LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL 12

MIGRATION

Oleh:

Aulia Jasifa Br Ginting 2311104060

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

A. Dasar Teori

Migration pada framework Laravel merupakan mekanisme pengelolaan struktur database berbasis kode yang berfungsi sebagai version control untuk database. Dengan migration, pengembang dapat membuat, mengubah, dan menghapus tabel maupun kolom database secara terstruktur tanpa harus langsung berinteraksi dengan database secara manual. Fitur ini sangat membantu dalam menjaga konsistensi struktur database antar lingkungan pengembangan, serta memungkinkan perubahan skema database dilakukan tanpa menghilangkan data yang sudah ada. Selain itu, migration mendukung proses rollback sehingga struktur database dapat dikembalikan ke kondisi sebelumnya apabila terjadi kesalahan, yang menjadikan pengembangan aplikasi lebih aman dan terkontrol.

Model dalam Laravel berperan sebagai representasi dari tabel database yang digunakan untuk mengelola data menggunakan konsep CRUD (Create, Read, Update, Delete). Model membantu memisahkan logika bisnis dari logika tampilan dan database sehingga kode menjadi lebih rapi, terstruktur, dan mudah dipelihara. Laravel menyediakan beberapa cara untuk mengakses database melalui model, yaitu DB Facade (Raw SQL), Query Builder, dan Eloquent ORM. Raw SQL digunakan untuk menjalankan perintah SQL secara langsung dan biasanya dipakai untuk query yang kompleks. Query Builder menyediakan cara yang lebih aman dan fleksibel dalam menyusun query SQL menggunakan metode-metode bawaan Laravel. Sementara itu, Eloquent ORM memungkinkan pengembang berinteraksi dengan database menggunakan pendekatan berbasis objek, sehingga setiap baris data direpresentasikan sebagai objek, yang membuat proses pengembangan aplikasi menjadi lebih cepat, efisien, dan aman dari serangan seperti SQL Injection.

B. Tujuan

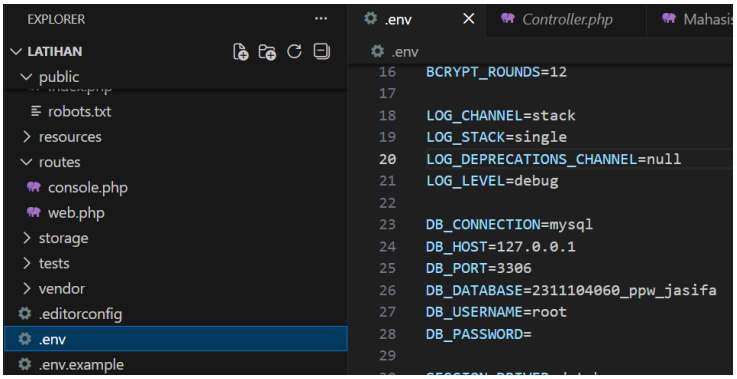
1. Memahami fungsi dan cara kerja model pada framework laravel
2. Mampu menerapkan migration pada aplikasi

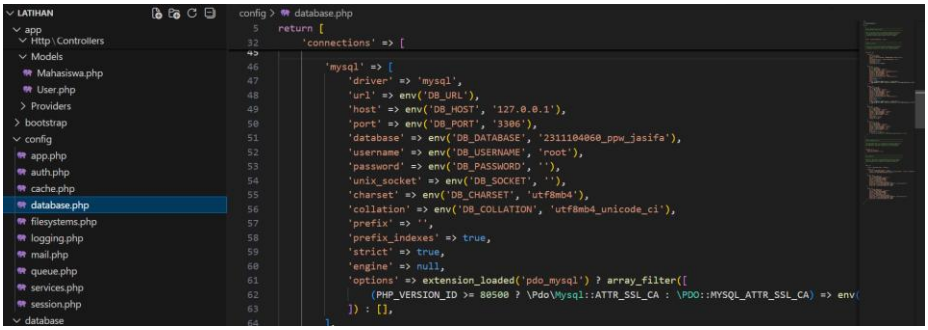
BAB II

HASIL

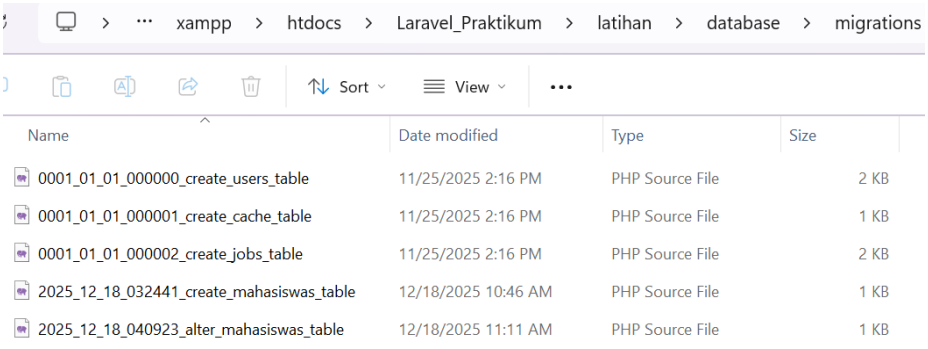
A. GUIDED (Praktikum Terbimbing)

1. File Konfigurasi

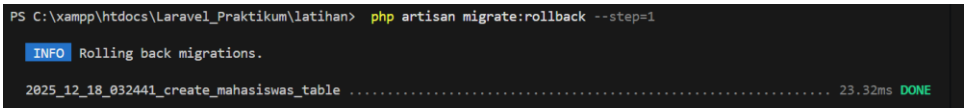




2. File migration bawaan laravel



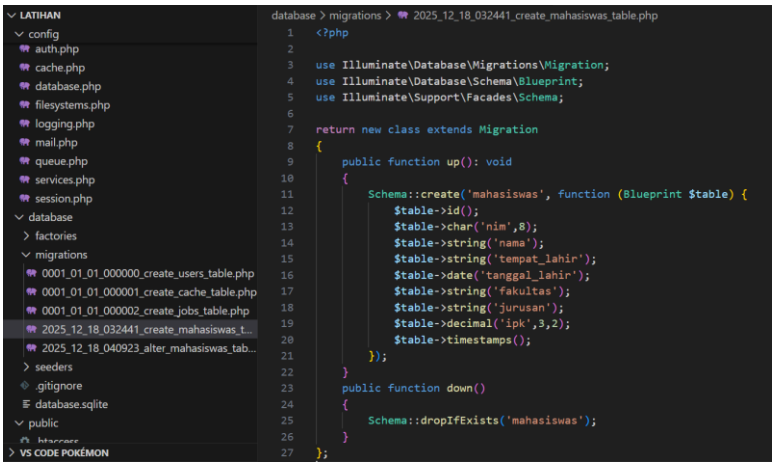
3. Migration Rollback

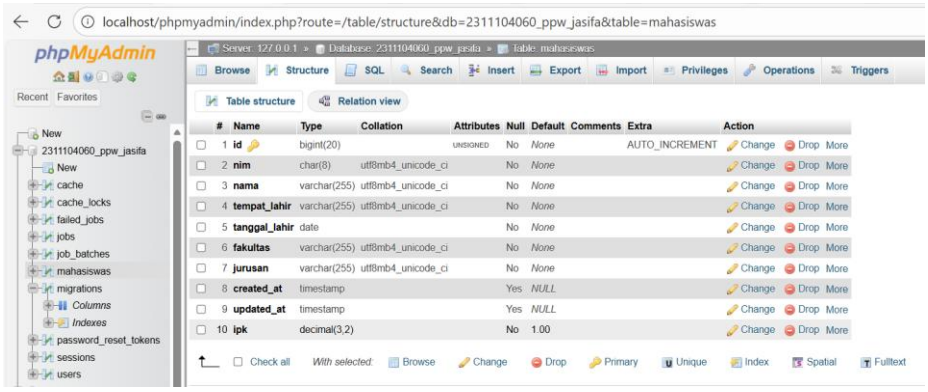


LATIHAN

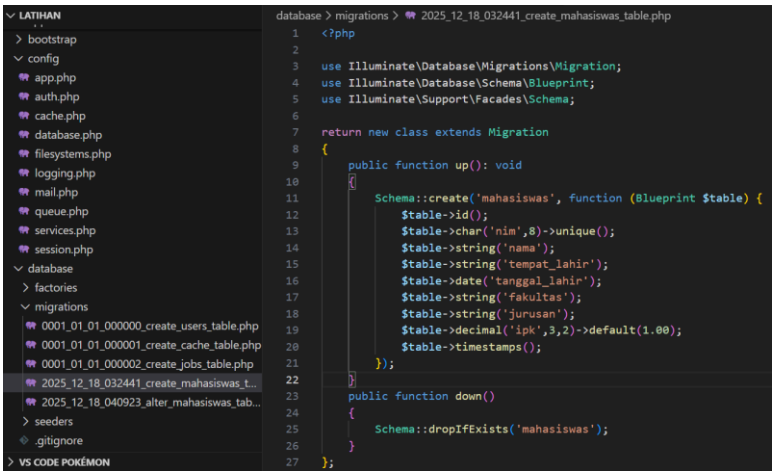
1. Membuat Migration

```
PS C:\xampp\htdocs\Laravel_Praktikum\latihan> php artisan make:migration create_mahasiswas_table --create=mahasiswas

  INFO  Migration [C:\xampp\htdocs\Laravel_Praktikum\latihan\database\migrations\2025_12_18_032441_create_mahasiswas_table.php] created successfully.
```





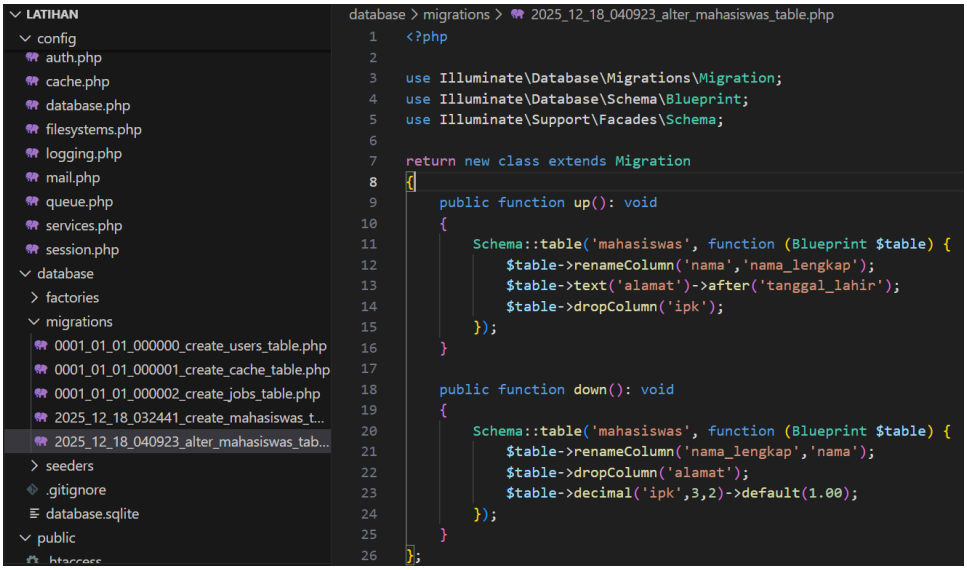
2. Modifikasi file migration



3. Alter table migration

```
PS C:\xampp\htdocs\Laravel_Praktikum\latihan> composer require doctrine/dbal
./composer.json has been updated
Running composer update doctrine/dbal
Loading composer repositories with package information
Updating dependencies
Lock file operations: 3 installs, 0 updates, 0 removals
  - Locking doctrine/dbal (4.4.1)
  - Locking doctrine/deprecations (1.1.5)
  - Locking psr/cache (3.0.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 3 installs, 0 updates, 0 removals
  - Downloading psr/cache (3.0.0)
  - Downloading doctrine/deprecations (1.1.5)
  - Downloading doctrine/dbal (4.4.1)
  - Installing psr/cache (3.0.0): Extracting archive
  - Installing doctrine/deprecations (1.1.5): Extracting archive
  - Installing doctrine/dbal (4.4.1): Extracting archive
Generating optimized autoload files
> Illuminate\Foundation\ComposerScripts::postAutoloadDump
> @php artisan package:discover --ansi
```

```
PS C:\xampp\htdocs\Laravel_Praktikum\latihan>  php artisan make:migration alter_mahasiswas_table --table=mahasiswas

  INFO  Migration [C:\xampp\htdocs\Laravel_Praktikum\latihan\database\migrations\2025_12_18_040923_alter_mahasiswas_table
.php] created successfully.
```



MODUL 12.2 (DB Facade , Eloquent ORM, Query Builder)

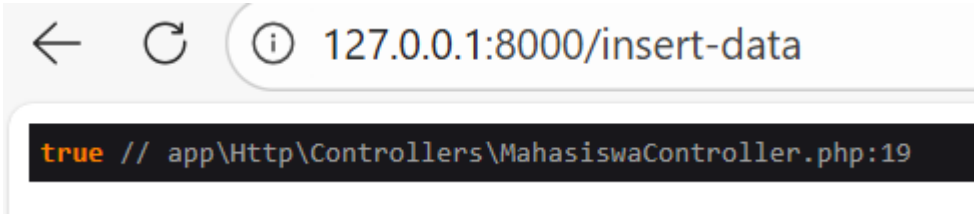
1. Input Data menggunakan raw SQL Queries

routes > web.php

```php
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\MahasiswaController;

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']);
```

app > Http > Controllers > MahasiswaController.php

```php
<?php
namespace App\Http\Controllers; use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

    class MahasiswaController extends Controller
{
        public function insertData()
    {
        $result = DB::insert("
            INSERT INTO mahasiswas
            (nim, nama_lengkap, tempat_lahir, tanggal_lahir, fakultas, jurusan, alamat)
            VALUES
            ('20104064', 'Muhammad Nur Hamada', 'Purwokerto', '2002-02-02', 'Teknik', 'Informatika',
        ");

        dump($result);
    }
}
```



2. Input Data menggunakan Query Builder

app > Http > Controllers > MahasiswaController.php

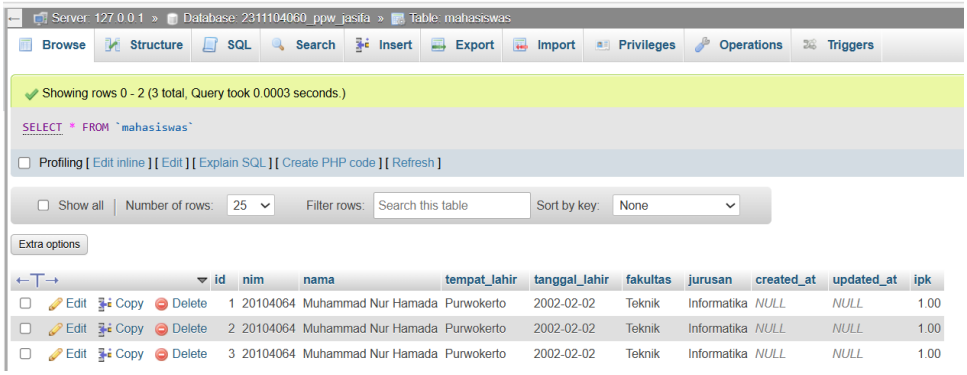
```php
<?php
namespace App\Http\Controllers; use Illuminate\Http\Request;

use Illuminate\Support\Facades\DB;

    class MahasiswaController extends Controller
{
        public function insertData(){
            $mahasiswa = new Mahasiswa;
            $mahasiswa->nim = '19003036';
            $mahasiswa->nama_lengkap = 'Muhammad Hamada';
            $mahasiswa->tempat_lahir = 'Bandung';
            $mahasiswa->tanggal_lahir = '2002-02-02';
            $mahasiswa->alamat = 'J1. Merdeka No. 10';
            $mahasiswa->fakultas = 'Teknik';
            $mahasiswa->jurusan = 'Informatika';
            $mahasiswa->save();

        }
}
```

3. Input Data menggunakan Eloquent ORM

app > Models > Mahasiswa.php

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class mahasiswas extends Model {
    protected $fillable = [
        'nim',
        'nama_lengkap',
        'tempat_lahir',
        'tanggal_lahir',
        'fakultas',
        'jurusan',
        'alamat'
    ];
}
```

```php
    public function insertData() {
        $mahasiswa = new Mahasiswa;
        $mahasiswa->nim = '19003036';
        $mahasiswa->nama_lengkap = 'Muhammad Hamada';
        $mahasiswa->tempat_lahir = 'Bandung';
        $mahasiswa->tanggal_lahir = '2002-02-02';
        $mahasiswa->alamat = 'Jl. Merdeka No. 10';
        $mahasiswa->fakultas = 'Teknik';
        $mahasiswa->jurusan = 'Informatika';
        $mahasiswa->save();

    }
}
```



Penjelasan:

Laravel berfungsi untuk mengelola struktur database dan melakukan manipulasi data secara terorganisir menggunakan fitur bawaan framework Laravel. Pada bagian migration, kode berfungsi untuk membuat, mengubah, dan menghapus tabel database melalui method up() dan down(). Method up() dijalankan saat perintah php artisan migrate dieksekusi, yang bertugas membangun atau memodifikasi struktur tabel sesuai dengan skema yang didefinisikan menggunakan Schema dan Blueprint. Sementara itu, method down() berfungsi sebagai kebalikan dari up(), yaitu mengembalikan perubahan struktur database saat proses rollback dijalankan. Mekanisme ini memungkinkan pengembang melakukan perubahan database secara aman, terkontrol, dan dapat dikembalikan ke kondisi sebelumnya jika terjadi kesalahan.

Pada pengolahan data, Laravel menyediakan tiga pendekatan utama, yaitu DB Facade (Raw SQL), Query Builder, dan Eloquent ORM. DB Facade

memungkinkan kode menjalankan perintah SQL secara langsung ke database, sehingga cocok digunakan untuk query sederhana maupun kompleks. Query Builder bekerja dengan cara menyusun query SQL menggunakan method Laravel, yang membuat kode lebih aman dan mengurangi risiko SQL Injection. Sementara itu, Eloquent ORM bekerja dengan merepresentasikan tabel database sebagai sebuah model berbasis objek, sehingga setiap data diperlakukan sebagai objek yang dapat diakses dan dimanipulasi melalui properti dan method. Ketiga pendekatan ini memberikan fleksibilitas kepada pengembang dalam memilih metode pengolahan data sesuai kebutuhan aplikasi, sekaligus menjaga keamanan, efisiensi, dan keterbacaan kode program.

B. UNGUIDED (Tugas Mandiri)

(Bagian ini berisi latihan atau tugas tambahan yang diberikan di akhir modul. Tuliskan nomor soal, isi soalnya, source code jawaban, screenshot output, dan penjelasan setiap nomor.)

1. **Soal 1:**

   Buatlah file migration untuk database yang akan digunakan pada tugas besar di kelas teori.

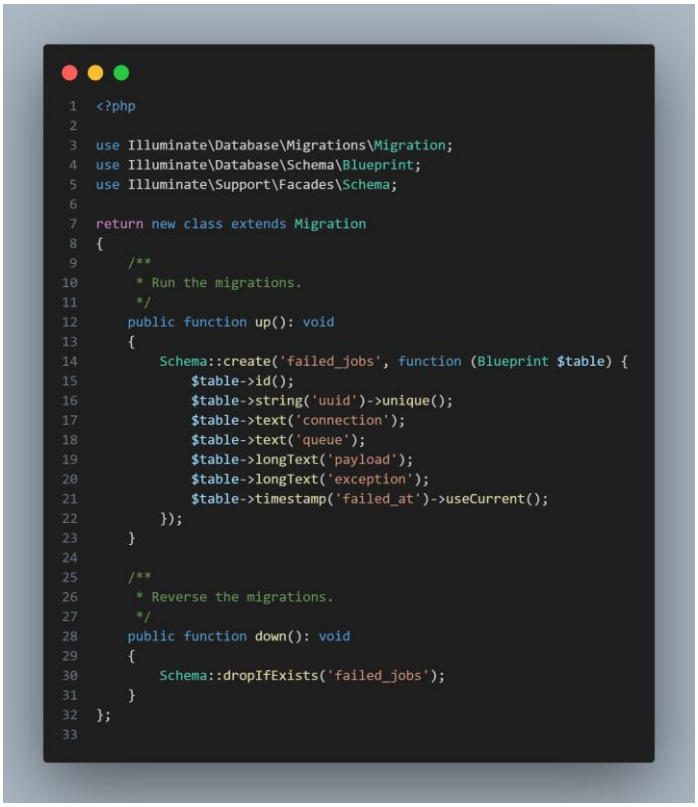
   Class database/migration/2014_10_12_000000_create_users_table.php

```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('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('username')->unique(); // Ganti email jadi username biar sinkron sama JS
            $table->string('role');
            $table->string('password');
            $table->rememberToken();
            $table->timestamps();
        });
    }

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

Class migration/2014_10_12_100000_create_password_reset_tokens_table.php

```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('password_reset_tokens', function (Blueprint $table) {
            $table->string('email')->primary();
            $table->string('token');
            $table->timestamp('created_at')->nullable();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('password_reset_tokens');
    }
};

```

Class database/migration/2019_08_19_000000_create_failed_jobs_table.php



```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('failed_jobs', function (Blueprint $table) {
            $table->id();
            $table->string('uuid')->unique();
            $table->text('connection');
            $table->text('queue');
            $table->longText('payload');
            $table->longText('exception');
            $table->timestamp('failed_at')->useCurrent();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('failed_jobs');
    }
};

```

Class migration/2019_12_14_000001_create_personal_access_tokens_table.php

```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('personal_access_tokens', function (Blueprint $table) {
            $table->id();
            $table->morphs('tokenable');
            $table->string('name');
            $table->string('token', 64)->unique();
            $table->text('abilities')->nullable();
            $table->timestamp('last_used_at')->nullable();
            $table->timestamp('expires_at')->nullable();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('personal_access_tokens');
    }
};

```

Class database/migration2025_12_23_034721_create_gurus_table.php

```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('gurus', function (Blueprint $table) {
        $table->id();
        // Relasi ke users (PENTING: ini yang nyambungin ke password/usn di tabel users)
        $table->foreignId('user_id')->constrained('users')->onDelete('cascade');

        // Data Pribadi Guru
        $table->string('nip')->unique(); // Nomor Induk Pegawai / ID Guru
        $table->string('nama_lengkap');
        $table->string('kelas')->nullable(); // Kelas yang diajar (bisa multiple, pisah koma)
        $table->enum('jenis_kelamin', ['Laki-laki', 'Perempuan']);
        $table->string('tempat_lahir');
        $table->date('tanggal_lahir');
        $table->text('alamat');
        $table->string('no_hp');
        $table->string('foto')->nullable();
        $table->timestamps();
    });
}
    /**
     * Reverse the migrations.
     */
    public function down(): void
    {
        Schema::dropIfExists('gurus');
    }
};

```

Class database/migration/2025_12_23_034831_create_siswas_table.php

```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('cascade');
        $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
    {
        //
    }
};

```

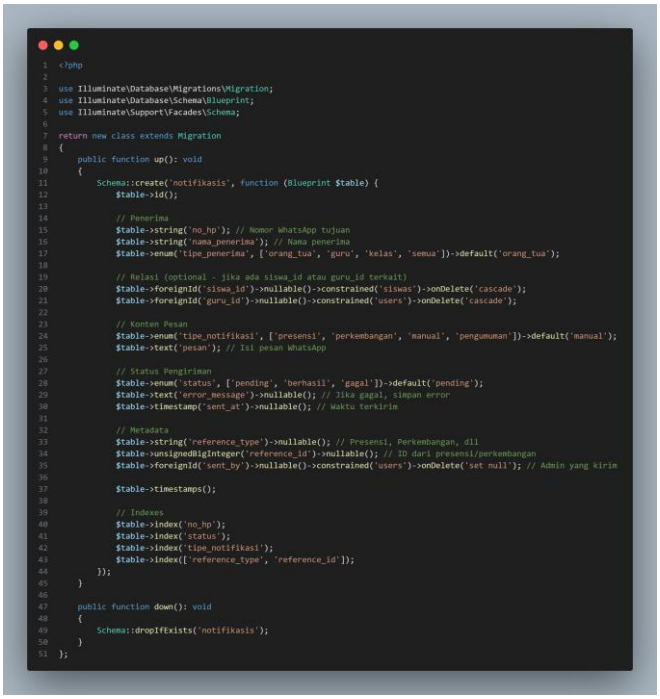
Class database/migration/2025_12_25_054310_create_presensis_table.php

```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('presensis', function (Blueprint $table) {
            $table->id();

            // Foreign Keys
            $table->foreignId('siswa_id')->constrained('siswas')->onDelete('cascade');

            // Data Presensi
            $table->date('tanggal'); // Tanggal presensi
            $table->enum('status', ['hadir', 'izin', 'sakit', 'alpha'])->default('alpha');
            $table->time('waktu')->nullable(); // Waktu absen (jam masuk)
            $table->text('keterangan')->nullable(); // Keterangan tambahan (misal: alasan izin)

            // Metadata
            $table->string('dicatat_oleh')->nullable(); // Guru yang mencatat

            $table->timestamps();

            // Index untuk query cepat
            $table->index(['siswa_id', 'tanggal']);
            $table->index('tanggal');
            $table->index('status');

            // Unique constraint: 1 siswa hanya bisa 1 presensi per hari
            $table->unique(['siswa_id', 'tanggal']);
        });
    }

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

Class migration/2025_12_26_081424_create_perkembangans_table.php

```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('perkembangans', function (Blueprint $table) {
            $table->id();

            // Foreign Keys
            $table->foreignId('siswa_id')->constrained('siswas')->onDelete('cascade');

            // Data Perkembangan
            $table->date('tanggal'); // Tanggal pencatatan

            // Tilawati/Bacaan
            $table->string('tilawati')->nullable(); // Jilid 1-6 / Al-Quran
            $table->string('halaman')->nullable(); // Halaman berapa

            // Kemampuan/Penilaian
            $table->enum('kemampuan', ['Sangat Baik', 'Baik', 'Cukup', 'Perlu Bimbingan'])->nullable();

            // Hafalan
            $table->string('hafalan')->nullable(); // Surah yang dihafal
            $table->string('ayat')->nullable(); // Ayat berapa

            // Perilaku & Catatan
            $table->text('tata_krama')->nullable(); // Catatan perilaku
            $table->text('catatan')->nullable(); // Catatan tambahan guru

            // Metadata
            $table->string('dicatat_oleh')->nullable(); // Guru yang mencatat

            $table->timestamps();

            // Index untuk query cepat
            $table->index(['siswa_id', 'tanggal']);
            $table->index('tanggal');

            // Unique constraint: 1 siswa 1 catatan per hari
            $table->unique(['siswa_id', 'tanggal']);
        });
    }

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

Class database/migration/2025_12_30_024144_create_notifikasis_table.php



```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up(): void
    {
        Schema::create('notifikasis', function (Blueprint $table) {
            $table->id();

            // Penerima
            $table->string('no_hp'); // Nomor WhatsApp tujuan
            $table->string('nama_penerima'); // Nama penerima
            $table->enum('tipe_penerima', ['orang_tua', 'guru', 'kelas', 'semua'])->default('orang_tua');

            // Relasi (optional - jika ada siswa_id atau guru_id terkait)
            $table->foreignId('siswa_id')->nullable()->constrained('siswas')->onDelete('cascade');
            $table->foreignId('guru_id')->nullable()->constrained('users')->onDelete('cascade');

            // Konten Pesan
            $table->enum('tipe_notifikasi', ['presensi', 'perkembangan', 'manual', 'pengumuman'])->default('manual');
            $table->text('pesan'); // Isi pesan WhatsApp

            // Status Pengiriman
            $table->enum('status', ['pending', 'berhasil', 'gagal'])->default('pending');
            $table->text('error_message')->nullable(); // Jika gagal, simpan error
            $table->timestamp('sent_at')->nullable(); // Waktu terkirim

            // Metadata
            $table->string('reference_type')->nullable(); // Presensi, Perkembangan, dll
            $table->unsignedBigInteger('reference_id')->nullable(); // ID dari presensi/perkembangan
            $table->foreignId('sent_by')->nullable()->constrained('users')->onDelete('set null'); // Admin yang kirim

            $table->timestamps();

            // Indexes
            $table->index('no_hp');
            $table->index('status');
            $table->index('tipe_notifikasi');
            $table->index(['reference_type', 'reference_id']);
        });
    }

    public function down(): void
    {
        Schema::dropIfExists('notifikasis');
    }
};
```

Output:

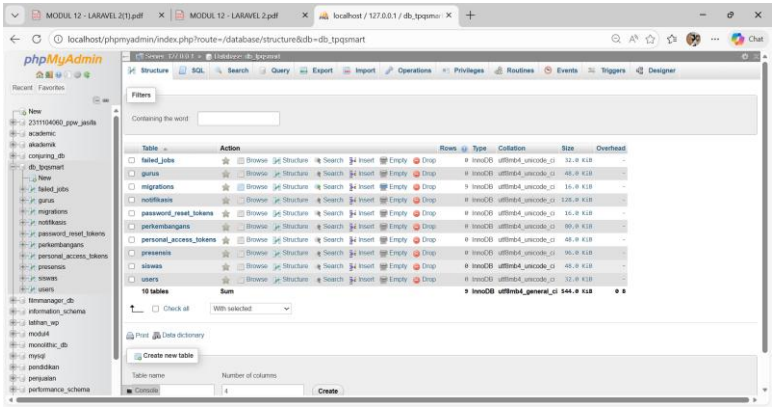


Table users

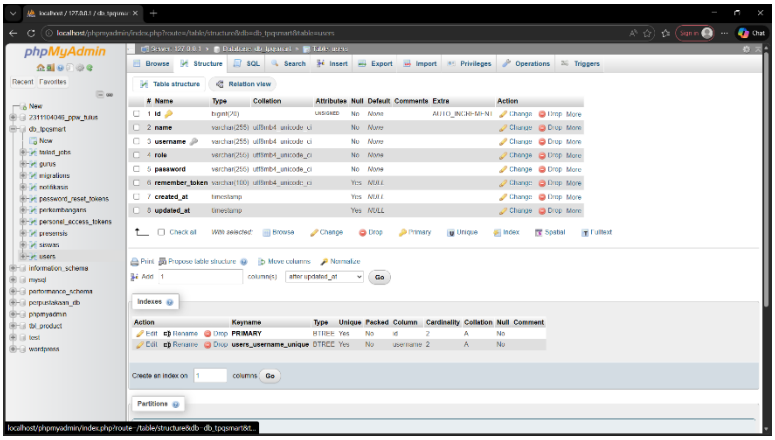


Table gurus

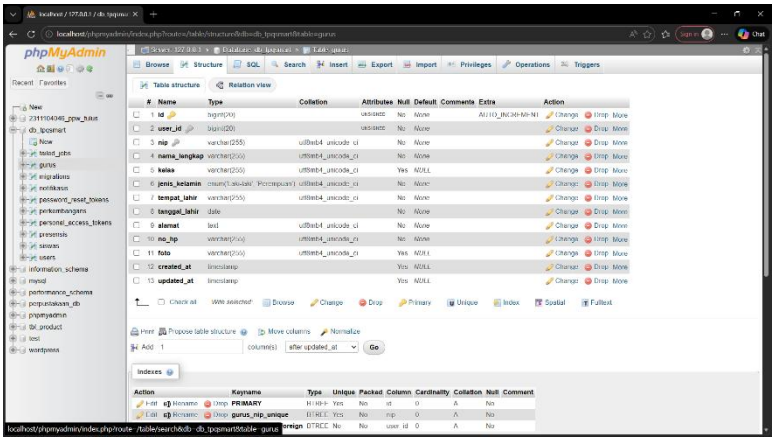


Table siswas

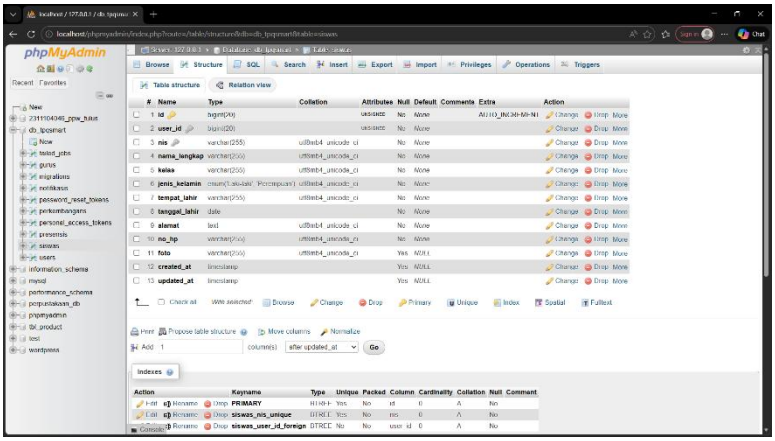


Table notifikasis

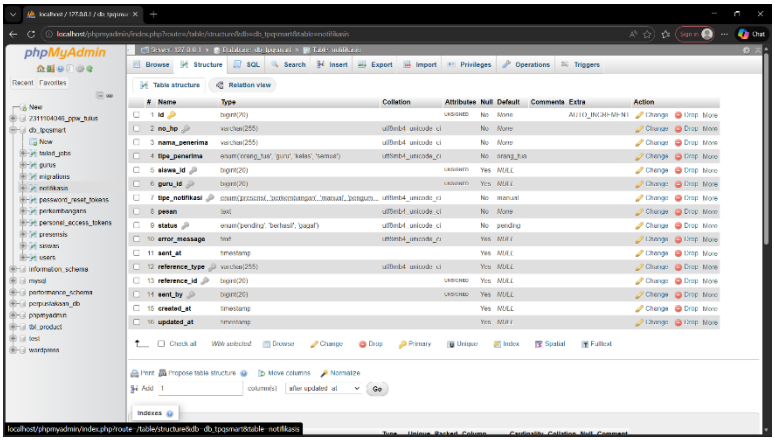


Table perkembangans

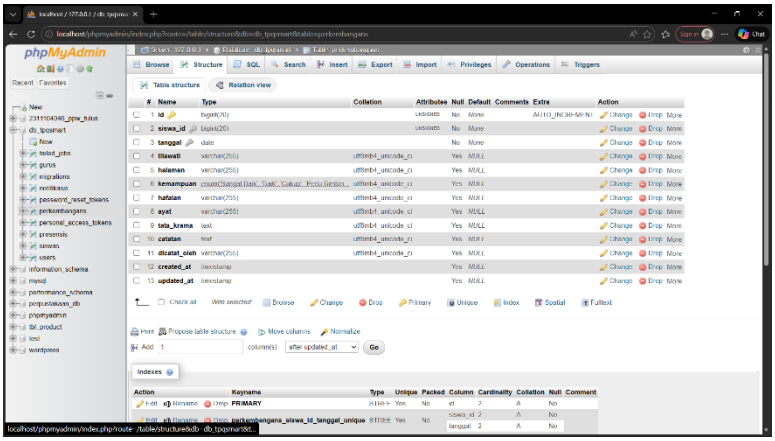
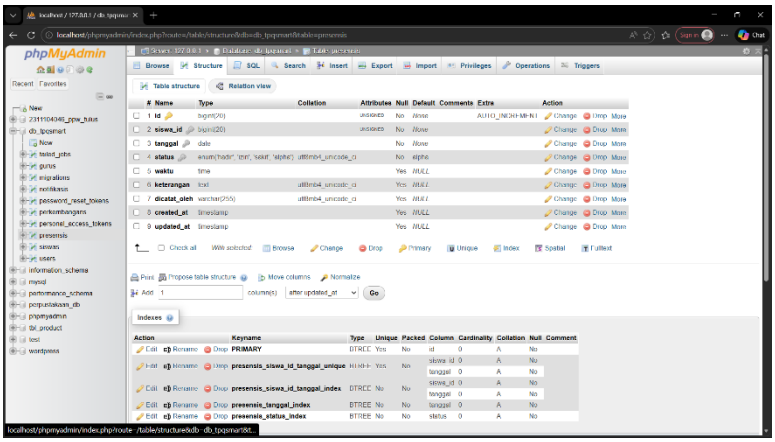


Table presensis



Penjelasan:

Migration tersebut berfungsi sebagai mekanisme untuk membangun, mengubah, dan mengelola struktur basis data aplikasi Laravel secara sistematis. Setiap migration berisi definisi pembuatan tabel tertentu, baik tabel bawaan Laravel seperti users, password_reset_tokens, failed_jobs, dan personal_access_tokens, maupun tabel khusus aplikasi seperti guru, siswa, presensi, perkembangan, dan notifikasi. Di dalam setiap file, method up() digunakan untuk mendefinisikan struktur tabel, termasuk nama kolom, tipe data, primary key, serta relasi antar tabel menggunakan foreign key. Ketika perintah php artisan migrate dijalankan, Laravel akan mengeksekusi migration tersebut secara berurutan berdasarkan timestamp pada nama file, sehingga database terbentuk sesuai rancangan sistem. Sebaliknya, method down() memungkinkan penghapusan tabel saat dilakukan rollback, sehingga perubahan database dapat dikontrol dan dikembalikan ke kondisi sebelumnya. Dengan penggunaan migration, pengelolaan struktur database menjadi lebih rapi, konsisten, dan memudahkan kolaborasi dalam pengembangan aplikasi.

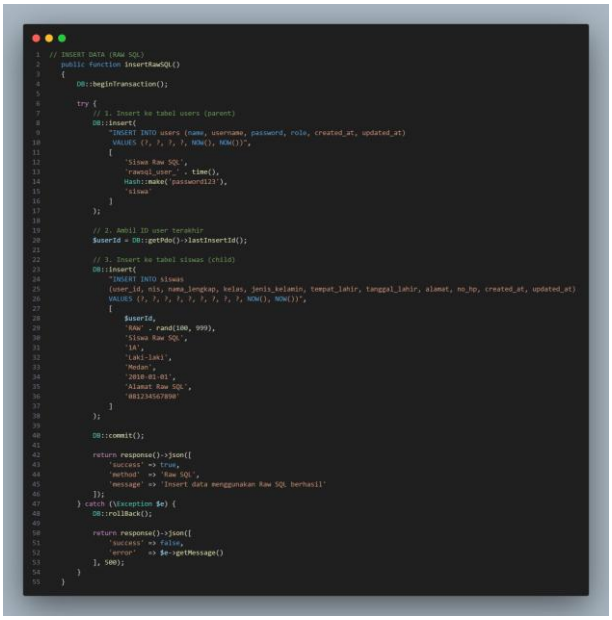
2. **Soal 2:**

   Buatlah fungsi-fungsi berikut untuk menyelesaikan tugas besar di matakuliah teori:

   1) Buat fungsi untuk insert data menggunakan raw SQL Queries
   2) Buat fungsi untuk insert data menggunakan Query Builder
   3) Buat fungsi untuk insert data menggunakan Eloquent ORM

Jawaban:

1) raw SQL Queries



```php
// INSERT DATA (Raw SQL)
public function insertRawSQL()
{
    DB::beginTransaction();

    try {
        // 1. Insert ke tabel users (parent)
        DB::insert(
            "INSERT INTO users (name, username, password, role, created_at, updated_at)
             VALUES (?, ?, ?, ?, NOW(), NOW())",
            [
                'Siswa Raw SQL',
                'rawsql_user_' . time(),
                Hash::make('password123'),
                'siswa'
            ]
        );

        // 2. Ambil ID user terakhir
        $userId = DB::getPdo()->lastInsertId();

        // 3. Insert ke tabel siswas (child)
        DB::insert(
            "INSERT INTO siswas
            (user_id, nis, nama_lengkap, kelas, jenis_kelamin, tempat_lahir, tanggal_lahir, alamat, no_hp, created_at, updated_at)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, NOW(), NOW())",
            [
                $userId,
                'RAW' . rand(100, 999),
                'Siswa Raw SQL',
                '1A',
                'Laki-laki',
                'Medan',
                '2010-01-01',
                'Alamat Raw SQL',
                '08123456789'
            ]
        );

        DB::commit();

        return response()->json([
            'success' => true,
            'method' => 'Raw SQL',
            'message' => 'Insert data menggunakan Raw SQL berhasil'
        ]);
    } catch (\Exception $e) {
        DB::rollBack();

        return response()->json([
            'success' => false,
            'error'   => $e->getMessage()
        ], 500);
    }
}
```

2) Query Builder

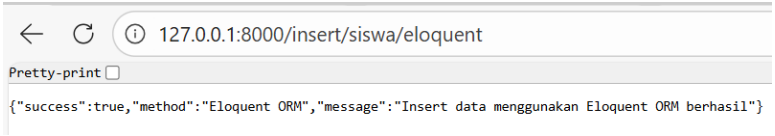
```php
// INSERT DATA (QUERY BUILDER)
public function insertQueryBuilder()
{
    DB::beginTransaction();

    try {
        // 1. Insert users
        $userId = DB::table('users')->insertGetId([
            'name'       => 'Siswa Query Builder',
            'username'   => 'qb_user_' . time(),
            'password'   => Hash::make('password123'),
            'role'       => 'siswa',
            'created_at' => now(),
            'updated_at' => now(),
        ]);

        // 2. Insert siswas
        DB::table('siswas')->insert([
            'user_id'       => $userId,
            'nis'           => 'QB' . rand(100, 999),
            'nama_lengkap'  => 'Siswa Query Builder',
            'kelas'         => '1B',
            'jenis_kelamin' => 'Perempuan',
            'tempat_lahir'  => 'Binjai',
            'tanggal_lahir' => '2011-02-02',
            'alamat'        => 'Alamat Query Builder',
            'no_hp'         => '081298765432',
            'created_at'    => now(),
            'updated_at'    => now(),
        ]);

        DB::commit();

        return response()->json([
            'success' => true,
            'method'  => 'Query Builder',
            'message' => 'Insert data menggunakan Query Builder berhasil'
        ]);
    } catch (\Exception $e) {
        DB::rollBack();

        return response()->json([
            'success' => false,
            'error'   => $e->getMessage()
        ], 500);
    }
}
```

3) Eloquent ORM

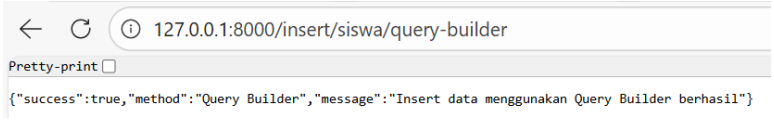
```php
// INSERT DATA (ELOQUENT ORM)
    public function insertEloquent()
    {
        DB::beginTransaction();

        try {
            $user = User::create([
                'name'     => 'Siswa Eloquent ORM',
                'username' => 'eloquent_user_' . time(),
                'password' => Hash::make('password123'),
                'role'     => 'siswa',
            ]);

            Siswa::create([
                'user_id'       => $user->id,
                'nis'           => 'EL' . rand(100, 999),
                'nama_lengkap'  => 'Siswa Eloquent ORM',
                'kelas'         => '1C',
                'jenis_kelamin' => 'Laki-laki',
                'tempat_lahir'  => 'Medan',
                'tanggal_lahir' => '2012-03-03',
                'alamat'        => 'Alamat Eloquent',
                'no_hp'         => '081200000000',
            ]);

            DB::commit();

            return response()->json([
                'success' => true,
                'method'  => 'Eloquent ORM',
                'message' => 'Insert data menggunakan Eloquent ORM berhasil'
            ]);
        } catch (\Exception $e) {
            DB::rollBack();

            return response()->json([
                'success' => false,
                'error'   => $e->getMessage()
            ], 500);
        }
    }
}
```

Output:

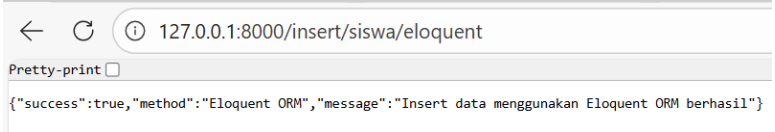
raw SQL Queries



Query Builder



Eloquent ORM



Penjelasan:
Kode yang dibuat berfungsi untuk melakukan proses penyimpanan (insert) data siswa ke dalam database dengan tiga pendekatan yang berbeda, yaitu Raw SQL Queries, Query Builder, dan Eloquent ORM. Ketiga fungsi tersebut berada di dalam SiswaController dan masing-masing menjalankan proses penyimpanan data ke dua tabel yang saling berelasi, yaitu tabel users sebagai parent dan tabel siswas sebagai child yang terhubung melalui user_id.

Pada metode Raw SQL Queries, proses insert dilakukan dengan menuliskan perintah SQL secara manual menggunakan DB::insert(). Setelah data user disimpan, sistem mengambil id terakhir menggunakan lastInsertId() untuk kemudian digunakan sebagai foreign key saat menyimpan data siswa. Seluruh proses dibungkus dalam transaksi database (DB::beginTransaction) untuk menjaga konsistensi data apabila terjadi kesalahan.

Pada metode Query Builder, Laravel menyediakan cara yang lebih aman dan terstruktur tanpa menuliskan SQL secara langsung. Proses insert dilakukan menggunakan DB::table()->insertGetId() untuk mendapatkan id user, lalu dilanjutkan dengan insert data siswa. Pendekatan ini mengurangi risiko kesalahan sintaks SQL serta lebih mudah dibaca dibandingkan Raw SQL.

Sedangkan pada metode Eloquent ORM, proses insert dilakukan menggunakan model User dan Siswa secara langsung melalui method create(). Laravel secara otomatis menangani relasi antar tabel dan proses penyimpanan data. Pendekatan ini menghasilkan kode yang lebih singkat, rapi, dan mudah dipelihara, sehingga sangat cocok digunakan pada pengembangan aplikasi berskala besar.

BAB III

PENUTUP

A. Kesimpulan

Berdasarkan hasil praktikum dan tugas mandiri pada Modul 12, dapat disimpulkan bahwa penggunaan migration dan model pada framework Laravel sangat membantu dalam pengelolaan struktur database dan manipulasi data secara terstruktur, aman, dan terkontrol. Praktikum ini memberikan pemahaman mengenai pembuatan dan modifikasi tabel database menggunakan migration, serta penerapan tiga pendekatan pengolahan data yaitu Raw SQL, Query Builder, dan Eloquent ORM. Kendala yang dihadapi terutama berkaitan dengan kesalahan penulisan struktur migration dan relasi antar tabel, namun dapat diatasi dengan memahami fungsi method up() dan down(), serta melakukan rollback dan perbaikan kode. Secara keseluruhan, praktikum ini meningkatkan pemahaman dalam perancangan database dan implementasi pengolahan data yang efektif pada aplikasi Laravel.