

LAPORAN PRAKTIKUM 3

Mata Kuliah Pemrograman Website Lanjut

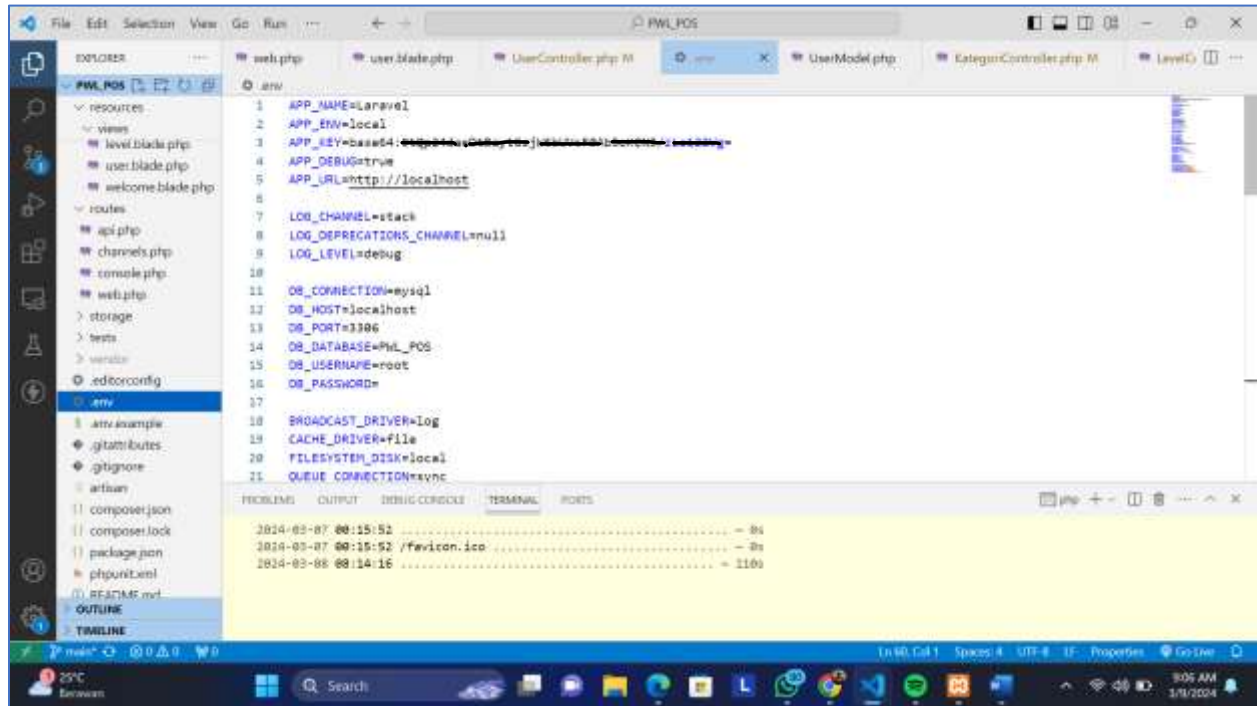


Oleh :

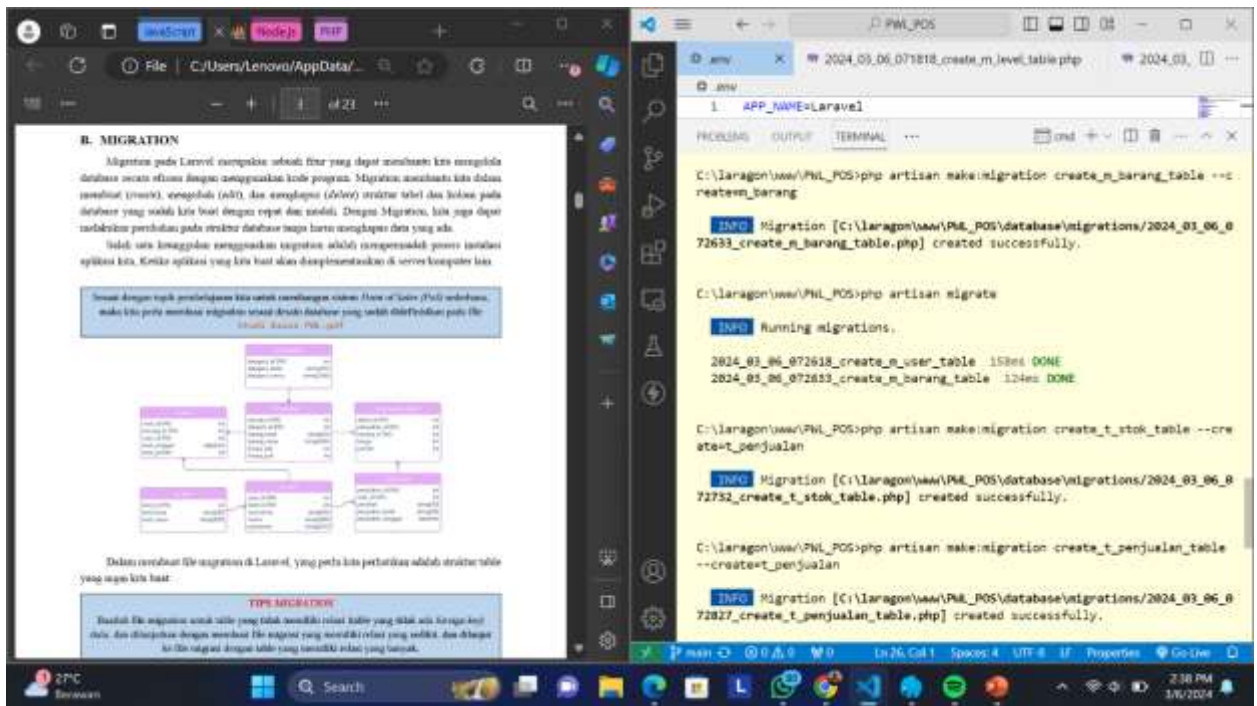
Avicenna Mumtaza
NIM 2241720112

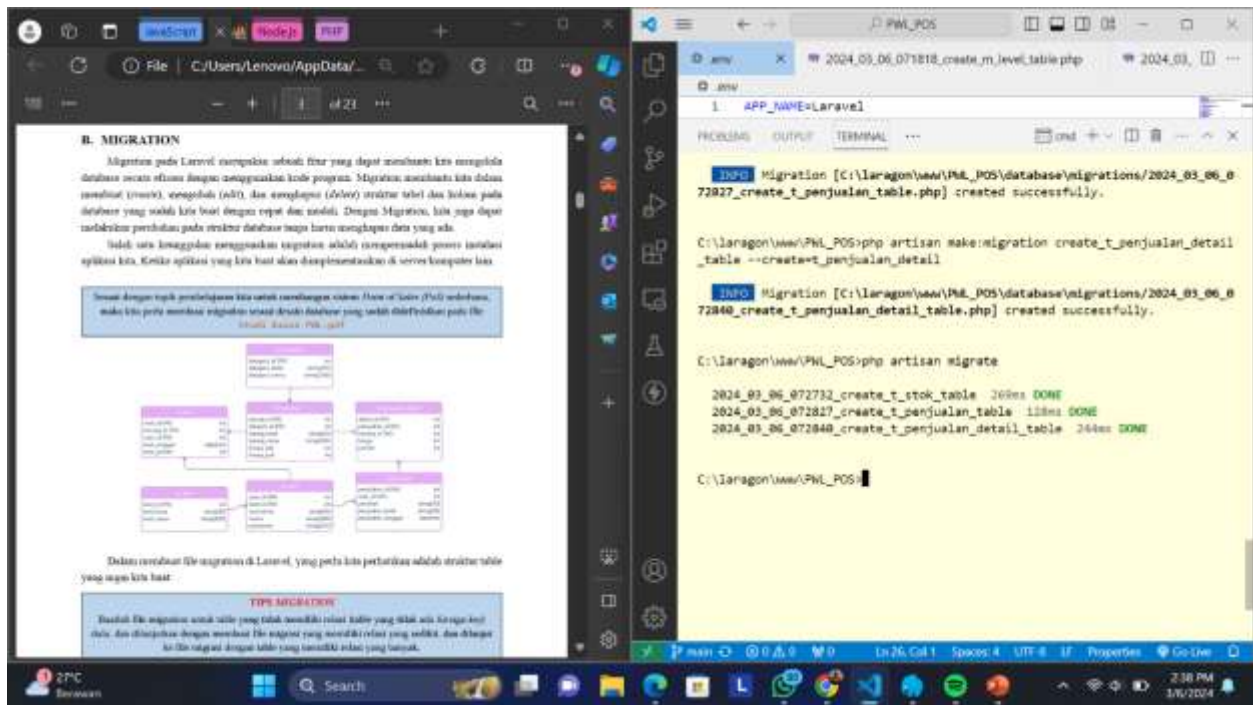
Teknik Informatika 2H

A. Pengaturan Database

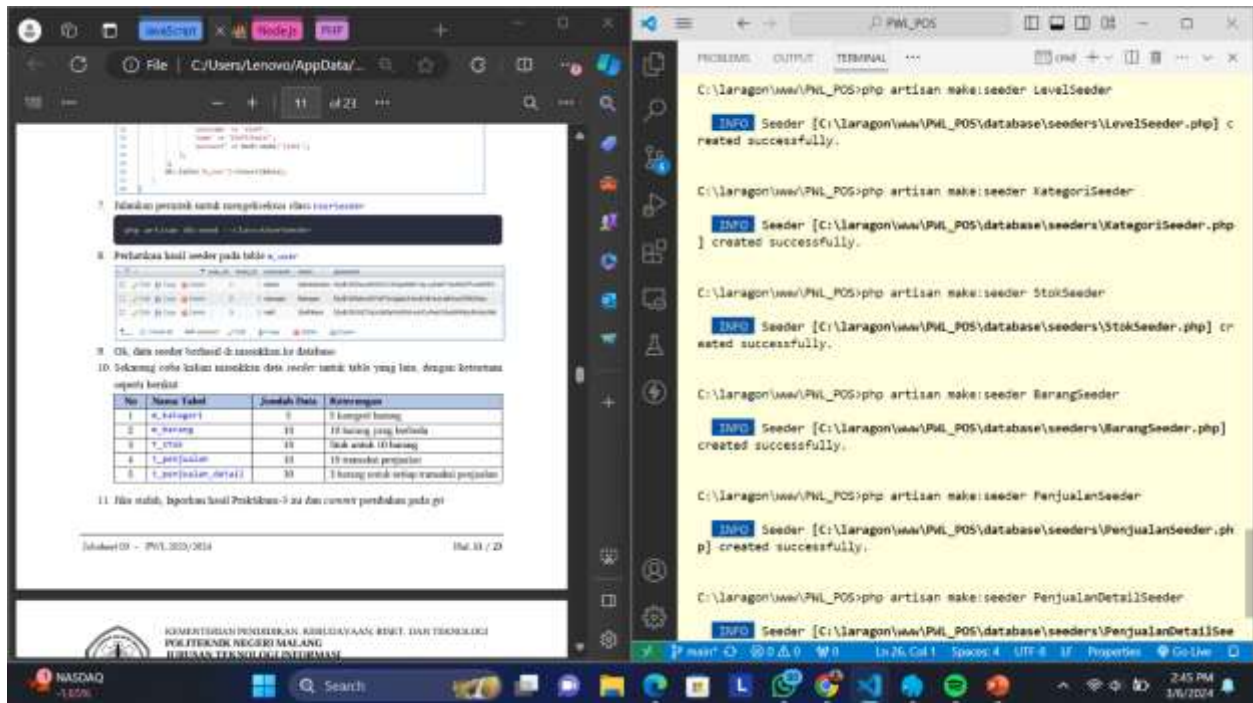


B. Migration

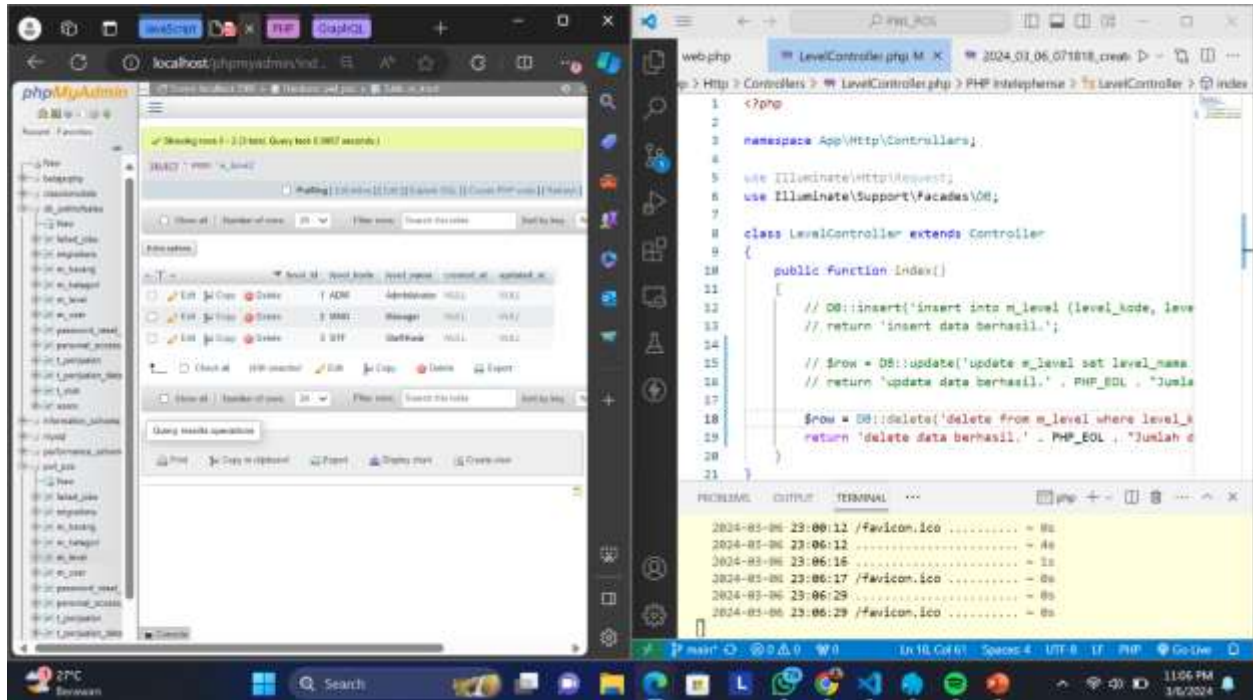




C. Seeder



D. DB Façade



The screenshot displays a web application interface on the left and a code editor on the right. The web application shows a table named 'm_level' with the following data:

level_kode	level_nama	level_jenis	created_at	updated_at
1	ADMIN	Administrator	2024-03-06 23:06:12	2024-03-06 23:06:12
2	MANAGER	Manager	2024-03-06 23:06:16	2024-03-06 23:06:16
3	STAFF	Staff	2024-03-06 23:06:17	2024-03-06 23:06:17

The code editor shows the 'LevelController.php' file with the following code:

```
<?php
namespace App\Http\Controllers;

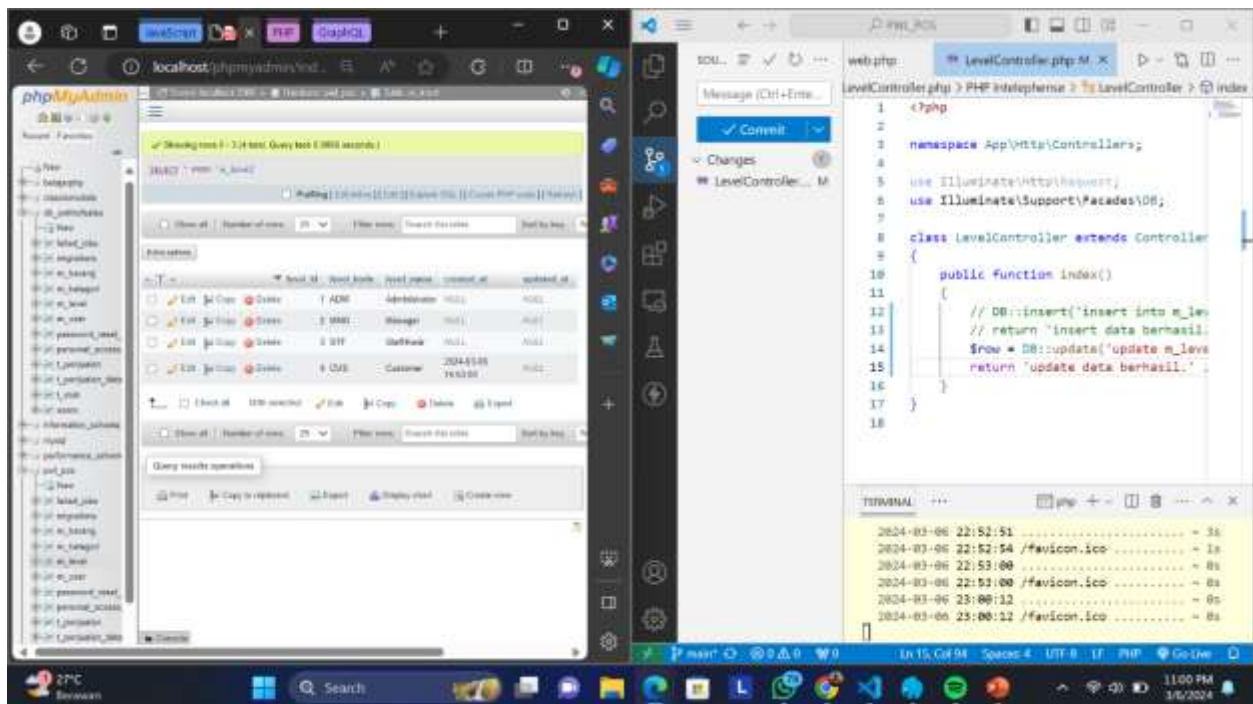
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController extends Controller
{
    public function index()
    {
        // DB::insert('insert into m_level (level_kode, level_nama, level_jenis) values (1, "ADMIN", "Administrator")');
        // return 'insert data berhasil.';

        // $row = DB::update('update m_level set level_nama = "Jumlah"');
        // return 'update data berhasil.' . PHP_EOL . "Jumlah";

        // $row = DB::delete('delete from m_level where level_kode = 1');
        // return 'delete data berhasil.' . PHP_EOL . "Jumlah";
    }
}
```

The terminal shows the application running successfully, displaying the output of the index method.



The screenshot displays the same web application interface on the left and the code editor on the right. The web application shows the 'm_level' table with the same data as before. The code editor shows the 'LevelController.php' file with the following code:

```
<?php
namespace App\Http\Controllers;

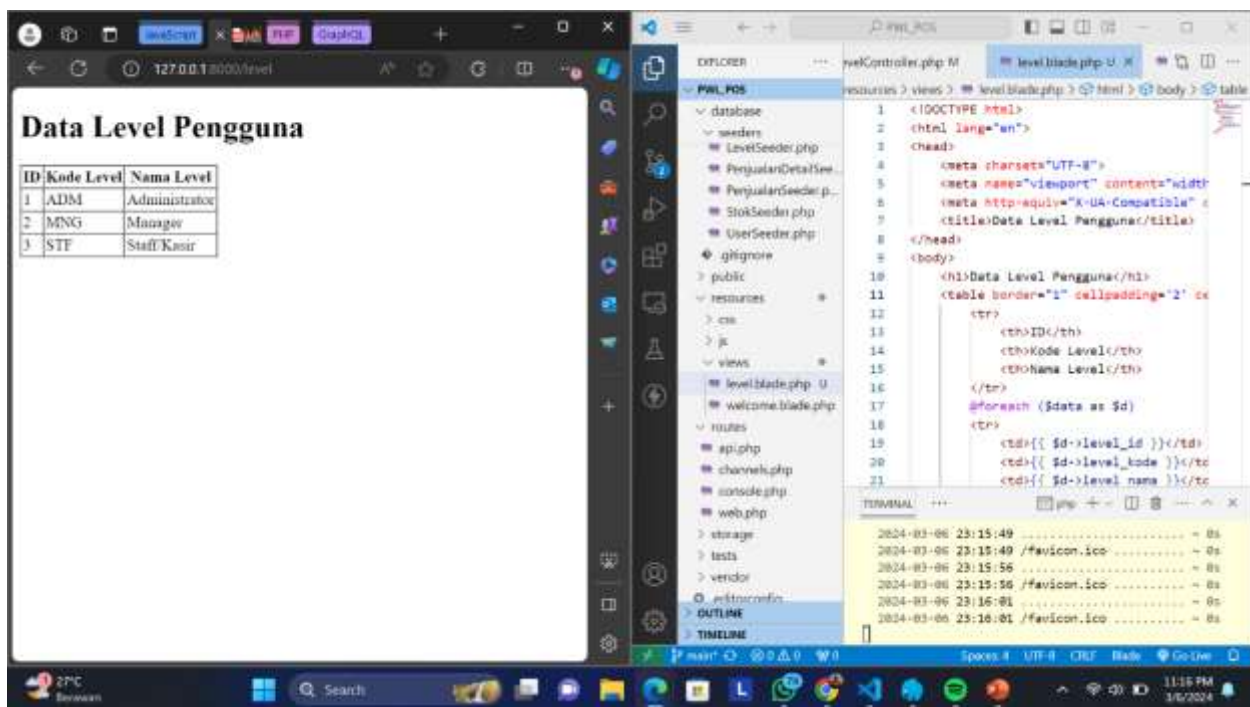
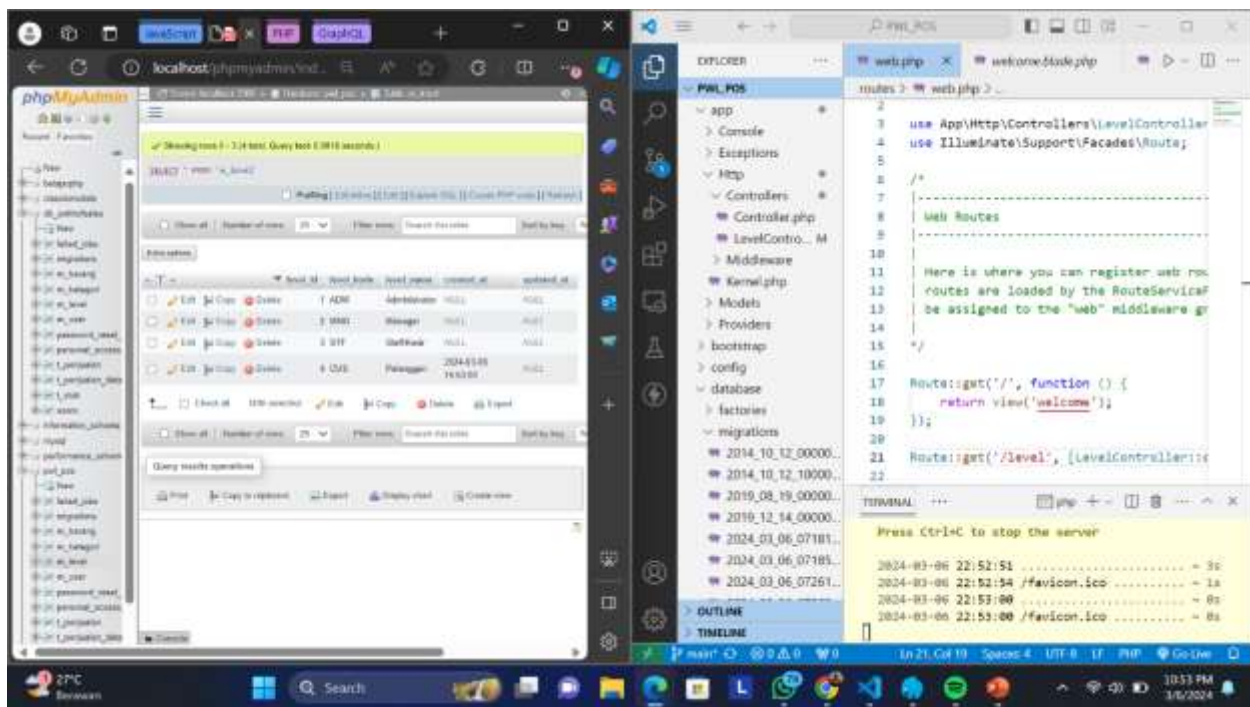
use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;

class LevelController extends Controller
{
    public function index()
    {
        // DB::insert('insert into m_level (level_kode, level_nama, level_jenis) values (1, "ADMIN", "Administrator")');
        // return 'insert data berhasil.';

        // $row = DB::update('update m_level set level_nama = "Jumlah"');
        // return 'update data berhasil.' . PHP_EOL . "Jumlah";

        // $row = DB::delete('delete from m_level where level_kode = 1');
        // return 'delete data berhasil.' . PHP_EOL . "Jumlah";
    }
}
```

The terminal shows the application running successfully, displaying the output of the index method.



E. Query Builder

The screenshot displays a web application interface and its underlying PHP code. The left pane shows a web browser at `127.0.0.1:8000/kategori` with the title "Data Kategori Barang". It contains a table with 5 rows of category data. The right pane shows the `KategoriController.php` file in a code editor, which uses Blade templating to render the table. The terminal at the bottom shows the server running on `localhost`.

Data Kategori Barang

ID	Kode Kategori	Nama Kategori
1	KAT001	Elektronik
2	KAT002	Pakaian
3	KAT003	Makanan
4	KAT004	Minuman
5	KAT005	Buku

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

phpMyAdmin

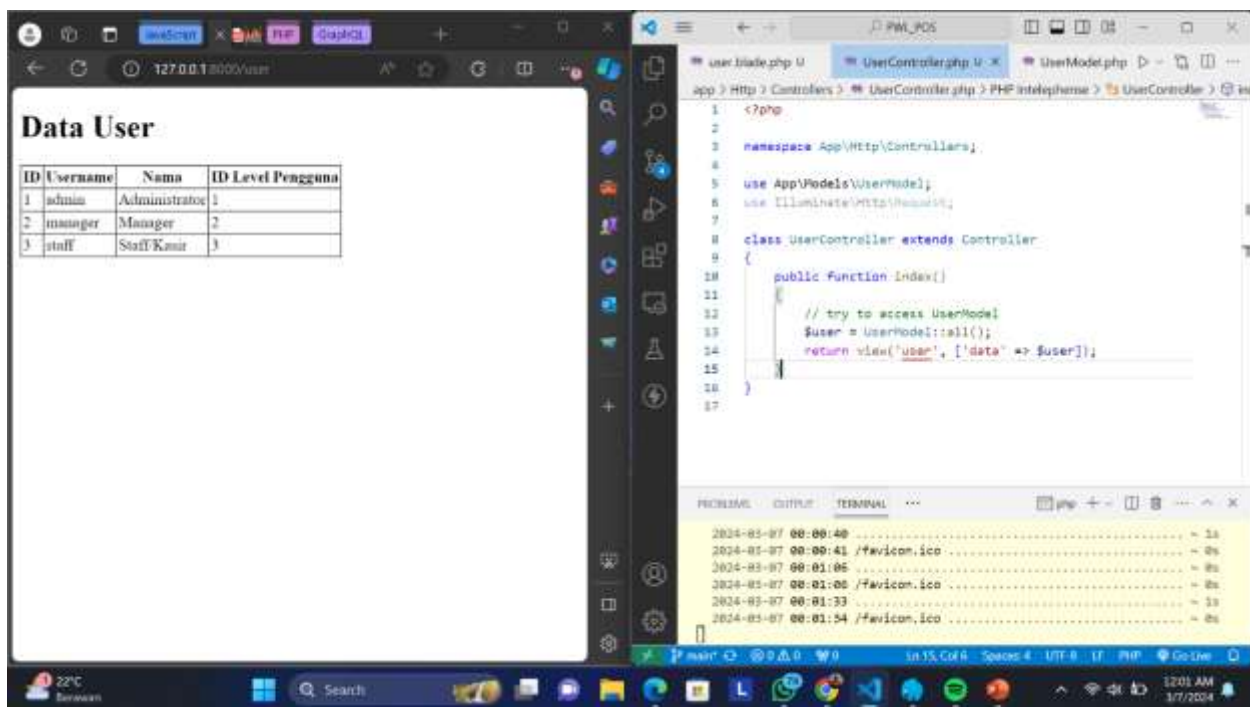
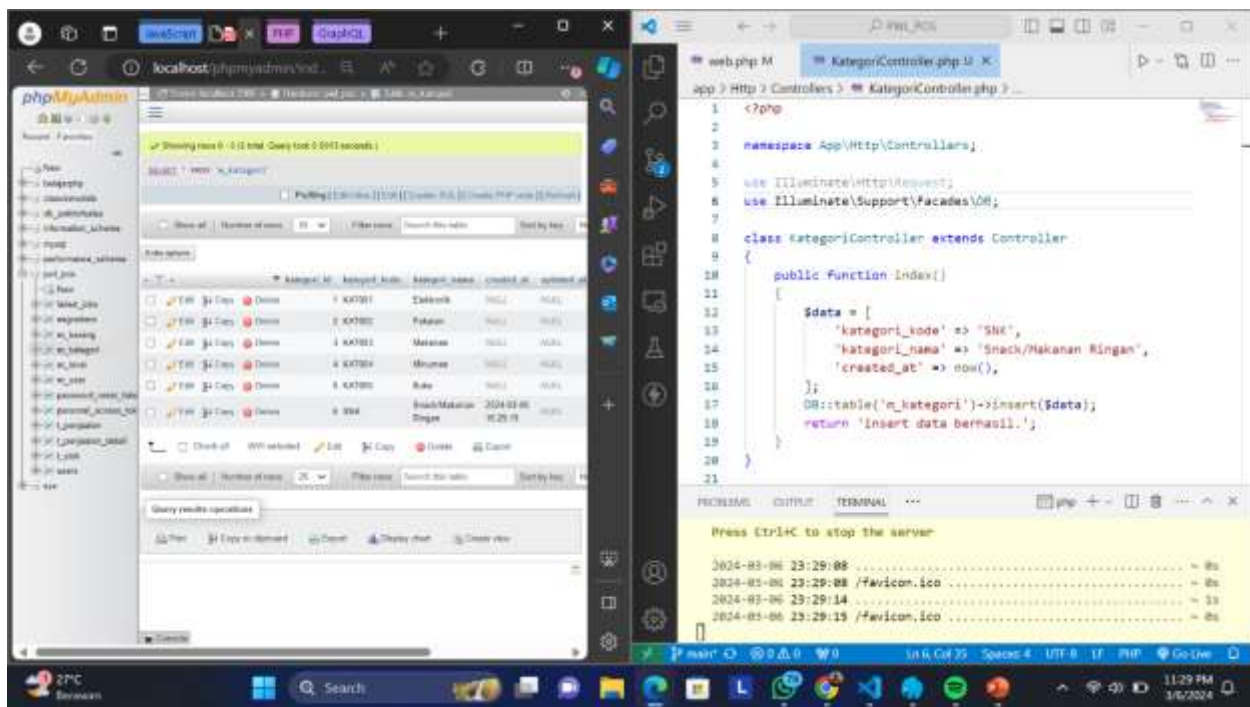
Showing rows 6 - 4 (4 total, Query took 0.0007 seconds)

Database: kategori

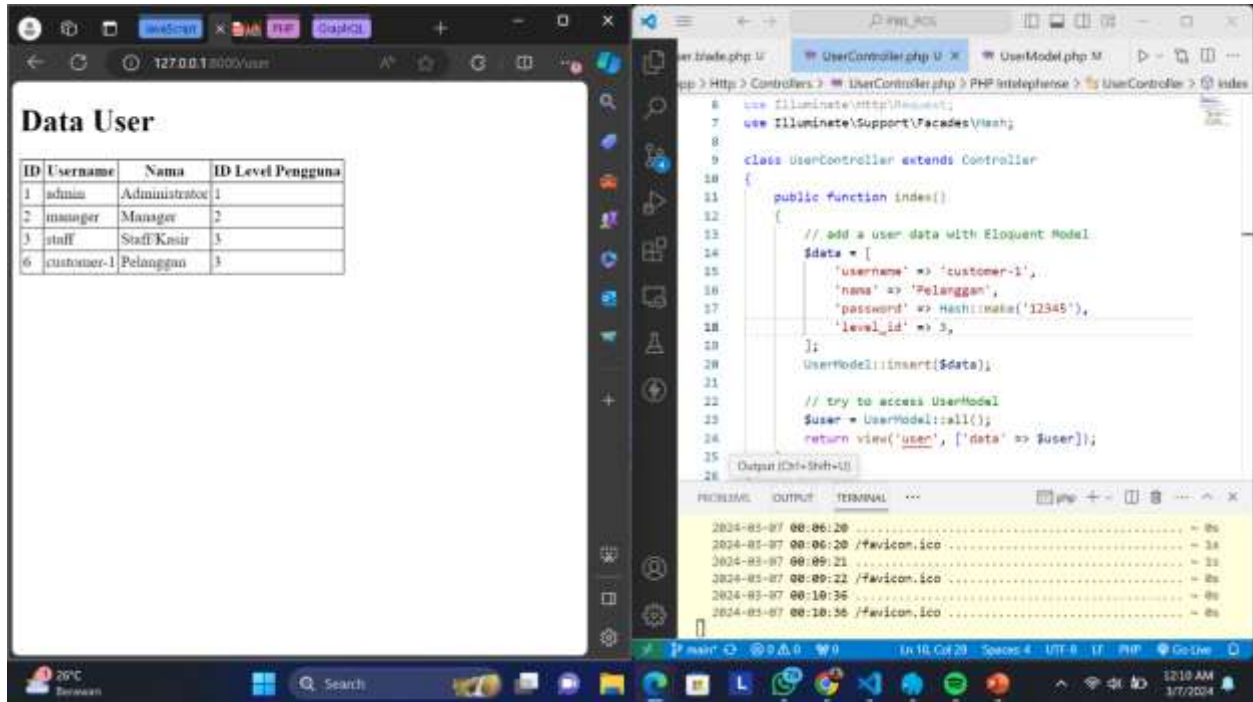
ID	Kode	Nama	created_at	updated_at
1	KAT001	Elektronik	2024-05-06 23:36:23	2024-05-06 23:36:23
2	KAT002	Pakaian	2024-05-06 23:37:15	2024-05-06 23:37:15
3	KAT003	Makanan	2024-05-06 23:37:15	2024-05-06 23:37:15
4	KAT004	Minuman	2024-05-06 23:44:36	2024-05-06 23:44:36
5	KAT005	Buku	2024-05-06 23:44:36	2024-05-06 23:44:36

KategoriController.php

```
1 <?php
2 namespace App\Http\Controllers;
3
4 use Illuminate\Http\Request;
5 use Illuminate\Support\Facades\DB;
6
7 class KategoriController extends Controller
8 {
9     public function index()
10     {
11         // $data = [
12         //     'kategori_kode' => 'SNK',
13         //     'kategori_nama' => 'Snack/Makanan Ringan',
14         //     'created_at' => now(),
15         // ];
16         // DB::table('m_kategori')->insert($data);
17         // return 'insert data berhasil.';
18
19         // $row = DB::table('m_kategori')->where('kategori_kode', 'SNK');
20         // return 'update data berhasil', PHP_EOL, "Jumlah";
```

F. Eloquent ORM



The screenshot shows a web browser on the left and a code editor on the right. The browser displays a table titled "Data User" with the following data:

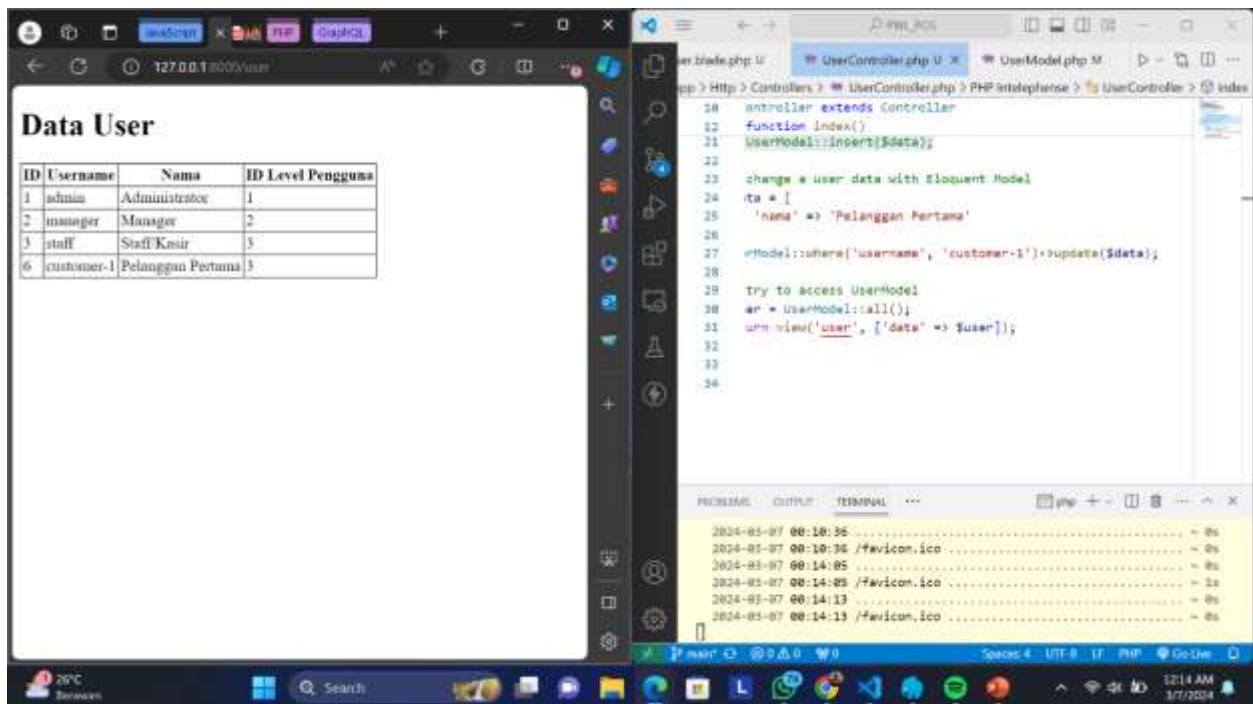
ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staf	Staff Ksatri	3
6	customer-1	Pelanggan	3

The code editor shows the `UserController.php` file with the following code:

```
1 use Illuminate\Http\Request;
2 use Illuminate\Support\Facades\Hash;
3
4 class UserController extends Controller
5 {
6     public function index()
7     {
8         // add a user data with Eloquent Model
9         $data = [
10             'username' => 'customer-1',
11             'nama' => 'Pelanggan',
12             'password' => Hash::make('12345'),
13             'level_id' => 3,
14         ];
15         UserModel::insert($data);
16
17         // try to access UserModel
18         $user = UserModel::all();
19         return view('user', ['data' => $user]);
20     }
21 }
```

The terminal output shows the following logs:

```
2024-05-07 00:06:20 ..... -> 0s
2024-05-07 00:06:20 /favicon.ico ..... -> 3s
2024-05-07 00:09:21 ..... -> 2s
2024-05-07 00:09:22 /favicon.ico ..... -> 0s
2024-05-07 00:10:35 ..... -> 0s
2024-05-07 00:10:36 /favicon.ico ..... -> 0s
```



The screenshot shows a web browser on the left and a code editor on the right. The browser displays a table titled "Data User" with the following data:

ID	Username	Nama	ID Level Pengguna
1	admin	Administrator	1
2	manager	Manager	2
3	staf	Staff Ksatri	3
6	customer-1	Pelanggan Pertama	3

The code editor shows the `UserController.php` file with the following code:

```
1 controller extends Controller
2
3 function index()
4 {
5     UserModel::insert($data);
6
7     // change a user data with Eloquent Model
8     $data = [
9         'nama' => 'Pelanggan Pertama'
10     ];
11     UserModel::where('username', 'customer-1')->update($data);
12
13     // try to access UserModel
14     $user = UserModel::all();
15     return view('user', ['data' => $user]);
16 }
```

The terminal output shows the following logs:

```
2024-05-07 00:10:35 ..... -> 0s
2024-05-07 00:10:36 /favicon.ico ..... -> 0s
2024-05-07 00:14:05 ..... -> 0s
2024-05-07 00:14:05 /favicon.ico ..... -> 3s
2024-05-07 00:14:13 ..... -> 0s
2024-05-07 00:14:13 /favicon.ico ..... -> 0s
```

Pertanyaan :

1. Pada Praktikum 1 - Tahap 5, apakah fungsi dari APP_KEY pada file setting .env Laravel? Sesuai namanya, APP_KEY dapat berperan sebagai kunci dari aplikasi di local device kita. APP_KEY melakukan enkripsi untuk melindungi data yang disimpan atau dikirimkan dalam aplikasi. Laravel menggunakan APP_KEY untuk mengenkripsi dan mendekripsi data yang disimpan dalam cookie atau sesi.

2. Pada Praktikum 1, bagaimana kita men-generate nilai untuk APP_KEY? Menjalankan perintah `php artisan key:generate` di terminal direktori project.

3. Pada Praktikum 2.1 - Tahap 1, secara default Laravel memiliki berapa file migrasi? dan untuk apa saja file migrasi tersebut? Laravel memiliki 4 file migrasi default dalam direktori `databases/migrations`. Berikut adalah penjelasan untuk keempat file migrasi default tersebut:

1. `2014_10_12_000000_create_users_table.php`:

File ini merupakan migrasi pertama yang dibuat secara otomatis ketika Anda pertama kali membuat proyek Laravel. File ini menciptakan tabel `users` yang sering digunakan untuk menyimpan informasi pengguna seperti nama, email, dan password. Tabel ini umumnya digunakan untuk otentikasi dan manajemen pengguna.

2. `2014_10_12_100000_create_password_resets_table.php`:

Migrasi ini menciptakan tabel `password_resets` yang digunakan untuk menyimpan token reset password. Ketika pengguna lupa password, aplikasi dapat menghasilkan token unik dan menyimpannya di tabel ini bersama dengan informasi waktu dan email pengguna. Tabel ini membantu dalam proses reset password.

3. `2019_08_19_000000_create_failed_jobs_table.php`:

File ini menciptakan tabel `failed_jobs` yang digunakan untuk melacak pekerjaan antrian (queue jobs) yang gagal dieksekusi. Jika pekerjaan antrian tidak berhasil, informasi tentang kegagalan tersebut disimpan di tabel ini, membantu Anda untuk menganalisis dan mengatasi masalah.

4. 2019_08_19_000000_create_password_resets_table.php:

Sama seperti file nomor 2, ini adalah duplikat. Tapi lebih terkait kepada pembuatan tabel `personal_access_tokens` yang mungkin terkait dengan otentikasi atau kontrol akses.

Ketika menjalankan perintah `php artisan migrate`, Laravel akan membaca file-file migrasi ini dan menjalankan perubahan struktur basis data yang didefinisikan di dalamnya. Misalnya, ketika Anda menjalankan migrasi pertama, tabel `users` akan dibuat di basis data Anda.

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

Untuk menyimpan data kapan data di baris dalam tabel dibuat (`created_at`) dan kapan data tersebut diperbarui (`updated_at`).

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

Untuk mengisi data pada kolom dengan metode auto increment sebagai primary key, tipe data yang dihasilkan adalah integer, lebih tepatnya dari `unsignedBigInteger()`.

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

Sebenarnya tidak ada perbedaan hasil dari 2 penggunaan tersebut, kalau kita bicara data. Tapi kalau membahas struktur tabel, perbedaannya terletak pada nama kolom atau column header. Apabila kita tidak memberikan parameter maka kolom tersebut diset nama default, yakni `id`. Jadi, parameter di situ berfungsi sebagai nilai yang akan dijadikan nama kolom pada tabel terkait.

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

Fungsi `->unique()` digunakan untuk menetapkan konstrain unik pada suatu kolom di tabel basis data. Konstrain unik membantu memastikan integritas data dan mencegah duplikasi yang tidak diinginkan di dalam tabel. Ini berarti nilai di kolom tersebut harus unik di seluruh baris dalam tabel. Jika mencoba memasukkan atau memperbarui data sehingga nilai di kolom tersebut tidak unik, maka operasi tersebut akan gagal, dan database akan melemparkan kesalahan.

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')` ?

Perbedaan terletak pada isi dari kedua method tersebut. Method `unsignedBigInteger()` hanya return nilai integer dengan length yang besar. Sama seperti pada method `id()`, hanya saja di method ini nilai yang dimasukkan tidak sembarangan, melalui auto increment.

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

Class Hash memiliki beberapa method seperti `check()`, `needRehash()`, dan sebagainya. Method `make()` adalah salah satunya, dari kode program di atas method tersebut bertujuan untuk mengubah string 1234 menjadi string lain yang sifatnya random dan unik. Manfaat lain ketika ada orang yang melakukan upaya pencurian, data tersebut tidak cukup bisa dipahami dengan hanya membacanya. Sehingga keamanan value sebenarnya dari data tersebut masih terjaga.

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

Tanda tanya tersebut berguna untuk fleksibilitas penggunaan DB façade, yang mana berfungsi sebagai wadah dari variable yang menampung data di database terkait. Dalam konteks praktikum kali ini, data yang dimasukkan adalah value dari kolom yang terlibat operasi insert, update, dan delete.

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

Kedua variable tersebut merupakan variable dari parent Model dan access modifier dari kedua variable tersebut diset `protected` di child `UserModel` supaya bisa meng-override atau menimpa nilai default dari variable asalnya (parent class Model).

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

Karena yang saya kerjakan di praktikum ini masih belum terlalu kompleks, saya akan menjawab lebih mudah DB Façade karena tidak perlu membuat class Model untuk set table dan primary key. Namun untuk penggunaan DB Façade yang lebih maksimal, diharuskan memiliki kemampuan memahami Queri sebelumnya.