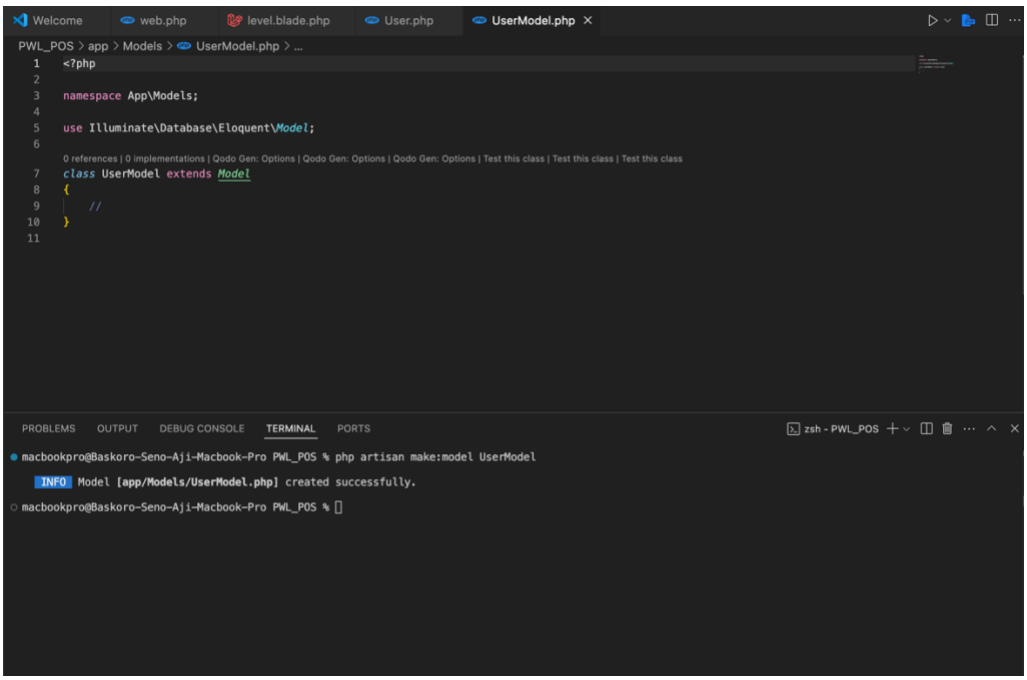
Eloquent ORM

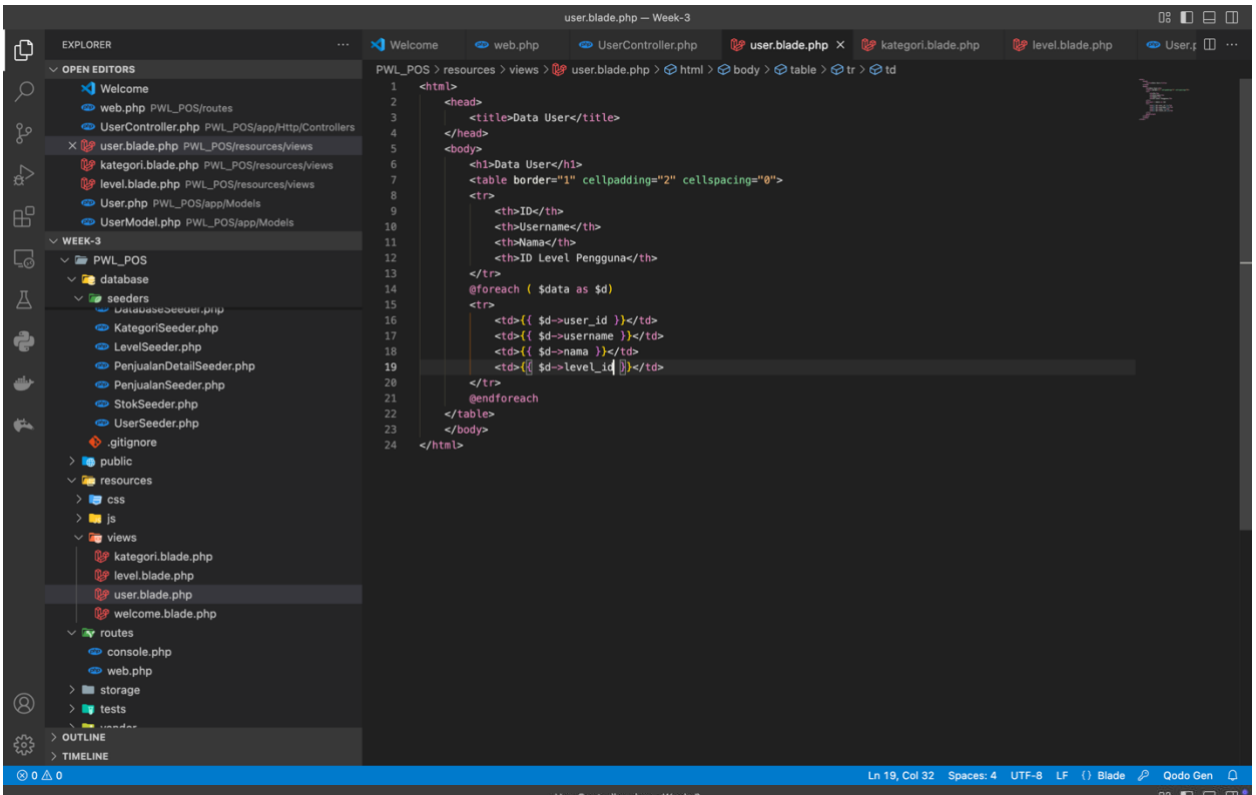
Eloquent ORM (Object-Relational Mapping) in Laravel is a feature that makes working with databases easier by allowing you to interact with database tables using PHP objects instead of writing raw SQL queries.

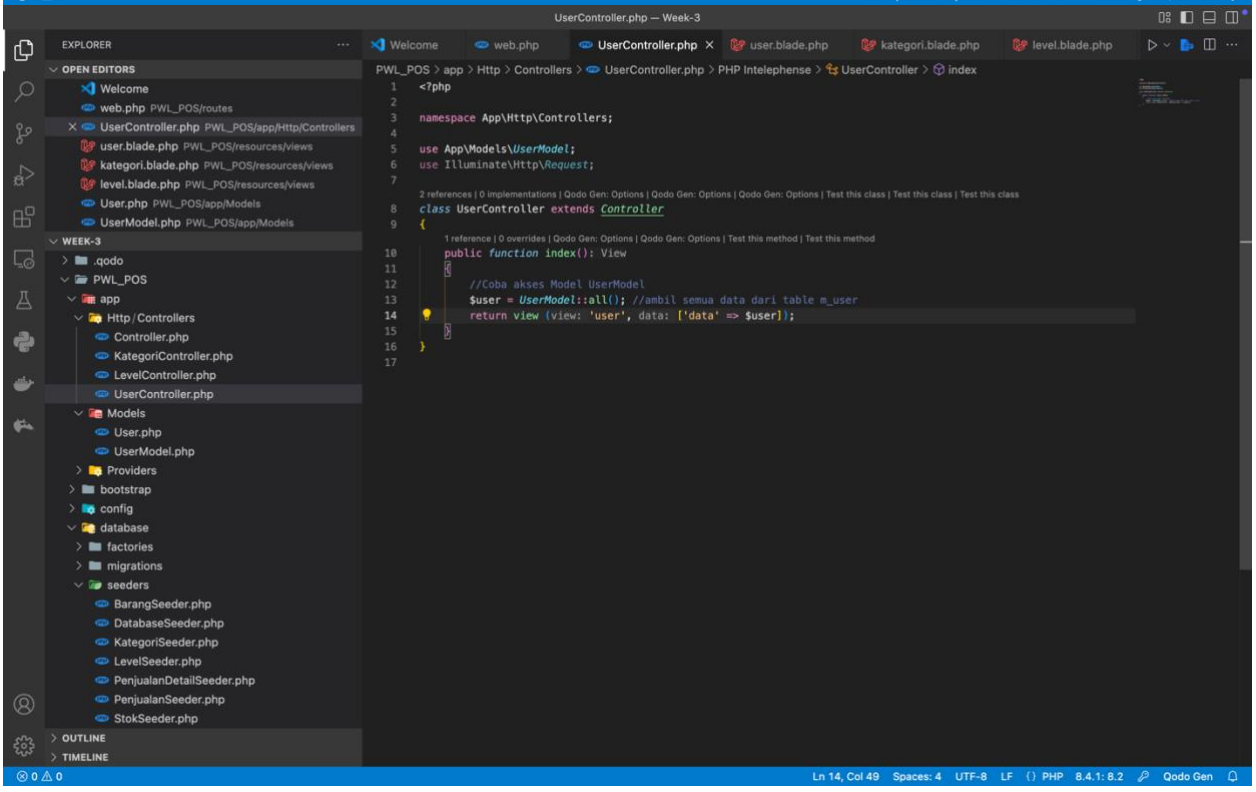


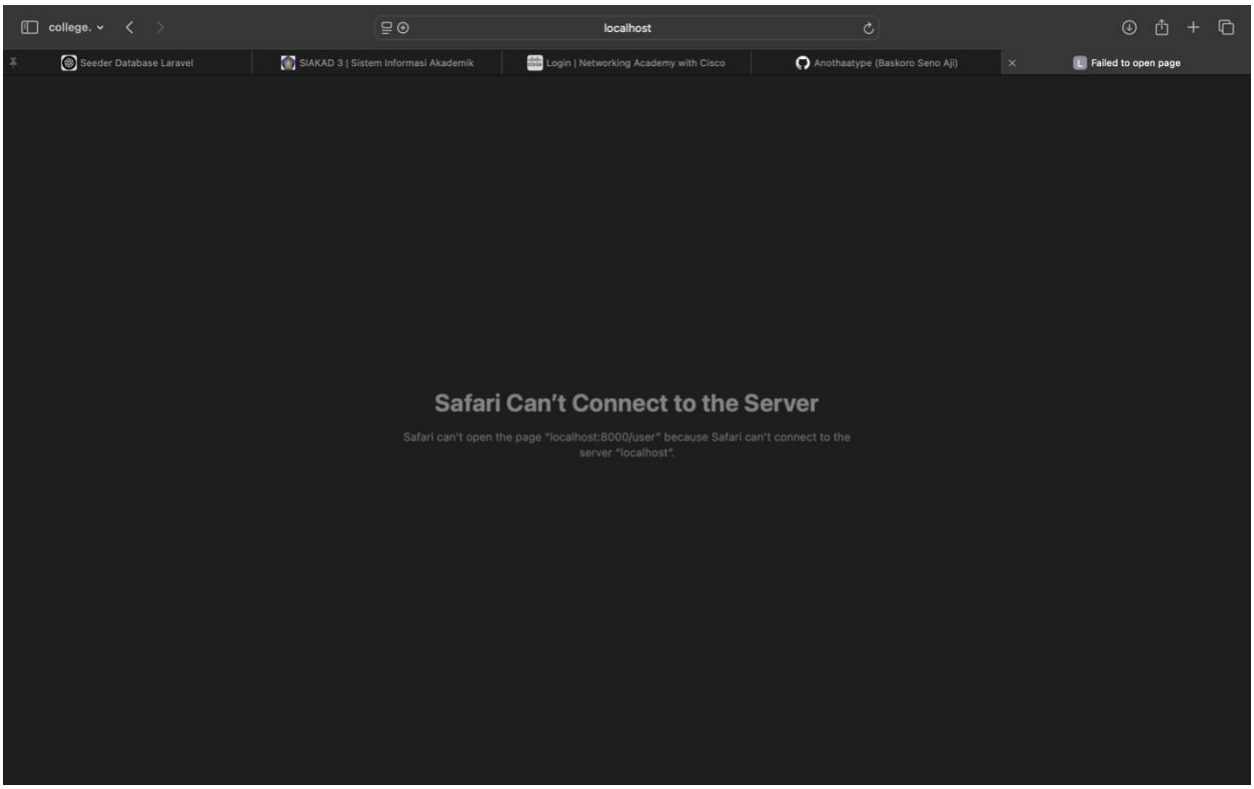
Praktikum 6 – Implementasi Eloquent Model

- First Model Controller




user.blade.php — Week-3
EXPLORER
OPEN EDITORS
Welcome
web.php PWL_POS/routes
UserController.php PWL_POS/app/Http/Controllers
user.blade.php PWL_POS/resources/views
kategori.blade.php PWL_POS/resources/views
level.blade.php PWL_POS/resources/views
User.php PWL_POS/app/Models
UserModel.php PWL_POS/app/Models
WEEK-3
PWL_POS
database
seeders
KategoriSeeder.php
LevelSeeder.php
PenjualanDetailSeeder.php
PenjualanSeeder.php
StokSeeder.php
UserSeeder.php
.gitignore
public
resources
css
js
views
kategori.blade.php
level.blade.php
user.blade.php
welcome.blade.php
routes
console.php
web.php
storage
tests
OUTLINE
TIMELINE
Welcome
web.php
UserController.php
user.blade.php
kategori.blade.php
level.blade.php
PWL_POS > resources > views > user.blade.php > html > body > table > tr > td
<html>
<head>
<title>Data User</title>
</head>
<body>
<h1>Data User</h1>
<table border="1" cellpadding="2" cellspacing="0">
<tr>
<th>ID</th>
<th>Username</th>
<th>Nama</th>
<th>ID Level Pengguna</th>
</tr>
@foreach ($data as $d)
<tr>
<td>{{ $d->user_id }}</td>
<td>{{ $d->username }}</td>
<td>{{ $d->nama }}</td>
<td>{{ $d->level_id }}</td>
</tr>
@endforeach
</table>
</body>
</html>
0 0
Ln 19, Col 32
Spaces: 4
UTF-8
LF
Blade
Qodo Gen


UserController.php — Week-3
EXPLORER
OPEN EDITORS
Welcome
web.php PWL_POS/routes
UserController.php PWL_POS/app/Http/Controllers
user.blade.php PWL_POS/resources/views
kategori.blade.php PWL_POS/resources/views
level.blade.php PWL_POS/resources/views
User.php PWL_POS/app/Models
UserModel.php PWL_POS/app/Models
WEEK-3
.qodo
PWL_POS
app
Http/Controllers
Controller.php
KategoriController.php
LevelController.php
UserController.php
Models
User.php
UserModel.php
Providers
bootstrap
config
database
factories
migrations
seeders
BarangSeeder.php
DatabaseSeeder.php
KategoriSeeder.php
LevelSeeder.php
PenjualanDetailSeeder.php
PenjualanSeeder.php
StokSeeder.php
OUTLINE
TIMELINE
Welcome
web.php
UserController.php
user.blade.php
kategori.blade.php
level.blade.php
PWL_POS > app > Http > Controllers > UserController.php > PHP Intelephense > UserController > index
<?php
namespace App\Http\Controllers;
use App\Models\UserModel;
use Illuminate\Http\Request;
2 references | 0 implementations | Qodo Gen: Options | Qodo Gen: Options | Qodo Gen: Options | Test this class | Test this class | Test this class
class UserController extends Controller
{
1 reference | 0 overrides | Qodo Gen: Options | Qodo Gen: Options | Test this method | Test this method
public function index(): View
{
//Coba akses Model UserModel
$user = UserModel::all(); //ambil semua data dari table m_user
return view('user', ['data' => $user]);
}
}
0 0
Ln 14, Col 49
Spaces: 4
UTF-8
LF
PHP
8.4.1: 8.2
Qodo Gen

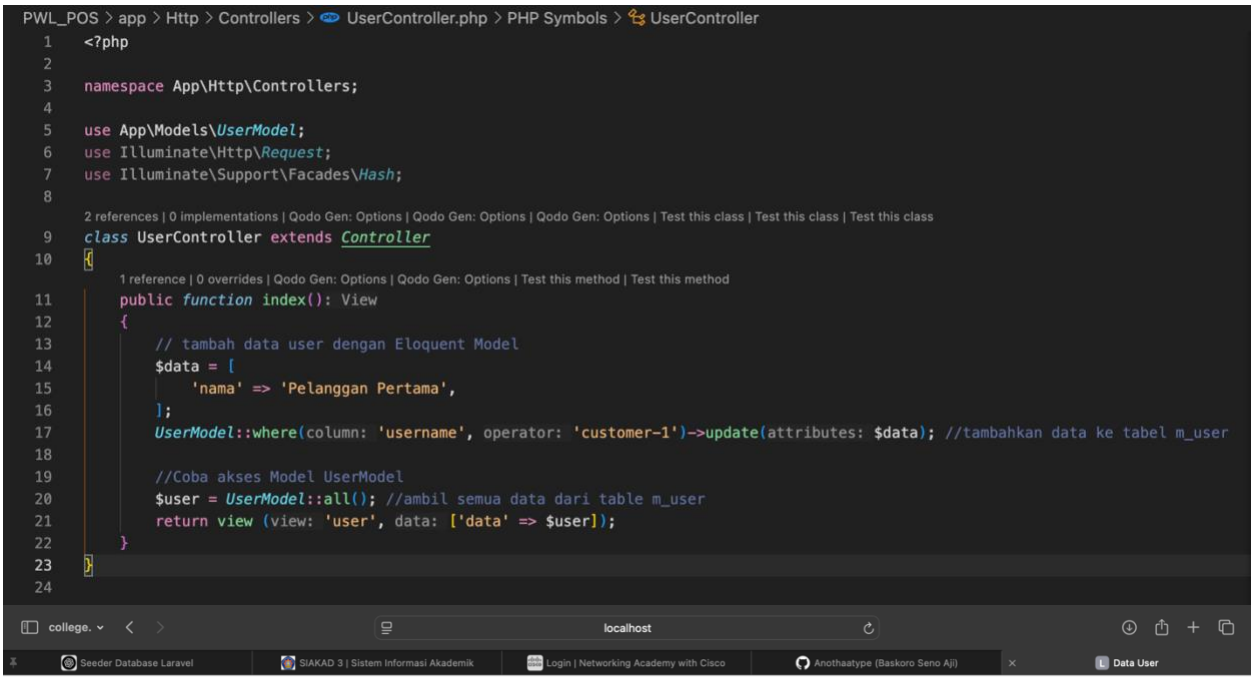


A. When we try to implement the Eloquent model we need to create the route, model (as an object) and also the view where we can use it as an output, but for the first model we can't implement it because the model is in the protected areas and we can't attain or connect into it

- Second Model Controller

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



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 `$tabel->unsignedBigInteger('level_id')`, sedangkan kolom `level_id` pada tabel `m_level` menggunakan `$tabel->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 the 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 create a Primary Key where it use 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 idcolumn 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.