LAPORAN PRAKTIKUM
PERANCANGAN DAN PEMROGRAMAN WEB

MODUL 12

LARAVEL II

Oleh:

Dhiemas Tulus Ikhsan - 2311104046

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
DIREKTORAT KAMPUS PURWOKERTO
UNIVERSITAS TELKOM
2025

BAB I

PENDAHULUAN

1.1 Dasar Teori

Migration di Laravel adalah fitur pengelolaan struktur database yang memungkinkan pembuatan, modifikasi, hingga penghapusan tabel dan kolom melalui kode program tanpa harus mengetik perintah SQL manual. Berfungsi sebagai sistem kontrol versi (version control) untuk database, setiap perubahan tersimpan dalam file migrasi yang bisa dijalankan ulang di berbagai lingkungan pengembangan, sehingga mempermudah kolaborasi tim serta proses deployment. Selain itu, Laravel menawarkan tiga metode utama dalam mengolah data: DB Facade (Raw SQL) untuk menjalankan perintah SQL mentah pada query yang kompleks, Query Builder yang menyediakan antarmuka berbasis metode yang lebih aman dari SQL Injection serta mudah dibaca, dan Eloquent ORM yang memetakan tabel database menjadi model serta baris data menjadi objek, sehingga pengelolaan data dapat dilakukan dengan pendekatan berorientasi objek yang lebih rapi dan efisien.

1.2 Tujuan

Tujuan dari praktikum ini adalah untuk memahami konsep dan fungsi Migration pada framework Laravel dalam pengelolaan database. Selain itu, praktikum ini bertujuan agar praktikan mampu menerapkan migration untuk membuat, mengubah, dan menghapus struktur tabel database secara terstruktur menggunakan perintah artisan. Praktikan juga diharapkan dapat memahami konsep pengelolaan database yang terintegrasi dengan Laravel melalui pendekatan DB Facade, Query Builder, dan Eloquent ORM.

BAB II

HASIL PRAKTIKUM

2.1 GUIDED (Praktikum Terbimbing)

Konfigurasi Database Laravel

Tahap awal sebelum mengeksekusi migration adalah menyelaraskan konfigurasi database pada Laravel. Pengaturan ini dilakukan melalui file .env untuk kebutuhan lingkungan lokal dan file config/database.php untuk skala global. Dalam file .env, kamu perlu menyesuaikan parameter seperti nama database, username, dan password agar sinkron dengan database yang telah disiapkan di phpMyAdmin. Langkah konfigurasi ini sangat krusial guna memastikan Laravel berhasil terhubung dengan sistem manajemen database MySQL yang digunakan selama praktikum.

Contoh konfigurasi pada file .env:

```
23  DB_CONNECTION=mysql
24  DB_HOST=127.0.0.1
25  DB_PORT=3306
26  DB_DATABASE=coba
27  DB_USERNAME=root
28  DB_PASSWORD=
```

Migration Bawaan Laravel

Secara standar, Laravel telah menyertakan sejumlah file migrasi awal di dalam direktori database/migrations untuk membangun tabel-tabel esensial seperti users, password_resets, failed_jobs, dan personal_access_tokens. Penamaan setiap file migrasi tersebut diawali dengan timestamp guna mengatur urutan eksekusi berdasarkan waktu pembuatannya. Untuk menerapkan skema tersebut ke dalam database, kamu cukup mengeksekusi perintah php artisan migrate melalui terminal. Begitu proses selesai, tabel-tabel bawaan tersebut akan langsung tercipta secara otomatis di dalam database yang telah dikonfigurasi.

Pembuatan File Migration

Dalam praktikum ini, sebuah migrasi baru untuk tabel mahasiswas dibuat menggunakan perintah artisan guna memastikan struktur filenya memenuhi standar Laravel. Perintah yang dieksekusi adalah php artisan make:migration

create_mahasiswas_table --create=mahasiswas. Perintah ini secara otomatis menghasilkan file migrasi baru di dalam direktori database/migrations dengan nama yang diawali oleh timestamp sebagai penanda waktu pembuatannya.

Implementasi Struktur Tabel Migration

Begitu file migrasi selesai dibuat, langkah berikutnya adalah mendefinisikan struktur tabel di dalam method up() dan down(). Fungsi up() bertanggung jawab untuk membangun tabel beserta kolom-kolomnya, sementara fungsi down() berperan sebagai kebalikannya, yaitu menghapus tabel ketika perintah rollback dijalankan.

Contoh kode migration tabel mahasiswas:

```
<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

/**

* Run the migrations.

*/

public function up()

{

Schema::create('mahasiswas', function (Blueprint $table) {

$table->id();

$table->char('nim', 8);

$table->string('nama');

$table->string('tempat_lahir');

$table->date('tanggal_lahir');

$table->string('fakultas');

$table->string('jurusan');

$table->decimal('ipk', 3, 2);
```

```
$table->timestamps();

});

}

public function down()

{

Schema::dropIfExists('mahasiswas');

}

};
```

Setelah skema tabel selesai ditentukan, jalankan kembali migrasi dengan perintah php artisan migrate. Eksekusi perintah ini akan merealisasikan rancangan kode tersebut menjadi tabel mahasiswas yang nyata di dalam database MySQL.

Rollback Migration

Rollback migration digunakan untuk mengembalikan struktur database ke kondisi sebelumnya jika terjadi kesalahan. Untuk melakukan rollback satu langkah terakhir, digunakan perintah: php artisan migrate:rollback --step=1. Perintah ini akan menghapus tabel yang terakhir kali dibuat melalui migration.

Modifikasi Struktur Migration

Migration yang sudah dibuat dapat dimodifikasi sesuai kebutuhan. Pada praktikum ini dilakukan perubahan dengan menambahkan constraint dan nilai default pada kolom tertentu.

Contoh perubahan pada file migration:

```
$table->char('nim', 8)->unique();

$table->decimal('ipk', 3, 2)->default(1.00);
```

Setelah melakukan perubahan, migration dijalankan kembali agar perubahan dapat diterapkan ke database.

Alter Table Migration

Selain membuat tabel baru, migration juga dapat digunakan untuk memodifikasi struktur tabel yang sudah ada (ALTER TABLE). Untuk melakukan alter table, Laravel membutuhkan library tambahan yaitu Doctrine DBAL. Instalasi Doctrine DBAL:

composer require doctrine/dbal. Setelah itu dibuat file migration baru: php artisan make:migration alter_mahasiswas_table --table=mahasiswas

Implementasi Alter Table Migration

Pada file migration alter, dilakukan beberapa perubahan struktur tabel, seperti mengubah nama kolom, menambah kolom baru, dan menghapus kolom.

Contoh kode alter table:

```
public function up()

{

Schema::table('mahasiswas', function (Blueprint $table) {

$table->renameColumn('nama', 'nama_lengkap');

$table->text('alamat')->after('tanggal_lahir');

$table->dropColumn('ipk');

});

}

public function down()

{

Schema::table('mahasiswas', function (Blueprint $table) {

$table->renameColumn('nama_lengkap', 'nama');

$table->dropColumn('alamat');

$table->decimal('ipk', 3, 2)->default(1.00);

});

}
```

Kode pada method up() dan down() harus saling berpasangan agar proses rollback dapat berjalan dengan baik. Setelah migration dijalankan, struktur tabel mahasiswas akan berubah sesuai dengan modifikasi yang dilakukan.

Pembuatan Route

Sebelum melakukan pengolahan data, dibuat route untuk mengakses fungsi insert, select, update, dan delete data mahasiswa.

routes/web.php

```
<?php

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\MahasiswaController;

Route::get('/', function () {

return view('welcome');

});

Route::get('/insert-data', [MahasiswaController::class, 'insertData']);

Route::get('/select-data', [MahasiswaController::class, 'selectData']);

Route::get('/update-data', [MahasiswaController::class, 'updateData']);

Route::get('/delete-data', [MahasiswaController::class, 'deleteData']);
```

Pembuatan Controller

Controller digunakan untuk menampung logika pengolahan data database.

Jalankan perintah: php artisan make:controller MahasiswaController

Insert Data Menggunakan DB Facade (Raw SQL)

Metode DB Facade digunakan untuk menjalankan query SQL secara langsung.

```
<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\DB;

class MahasiswaController extends Controller

{

public function insertData()

{

DB::insert(

"INSERT INTO mahasiswas

(nim, nama_lengkap, tempat_lahir, tanggal_lahir, alamat, fakultas, jurusan)

VALUES (?, ?, ?, ?, ?, ?, ?)",

[
```

```
'23111046',

'Dhiemas Ikhsan',

'SKJ',

'2005-05-20',

'Jl. Mlatiharjo',

'Informatika',

'RPL'

]

);

return "Data berhasil ditambahkan (Raw SQL)";

}

}
```

Jalankan di browser:

http://127.0.0.1:8000/insert-data

Select Data Menggunakan Query Builder

Query Builder digunakan untuk mengambil data database dengan metode Laravel.

```
public function selectData()

{

$data = DB::table('mahasiswas')->get();

return $data;

}
```

Akses:

http://127.0.0.1:8000/select-data

Update Data Menggunakan Query Builder

Query Builder juga dapat digunakan untuk memperbarui data.

```
public function updateData()

{

DB::table('mahasiswas')
```

```
->where('nim', '23111040')

->update(['fakultas' => 'Fakultas Ilmu Komputer']);

return "Data berhasil diupdate (Query Builder)";

}
```

Akses:

http://127.0.0.1:8000/update-data

Delete Data Menggunakan Query Builder

Untuk menghapus data:

```
public function deleteData()

{

DB::table('mahasiswas')

->where('nim', '23111040')

->delete();

return "Data berhasil dihapus";

}
```

Akses:

http://127.0.0.1:8000/delete-data

Insert Data Menggunakan Eloquent ORM

Eloquent ORM menggunakan model sebagai representasi tabel.

Membuat Model, gunakan php artisan make:model Mahasiswa

```
app/Models/Mahasiswa.php

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Mahasiswa extends Model

{

use HasFactory;
```

```
protected $table = 'mahasiswas';

protected $fillable = [

'nim',

'nama_lengkap',

'tempat_lahir',

'tanggal_lahir',

'alamat',

'fakultas',

'jurusan'

];

}
```

Insert Data dengan Eloquent

Tambahan dalam controller

```
app\Http\Controllers\MahasiswaController.php

use App\Models\Mahasiswa;

public function insertEloquent()

{

Mahasiswa::create([

'nim' => '23111041',

'nama_lengkap' => 'Alya Rabani',

'tempat_lahir' => 'Purbalingga',

'tanggal_lahir' => '2003-02-02',

'fakultas' => 'Informatika',

'jurusan' => 'Teknik Informatika'

]);

return "Data berhasil ditambahkan (Eloquent ORM)";

}
```

Tambahan pada route, routes\web.php

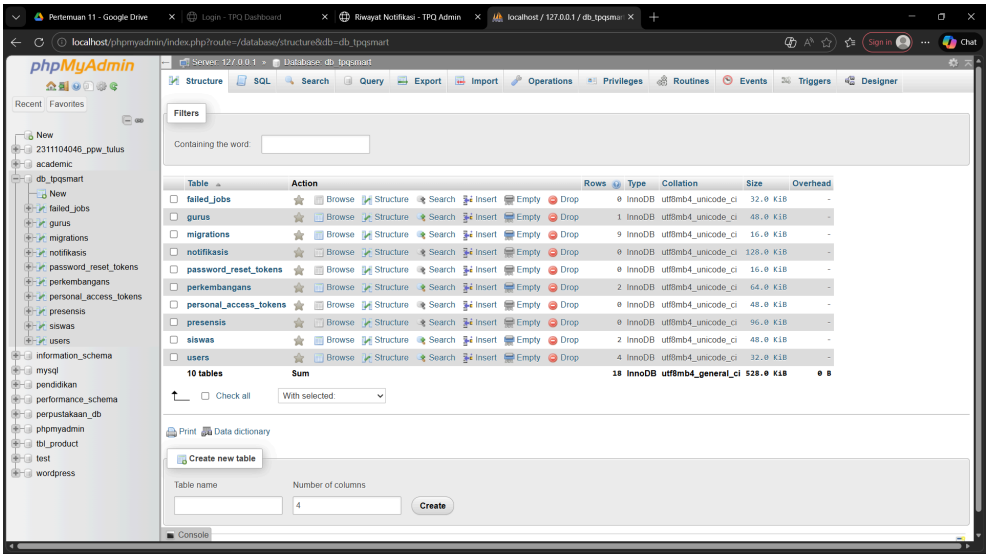
```
Route::get('/insert-eloquent', [MahasiswaController::class, 'insertEloquent']);
```

Akses:

http://127.0.0.1:8000/insert-eloquent

2.2 UNGUIDED (Tugas Mandiri)

Pada bagian unguided ini, saya mengimplementasikan materi Migration serta pengolahan data database menggunakan framework Laravel dengan mengacu pada konteks tugas besar yang sedang dikerjakan, yaitu sistem pengelolaan data santri dan operasional pada TPQSmart. Sistem tugas besar tersebut memiliki struktur database yang cukup kompleks dan terdiri dari banyak tabel yang saling berelasi, serta melibatkan berbagai jenis pengguna, seperti admin, guru, dan orang tua santri. Namun, untuk keperluan praktikum Modul 12, struktur database tersebut disederhanakan agar fokus pembelajaran dapat diarahkan pada pemahaman konsep migration dan implementasi pengolahan data menggunakan Laravel.



Sebagai bentuk penyederhanaan, saya hanya menggunakan satu tabel utama, yaitu tabel siswas, yang merepresentasikan data inti yang dikelola dalam sistem TPQSmart. Pemilihan tabel ini didasarkan pada peran admin dan guru yang paling dominan dalam melakukan operasi CRUD pada sistem tugas besar, seperti mencatat data santri baru, mengelola absensi, atau memproses laporan perkembangan. Dengan demikian, meskipun hanya menggunakan satu tabel, konteks dan alur sistem tetap selaras dengan rancangan sistem TPQSmart yang sebenarnya.

Salah satu contoh database di folder migrations:

```php
<?php


use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;


return new class extends Migration

{

    /**

     * Run the migrations.

     */

    public function up(): void

{

    Schema::create('siswas', function (Blueprint $table) {

        $table->id();

        // Relasi ke tabel users (Menghubungkan profil dengan
akun login)


$table->foreignId('user_id')->constrained('users')->onDelete('cas
cade');

        $table->string('nis')->unique(); // ID Siswa

        $table->string('nama_lengkap');

        $table->string('kelas');

        $table->enum('jenis_kelamin', ['Laki-laki',
'Perempuan']);

        $table->string('tempat_lahir');
```

```
            $table->date('tanggal_lahir'); // Pake tipe date buat
kalender
            $table->text('alamat');
            $table->string('no_hp');
            $table->string('foto')->nullable();
            $table->timestamps();
        });
    }


    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        //
    }
};
```

Salah satu contoh controller dari folder app/http yang mengontrol table siswas

```
<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use App\Models\User;
use App\Models\Siswa;
```

```php
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;


class SiswaController extends Controller
{
    public function index()
    {
        $siswaList = Siswa::with('user')->get();
        return view('admin.data_pengguna', compact('siswaList'));
    }


    /**
     * Show detail siswa
     */
    public function show($id)
    {
        // Ambil data siswa dengan relasi user
        $siswa = Siswa::with('user')->findOrFail($id);


        // Debug - hapus setelah berhasil
        \Log::info('Detail Siswa:', [
            'id' => $siswa->id,
            'nama' => $siswa->nama_lengkap,
            'user_id' => $siswa->user_id
        ]);
```

```
        return view('admin.data_pengguna.siswa.detail_siswa',
compact('siswa'));
    }

    public function store(Request $request)
    {
        Log::info('Siswa Store Request:', $request->all());

        // 1. Validasi Input
        $validated = $request->validate([
            'username' => 'required|unique:users,username|min:5',
            'password' => 'required|min:8',
            'idSiswa' => 'required|unique:siswas,nis',
            'nama' => 'required|string|max:255',
            'kelas' => 'required|string',
            'jenisKelamin' => 'required|in:Laki-laki,Perempuan',
            'tempat_lahir' => 'required|string|max:255',
            'tanggal_lahir' => 'required|date',
            'alamat' => 'required|string',
            'no_hp' => 'required|string|min:10',
            'foto' =>
'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        ], [
            'username.required' => 'Username wajib diisi',
            'username.unique' => 'Username sudah digunakan',
            'username.min' => 'Username minimal 5 karakter',
            'password.required' => 'Password wajib diisi',
```

```
            'password.min' => 'Password minimal 8 karakter',
            'idSiswa.required' => 'ID Siswa wajib diisi',
            'idSiswa.unique' => 'ID Siswa sudah digunakan',
            'nama.required' => 'Nama lengkap wajib diisi',
            'kelas.required' => 'Kelas wajib dipilih',
            'jenisKelamin.required' => 'Jenis kelamin wajib
dipilih',
            'tempat_lahir.required' => 'Tempat lahir wajib
diisi',
            'tanggal_lahir.required' => 'Tanggal lahir wajib
diisi',
            'tanggal_lahir.date' => 'Format tanggal lahir tidak
valid',
            'alamat.required' => 'Alamat wajib diisi',
            'no_hp.required' => 'Nomor handphone orang tua wajib
diisi',
            'no_hp.min' => 'Nomor handphone minimal 10 digit',
            'foto.image' => 'File harus berupa gambar',
            'foto.max' => 'Ukuran foto maksimal 2MB',
        ]);

        try {
            DB::beginTransaction();

            // 2. Handle foto upload
            $fotoPath = null;
            if ($request->hasFile('foto')) {
                $foto = $request->file('foto');
```

```
                $fotoName = time() . '_' . $validated['idSiswa']
. '.' . $foto->getClientOriginalExtension();

                // ✅ Simpan ke storage/app/public/photos/siswa

                $fotoPath = $foto->storeAs('photos/siswa',
$fotoName, 'public');

            }



            // 3. Simpan ke tabel Users (untuk login)

            $user = User::create([

                'name'     => $validated['nama'],

                'username' => $validated['username'],

                'password' => Hash::make($validated['password']),

                'role'     => 'siswa',

            ]);



            // 4. Simpan ke tabel Siswas (biodata) - ✅ FOTO
DISIMPAN DI SINI

            Siswa::create([

                'user_id'       => $user->id,

                'nis'           => $validated['idSiswa'],

                'nama_lengkap'  => $validated['nama'],

                'kelas'         => $validated['kelas'],

                'jenis_kelamin' => $validated['jenisKelamin'],

                'tempat_lahir'  => $validated['tempat_lahir'],

                'tanggal_lahir' => $validated['tanggal_lahir'],

                'alamat'        => $validated['alamat'],

                'no_hp'         => $validated['no_hp'],
```

```
                'foto'          => $fotoPath, // ✅ SIMPAN FOTO
DI TABEL SISWA
            ]);

            DB::commit();

            Log::info('Siswa created successfully:', ['user_id'
=> $user->id]);

            return response()->json([
                'success' => true,
                'message' => 'Data siswa berhasil ditambahkan!'
            ], 200);

        } catch (\Exception $e) {
            DB::rollback();

            Log::error('Error creating siswa:', [
                'message' => $e->getMessage(),
                'trace' => $e->getTraceAsString()
            ]);

            return response()->json([
                'success' => false,
                'message' => 'Terjadi kesalahan: ' .
$e->getMessage()
            ], 500);
        }
```

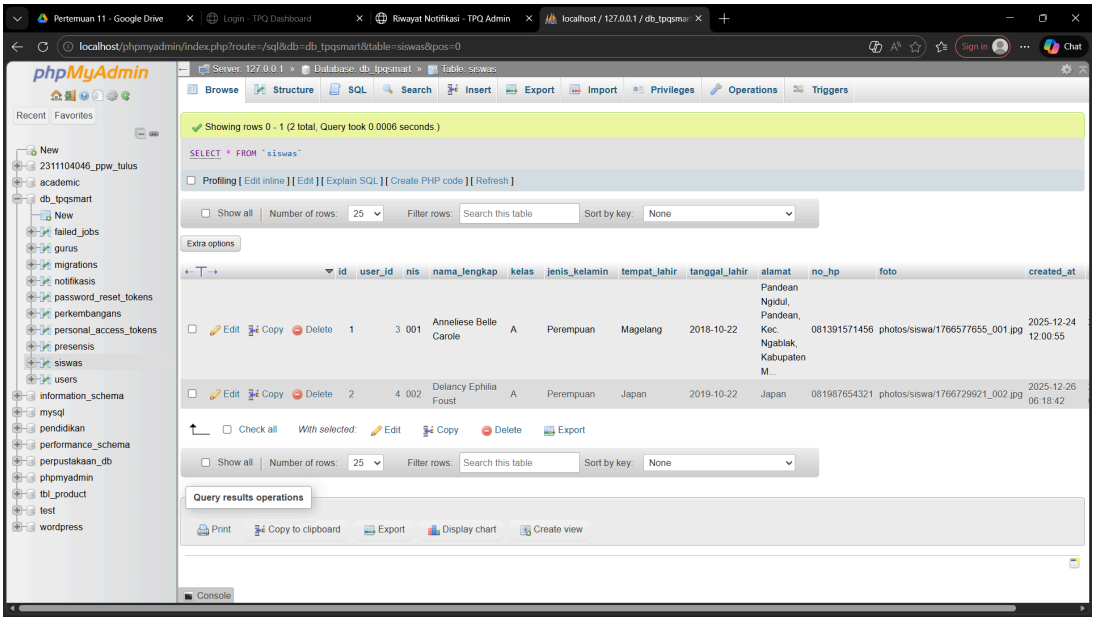
```
    }


    /**
     * Show the form for editing siswa
     */
    public function edit($id)
    {
        $siswa = Siswa::with('user')->findOrFail($id);
        return view('admin.data_pengguna.siswa.edit_siswa', compact('siswa'));
    }


    /**
     * Update siswa data
     */
    public function update(Request $request, $id)
    {
        Log::info('Siswa Update Request:', $request->all());


        $siswa = Siswa::with('user')->findOrFail($id);


        // Validasi Input
        $validated = $request->validate([
            'username' => 'required|min:5|unique:users,username,' . $siswa->user_id,
            'password' => 'nullable|min:8',
            'idSiswa' => 'required|unique:siswas,nis,' . $id,
            'nama' => 'required|string|max:255',
```

```
            'kelas' => 'required|string',
            'jenisKelamin' => 'required|in:Laki-laki,Perempuan',
            'tempat_lahir' => 'required|string|max:255',
            'tanggal_lahir' => 'required|date',
            'alamat' => 'required|string',
            'no_hp' => 'required|string|min:10',
            'foto' =>
'nullable|image|mimes:jpeg,png,jpg,gif|max:2048',
        ]);

        try {
            DB::beginTransaction();

            // Handle foto upload baru
            if ($request->hasFile('foto')) {
                // Hapus foto lama jika ada
                if ($siswa->foto &&
\Storage::disk('public')->exists($siswa->foto)) {
\Storage::disk('public')->delete($siswa->foto);
                }

                $foto = $request->file('foto');
                $fotoName = time() . '_' . $validated['idSiswa']
. '.' . $foto->getClientOriginalExtension();
                $validated['foto'] =
$foto->storeAs('photos/siswa', $fotoName, 'public');
            }
```

```
            // Update User
            $userData = [
                'name' => $validated['nama'],
                'username' => $validated['username'],
            ];

            if ($request->filled('password')) {
                $userData['password'] =
Hash::make($validated['password']);
            }

            $siswa->user->update($userData);

            // Update Siswa
            $siswa->update([
                'nis' => $validated['idSiswa'],
                'nama_lengkap' => $validated['nama'],
                'kelas' => $validated['kelas'],
                'jenis_kelamin' => $validated['jenisKelamin'],
                'tempat_lahir' => $validated['tempat_lahir'],
                'tanggal_lahir' => $validated['tanggal_lahir'],
                'alamat' => $validated['alamat'],
                'no_hp' => $validated['no_hp'],
                'foto' => $validated['foto'] ?? $siswa->foto,
            ]);

            DB::commit();
```

```
            return response()->json([
                'success' => true,
                'message' => 'Data siswa berhasil diperbarui!'
            ], 200);

        } catch (\Exception $e) {
            DB::rollback();

            Log::error('Error updating siswa:', [
                'message' => $e->getMessage(),
                'trace' => $e->getTraceAsString()
            ]);

            return response()->json([
                'success' => false,
                'message' => 'Terjadi kesalahan: ' .
$e->getMessage()
            ], 500);
        }
    }

    /**
     * Delete siswa
     */
    public function destroy($id)
    {
```

```
        try {
            $siswa = Siswa::with('user')->findOrFail($id);

            // Hapus foto jika ada
            if ($siswa->foto &&
\Storage::disk('public')->exists($siswa->foto)) {
                \Storage::disk('public')->delete($siswa->foto);
            }

            // Hapus user (cascade akan hapus siswa juga)
            $siswa->user->delete();

            return redirect()->route('admin.data_pengguna')
                             ->with('success', 'Data siswa berhasil
dihapus!');

        } catch (\Exception $e) {
            Log::error('Error deleting siswa:', [
                'message' => $e->getMessage()
            ]);

            return redirect()->route('admin.data_pengguna')
                             ->with('error', 'Gagal menghapus data
siswa!');
        }
    }
}
```


Pertemuan 11 - Google Drive
Login - TPQ Dashboard
Riwayat Notifikasi - TPQ Admin
localhost / 127.0.0.1 / db_tpqsmart
localhost/phpmyadmin/index.php?route=/sql&db=db_tpqsmart&table=siswas&pos=0
phpMyAdmin
Recent Favorites
New
2311104046_ppw_tulus
academic
db_tpqsmart
New
failed_jobs
gurus
migrations
notifikasis
password_reset_tokens
perkembangans
personal_access_tokens
presensis
siswas
users
information_schema
mysql
pendidikan
performance_schema
perpustakaan_db
phpmyadmin
tbl_product
test
wordpress
Server: 127.0.0.1 » Database: db_tpqsmart » Table: siswas
Browse Structure SQL Search Insert Export Import Privileges Operations Triggers
Showing rows 0 - 1 (2 total, Query took 0.0006 seconds.)
SELECT * FROM `siswas`
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
id user_id nis nama_lengkap kelas jenis_kelamin tempat_lahir tanggal_lahir alamat no_hp foto created_at
Edit Copy Delete 1 3 001 Anneliese Belle Carole A Perempuan Magelang 2018-10-22 Pandean Ngidul, Pandean, Kec. Ngablak, Kabupaten M... 081391571456 photos/siswa/1766577655_001.jpg 2025-12-24 12:00:55
Edit Copy Delete 2 4 002 Delancy Ephilia Foust A Perempuan Japan 2019-10-22 Japan 081987654321 photos/siswa/1766729921_002.jpg 2025-12-26 06:18:42
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

BAB III
PENUTUP

KESIMPULAN

Kesimpulan dari praktikum Modul 12 menunjukkan bahwa Laravel menawarkan ekosistem yang komprehensif dan terorganisir untuk manajemen database melalui fitur Migration, DB Facade, Query Builder, serta Eloquent ORM. Fitur Migration sangat membantu dalam membangun dan mengelola skema database secara sistematis tanpa ketergantungan pada perintah SQL manual, sekaligus memberikan keamanan tambahan melalui fitur rollback untuk membatalkan perubahan jika terjadi kesalahan.

Di sisi lain, pengembang diberikan keleluasaan dalam memproses data melalui tiga metode berbeda. DB Facade (Raw SQL) menawarkan kontrol penuh dan fleksibilitas untuk menjalankan perintah SQL mentah secara langsung. Query Builder memberikan jalan tengah dengan menyediakan sintaks yang lebih ringkas, aman, dan mudah dibaca untuk menyusun perintah database. Terakhir, Eloquent ORM menyederhanakan manipulasi data dengan pendekatan berorientasi objek yang menjadikan kode program lebih bersih, modular, dan intuitif untuk dikembangkan.