LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL 12

LARAVEL 2

Oleh:

Tiurma Grace Angelina 2311104042

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

A. Dasar Teori

Migration adalah fitur Laravel yang memungkinkan kita mengelola struktur database menggunakan kode PHP. Dengan migration, kita dapat membuat, mengubah, dan menghapus tabel serta kolom database tanpa harus menulis query SQL secara manual di phpMyAdmin. Migration bekerja seperti version control untuk database, sehingga setiap perubahan struktur database dapat dilacak dan dikembalikan (rollback) jika diperlukan.

Keuntungan menggunakan migration antara lain: perubahan database dapat dilakukan tanpa menghapus data yang sudah ada, memudahkan kolaborasi tim karena perubahan database dapat dibagikan melalui file migration, dan mempercepat proses deployment ke server produksi karena cukup menjalankan perintah migrate.

Model pada Laravel

Model adalah representasi dari tabel database dalam bentuk class PHP. Model memudahkan kita untuk melakukan operasi CRUD (Create, Read, Update, Delete) pada database tanpa harus menulis query SQL yang panjang. Dengan menggunakan model, kode program menjadi lebih bersih, mudah dipahami, dan lebih aman dari serangan seperti SQL Injection.

Laravel menyediakan tiga cara untuk berinteraksi dengan database:

1. **Raw SQL Queries (DB Facade)**: Cara paling dasar dengan menulis query SQL secara langsung menggunakan DB::insert(), DB::select(), dan method lainnya. Metode ini cocok untuk query yang kompleks dan custom.
2. **Query Builder**: Interface khusus Laravel untuk mengakses database menggunakan method PHP, bukan menulis SQL langsung. Contohnya DB::table('nama_tabel')->insert(). Metode ini lebih aman dari SQL injection dan sintaksnya lebih mudah dibaca.
3. **Eloquent ORM (Object-Relational Mapping)**: Cara paling modern dan mudah, dimana setiap baris tabel dianggap sebagai object. Menggunakan model untuk operasi database seperti Model::create(). Metode ini paling direkomendasikan untuk operasi CRUD standar karena paling simple dan otomatis menghandle timestamps.

B. Tujuan

Tujuan dari praktikum ini adalah:

1. Memahami konsep dan fungsi migration pada Laravel untuk mengelola struktur database
2. Mampu membuat dan menjalankan file migration untuk membuat tabel baru

3. Mampu memodifikasi struktur tabel menggunakan migration (alter table)
4. Memahami konsep model sebagai representasi tabel database
5. Mampu melakukan operasi insert data menggunakan tiga metode: Raw SQL Queries, Query Builder, dan Eloquent ORM
6. Memahami perbedaan, kelebihan, dan kekurangan dari ketiga metode insert data
7. Mampu mengimplementasikan migration dan model untuk kebutuhan tugas besar

BAB II

HASIL

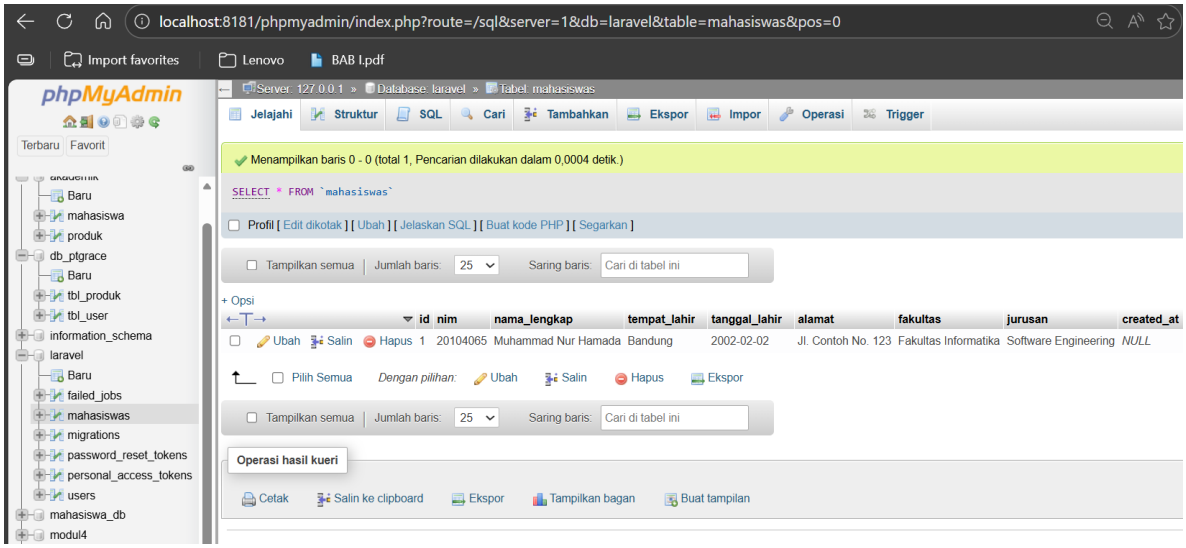
A. GUIDED (Praktikum Terbimbing)

Langkah 1: Konfigurasi Database

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=
```

Penjelasan: File .env berisi konfigurasi lokal untuk koneksi database.

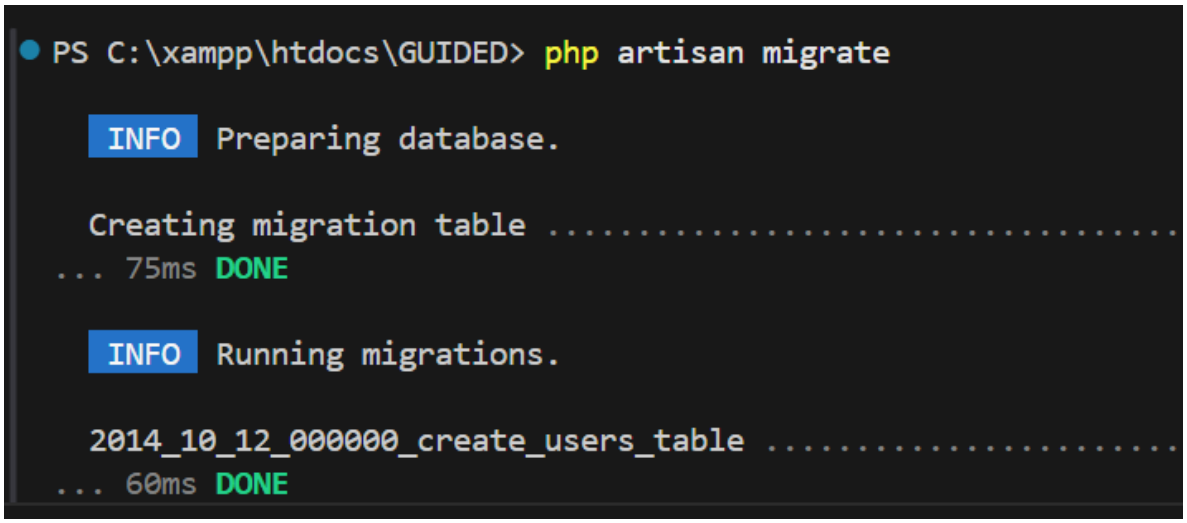
Langkah 2: Buat Database di phpMyAdmin



Langkah 3: Menjalankan Migration Bawaan

php artisan migrate

Penjelasan: Perintah ini akan menjalankan semua file migration yang ada di folder `database/migrations` dan membuat tabel-tabel bawaan Laravel seperti `users`, `password_resets`, dll.



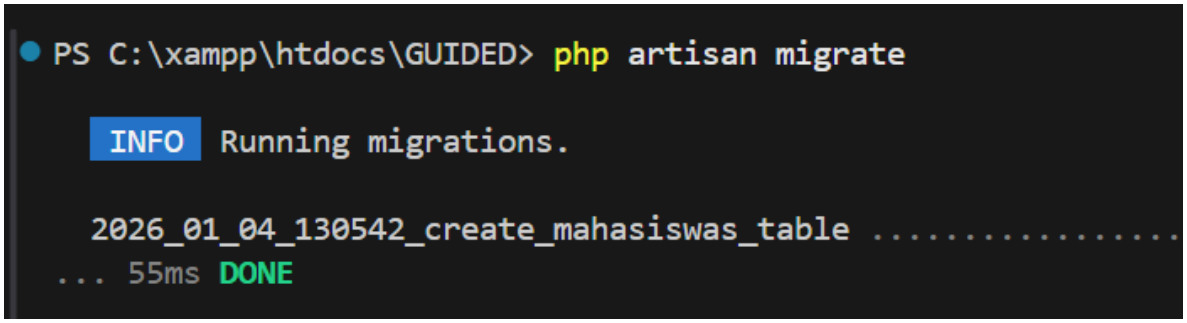
Langkah 4: Buat File Migration

```
<?php

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

class CreateMahasiswasTable extends Migration
{
    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');
    }
}
```

Penjelasan:

- up(): Berisi perintah untuk membuat tabel beserta kolom-kolomnya
- down(): Berisi perintah untuk menghapus tabel (rollback)
- $table->char('nim', 8): Membuat kolom nim dengan tipe CHAR(8)
- $table->timestamps(): Otomatis menambahkan kolom created_at dan updated_at

Langkah 6: Jalankan Migration



Langkah 7: Edit File Migration (Tambah Constraint)

```
<?php

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

class CreateMahasiswasTable extends Migration
{
    public function up()
    {
        Schema::create('mahasiswas', function (Blueprint $table) {
            $table->id();
            $table->char('nim', 8)->unique();
            $table->string('nama');
            $table->string('tempat_lahir');
            $table->date('tanggal_lahir');
            $table->string('fakultas');
            $table->string('jurusan');
            $table->decimal('ipk', 3, 2)->default(1.00);
            $table->timestamps();
        });
    }
```

Penjelasan:

- unique(): Membuat kolom nim harus unik (tidak boleh duplikat)
- default(1.00): Memberikan nilai default 1.00 untuk kolom ipk

Langkah 8: Rollback dan Migrate Ulang

```
PS C:\xampp\htdocs\GUIDED> php artisan migrate:rollback --step=1

  INFO  Rolling back migrations.

  2026_01_04_130542_create_mahasiswas_table ..................................
 ... 19ms DONE

PS C:\xampp\htdocs\GUIDED> php artisan migrate

  INFO  Running migrations.

  2026_01_04_130542_create_mahasiswas_table ..................................
 ... 51ms DONE
```

Langkah 9: Install Doctrine DBAL

```
PS C:\xampp\htdocs\GUIDED> composer require doctrine/dbal
  nesbot/carbon ..................................................
........ DONE
  nunomaduro/collision ...........................................
........ DONE
  nunomaduro/termwind ............................................
........ DONE
  spatie/laravel-ignition ........................................
........ DONE

82 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
> @php artisan vendor:publish --tag=laravel-assets --ansi --force

  INFO  No publishable resources for tag [laravel-assets].

No security vulnerability advisories found.
Using version ^3.10 for doctrine/dbal
```

Penjelasan: Library ini diperlukan untuk memodifikasi struktur tabel yang sudah ada.

Langkah 10: Buat Migration Alter

```
PS C:\xampp\htdocs\GUIDED> php artisan make:migration alter_mahasiswas_table --table=mahasiswas
  INFO  Migration [C:\xampp\htdocs\GUIDED\database\migrations/2026_01_04_130911_alter_mahasiswas_table.php
] created successfully.
```

Langkah 11: Edit File Alter Migration

```php
<?php

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

class AlterMahasiswasTable extends Migration
{
    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);
        });
    }
}
```

Penjelasan:

- renameColumn(): Mengubah nama kolom
- text('alamat')->after(): Menambah kolom baru setelah kolom tertentu
- dropColumn(): Menghapus kolom

Langkah 12: Jalankan Migration

```
PS C:\xampp\htdocs\GUIDED> php artisan migrate

  INFO  Running migrations.

  2026_01_04_130911_alter_mahasiswas_table ..................
 .. 176ms DONE
```

Langkah 13: Buat Controller

```php
<?php

namespace App\Http\Controllers;

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

class MahasiswaController extends Controller
{
    public function index()
    {
        return "Index untuk mahasiswa";
    }
}
```

Penjelasan:

- use Illuminate\Support\Facades\DB; → Import DB Facade untuk query database
- Method index() → Method dasar untuk testing

Langkah 15: Tambah Method insertSql

MahasiswaController.php

```php
public function insertSql()
{
    $result1 = DB::insert("
        INSERT INTO mahasiswas
        (nim, nama_lengkap, tempat_lahir, tanggal_lahir, alamat, fakultas, jurusan)
        VALUES
        ('20104065',
        'Muhammad Nur Hamada',
        'Bandung',
        '2002-02-02',
        'Jl. Contoh No. 123',
        'Fakultas Informatika',
        'Software Engineering')"
    );

    dump($result1);
}
```

Penjelasan:

- DB::insert() → Method untuk menjalankan query INSERT mentah
- dump() → Menampilkan hasil (true/false)
- Query SQL ditulis langsung seperti di phpMyAdmin

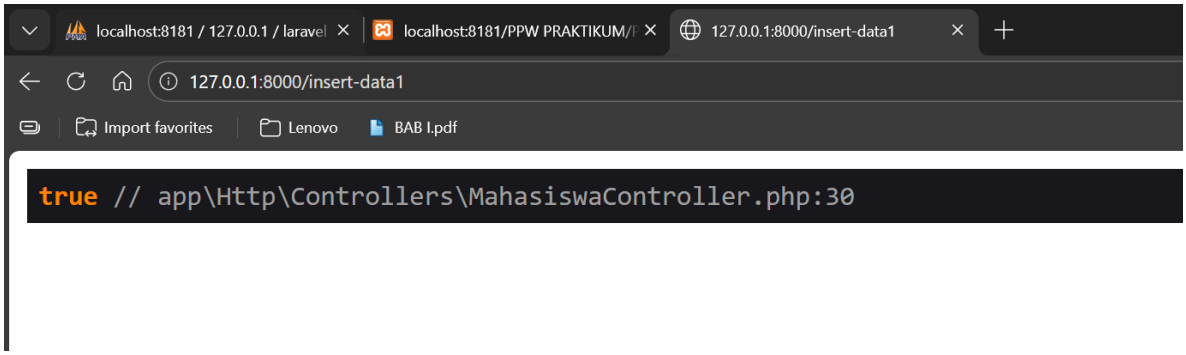
Langkah 16: Buat Route

```php
<?php

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

Route::get('/insert-data1', [MahasiswaController::class, 'insertSql']);
```

Penjelasan:

- Route GET untuk mengakses method insertSql
- Format: Route::get('url', [Controller::class, 'method'])

Langkah 17: Testing



Langkah 18: Tambah Method insertQB

MahasiswaController.php:

```php
public function insertQB()
{
    $result2 = DB::table('mahasiswas')->insert([
        'nim' => '20104070',
        'nama_lengkap' => 'Aulia Putri Ramadhani',
        'tempat_lahir' => 'Surabaya',
        'tanggal_lahir' => '2003-05-14',
        'alamat' => 'Jl. Mawar No. 10',
        'fakultas' => 'Fakultas Teknik',
        'jurusan' => 'Teknik Informatika',
    ]);

    dump($result2);
}
```

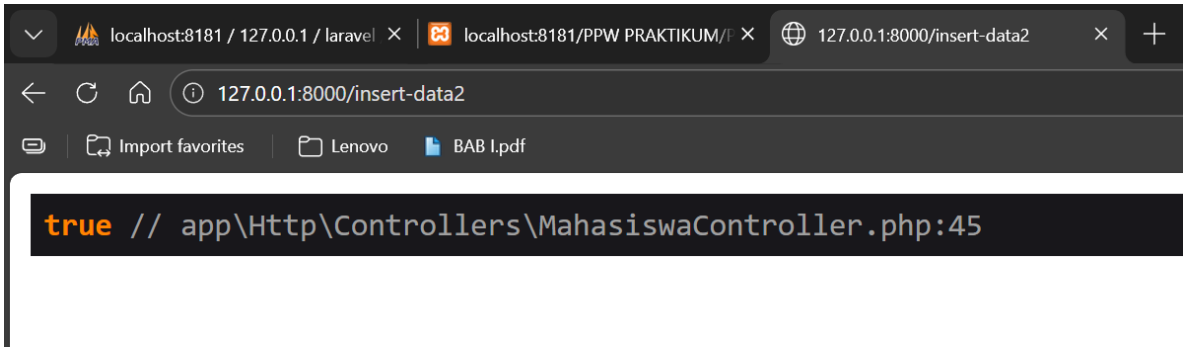
Penjelasan:

- DB::table('mahasiswas') → Menargetkan tabel mahasiswas
- ->insert([]) → Method insert dengan format array key-value
- Lebih rapi dan aman dari SQL injection dibanding raw query

Langkah 19: Buat Route

```
Route::get('/insert-data2', [MahasiswaController::class, 'insertQB']);
```

Langkah 20: Testing



Langkah 21: Buat Model
app/Models/Mahasiswa.php:

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Mahasiswa extends Model
{
    protected $table = 'mahasiswas';

    protected $fillable = [
        'nim',
        'nama_lengkap',
        'tempat_lahir',
        'tanggal_lahir',
        'alamat',
        'fakultas',
        'jurusan',
    ];
}
```

Penjelasan:

- $table → Mendefinisikan nama tabel yang digunakan
- $fillable → Kolom-kolom yang boleh diisi secara mass assignment (keamanan)
- Model menjadi representasi object dari tabel database

Langkah 23: Import Model di Controller

Edit MahasiswaController.php

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

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use App\Models\Mahasiswa;
```

Langkah 24: Tambah Method insertEloquent

MahasiswaController.php:

```
public function insertEloquent()
{
    $mhs = Mahasiswa::create([
        'nim' => '20104080',
        'nama_lengkap' => 'Rizky Ananda',
        'tempat_lahir' => 'Malang',
        'tanggal_lahir' => '2002-09-12',
        'alamat' => 'Jl. Kenanga No. 7',
        'fakultas' => 'Fakultas Informatika',
        'jurusan' => 'Software Engineering',
    ]);

    dump($mhs);
}
```

Penjelasan:

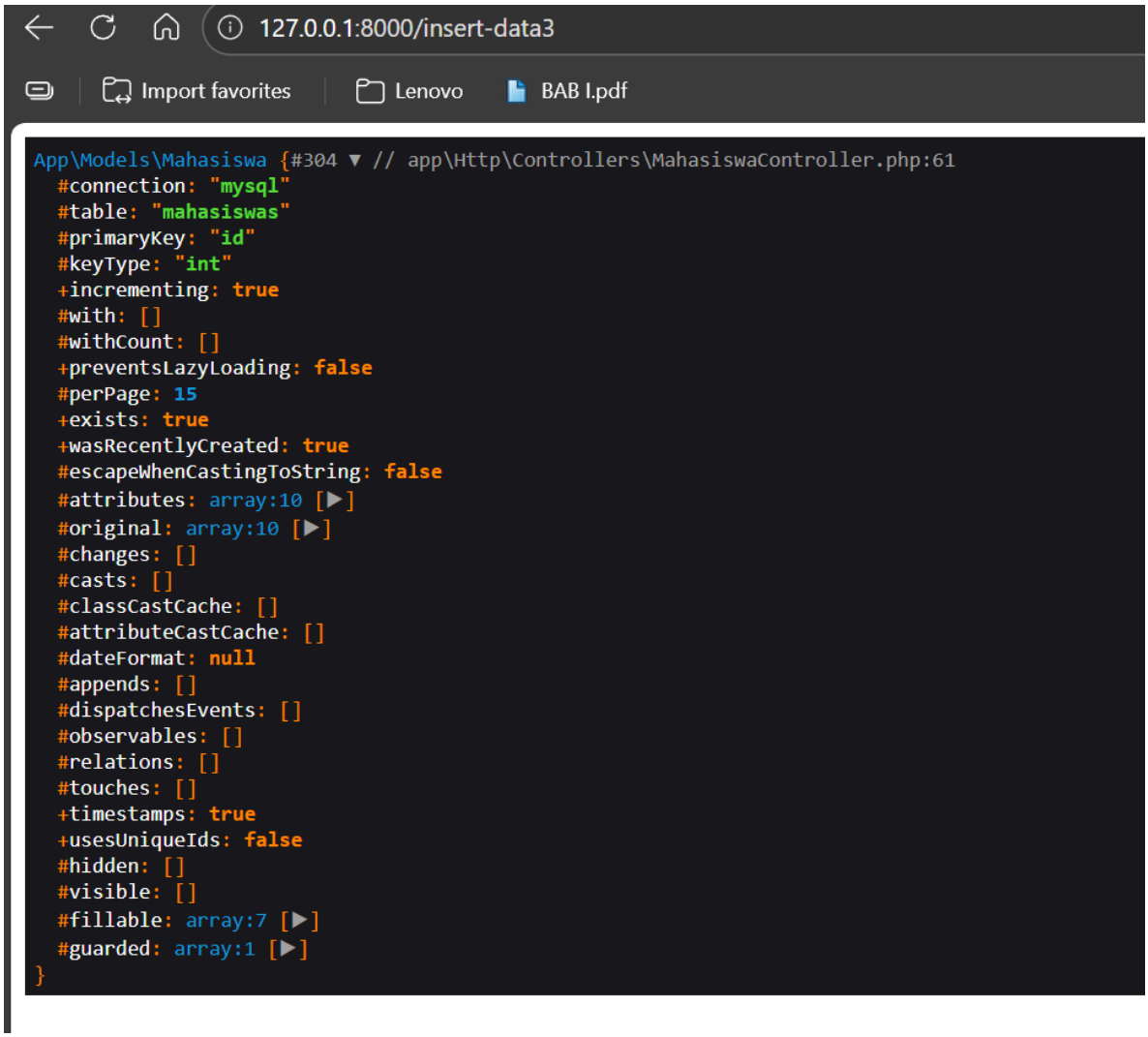
- Mahasiswa::create([]) → Method Eloquent untuk insert data
- Langsung menggunakan nama Model, bukan DB::table
- Hasil yang di-dump adalah object lengkap dengan semua atribut

Langkah 25: Buat Route

routes/web.php:

```
Route::get('/insert-data3', [MahasiswaController::class, 'insertEloquent']);
```

Langkah 26: Testing



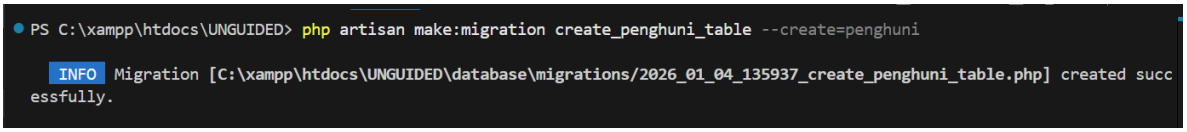
B. UNGUIDED (Tugas Mandiri)

1. **Soal 1:**

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

   Jawaban:

   Langkah 1: Membuat File Migration



   Penjelasan: Perintah ini akan membuat file migration baru di folder database/migrations dengan timestamp otomatis. Tabel penghuni dipilih karena

merupakan salah satu tabel utama dalam sistem Panti Wredha Budi Dharma Kasih yang digunakan untuk menyimpan data penghuni panti.

Langkah 2: Edit File Migration

```
<?php

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

class CreatePenghuniTable extends Migration
{
    public function up()
    {
        Schema::create('penghuni', function (Blueprint $table) {
            $table->id();
            $table->string('nama');
            $table->enum('jenis_kelamin', ['Laki-laki', 'Perempuan']);
            $table->date('tanggal_lahir');
            $table->text('alamat_asal');
            $table->date('tanggal_masuk');
            $table->enum('status_penghuni', ['Aktif', 'Tidak Aktif'])-
>default('Aktif');
            $table->timestamps();
        });
    }

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

Penjelasan:

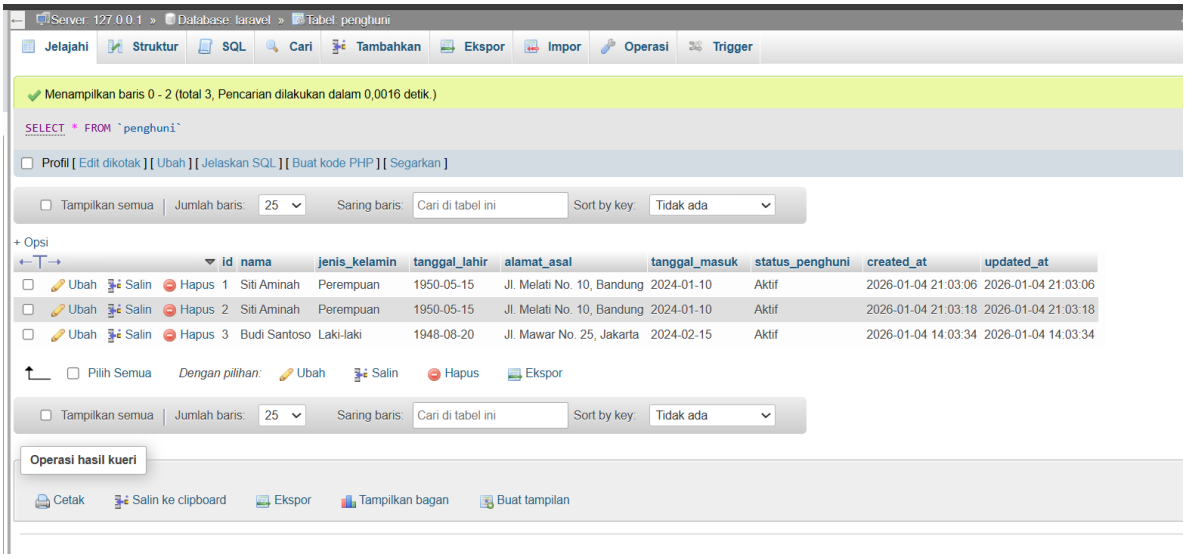
- id() → Primary key dengan auto increment
- string('nama') → Kolom untuk nama penghuni dengan tipe VARCHAR
- enum('jenis_kelamin') → Kolom dengan pilihan terbatas (Laki-laki/Perempuan)
- date('tanggal_lahir') → Kolom untuk tanggal lahir
- text('alamat_asal') → Kolom untuk alamat dengan tipe TEXT
- date('tanggal_masuk') → Kolom untuk tanggal masuk ke panti
- enum('status_penghuni')->default('Aktif') → Status dengan nilai default 'Aktif'

- timestamps() → Menambahkan kolom created_at dan updated_at otomatis
- Method down() berisi perintah untuk menghapus tabel saat rollback

Langkah 3: Jalankan Migration

```
PS C:\xampp\htdocs\UNGUIDED> php artisan migrate

  INFO  Running migrations.

  2026_01_04_135937_create_penghuni_table ..... 87ms DONE
```

Penjelasan: Migration berhasil dijalankan dan tabel penghuni telah dibuat di database dengan struktur kolom sesuai definisi di file migration.



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 ORMJawaban:**

Jawab:

Langkah 1: Membuat Model Penghuni

```
PS C:\xampp\htdocs\UNGUIDED> php artisan make:model Penghuni

  INFO  Model [C:\xampp\htdocs\UNGUIDED\app\Models\Penghuni.php] created successfully.
```

Penjelasan: Model akan menjadi representasi dari tabel penghuni dan memudahkan operasi database menggunakan Eloquent ORM.

Edit file app/Models/Penghuni.php:

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Penghuni extends Model
{
    protected $table = 'penghuni';

    protected $fillable = [
        'nama',
        'jenis_kelamin',
        'tanggal_lahir',
        'alamat_asal',
        'tanggal_masuk',
        'status_penghuni',
    ];
}
```

Penjelasan:

- $table → Mendefinisikan nama tabel yang digunakan (penghuni)
- $fillable → Array berisi nama kolom yang diizinkan untuk mass assignment (keamanan dari serangan mass assignment vulnerability)

Langkah 2: Membuat Controller

```
PS C:\xampp\htdocs\UNGUIDED> php artisan make:controller PenghuniController

  INFO  Controller [C:\xampp\htdocs\UNGUIDED\app\Http\Controllers\PenghuniController.php] created successfully.
```

Edit file app/Http/Controllers/PenghuniController.php:

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use App\Models\Penghuni;

class PenghuniController extends Controller
{
    // 1. Insert menggunakan Raw SQL Queries
    public function insertSql()
    {
        $result = DB::insert("
```

```
        INSERT INTO penghuni
        (nama, jenis_kelamin, tanggal_lahir, alamat_asal, tanggal_masuk,
status_penghuni, created_at, updated_at)
        VALUES
        ('Siti Aminah',
        'Perempuan',
        '1950-05-15',
        'Jl. Melati No. 10, Bandung',
        '2024-01-10',
        'Aktif',
        NOW(),
        NOW())"
    );

    dump($result);
  }

  // 2. Insert menggunakan Query Builder
  public function insertQB()
  {
    $result = DB::table('penghuni')->insert([
      'nama' => 'Budi Santoso',
      'jenis_kelamin' => 'Laki-laki',
      'tanggal_lahir' => '1948-08-20',
      'alamat_asal' => 'Jl. Mawar No. 25, Jakarta',
      'tanggal_masuk' => '2024-02-15',
      'status_penghuni' => 'Aktif',
      'created_at' => now(),
      'updated_at' => now(),
    ]);

    dump($result);
  }

  // 3. Insert menggunakan Eloquent ORM
  public function insertEloquent()
  {
    $penghuni = Penghuni::create([
      'nama' => 'Joko Widodo',
      'jenis_kelamin' => 'Laki-laki',
      'tanggal_lahir' => '1952-03-12',
      'alamat_asal' => 'Jl. Kenanga No. 5, Surabaya',
      'tanggal_masuk' => '2024-03-01',
      'status_penghuni' => 'Aktif',
```

```
        ]);

        dump($penghuni);
    }
}
```

Penjelasan:

a. Fungsi insertSql() - Raw SQL Queries:

- Menggunakan DB::insert() untuk menjalankan query SQL mentah
- Query ditulis langsung seperti SQL biasa
- Mengembalikan nilai true jika berhasil, false jika gagal
- Kelebihan: Fleksibel untuk query kompleks
- Kekurangan: Rentan SQL injection jika tidak hati-hati

b. Fungsi insertQB() - Query Builder:

- Menggunakan DB::table('penghuni')->insert() dengan array key-value
- Lebih aman dari SQL injection karena Laravel otomatis melakukan escaping
- Sintaks lebih bersih dan mudah dibaca dibanding raw SQL
- Mengembalikan nilai true jika berhasik

c. Fungsi insertEloquent() - Eloquent ORM:

- Menggunakan Model::create() untuk insert data
- Paling mudah dan elegant, data langsung dalam bentuk object
- Otomatis menghandle timestamps (created_at, updated_at)
- Mengembalikan object lengkap dengan semua atribut
- Recommended untuk operasi CRUD standar

Langkah 3: Menambahkan Route

Edit file routes/web.php

```
<?php

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

// Route Unguided
Route::get('/unguided/insert-sql', [PenghuniController::class, 'insertSql']);
Route::get('/unguided/insert-qb', [PenghuniController::class, 'insertQB']);
Route::get('/unguided/insert-eloquent', [PenghuniController::class,
'insertEloquent']);
```

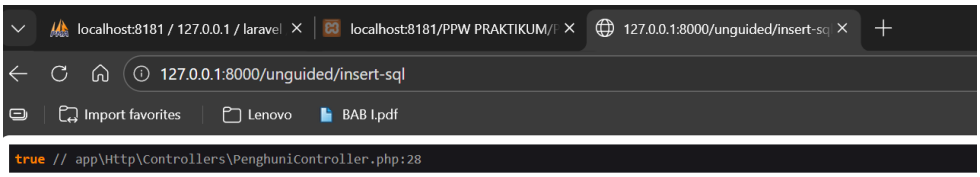
Penjelasan:

- Route GET digunakan untuk mengakses ketiga fungsi insert melalui browser
- Format: Route::get('url', [NamaController::class, 'namaMethod'])

- URL dibuat berbeda untuk memisahkan testing ketiga metode

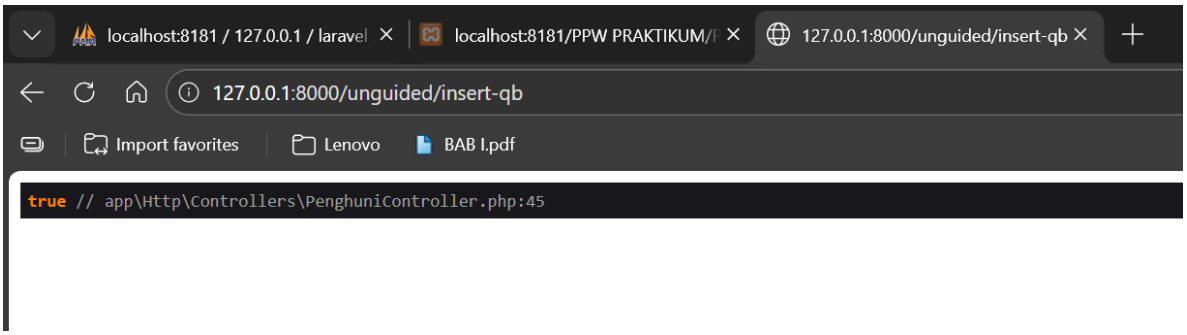
Langkah 4: Testing Fungsi Insert

a. Testing Raw SQL Queries



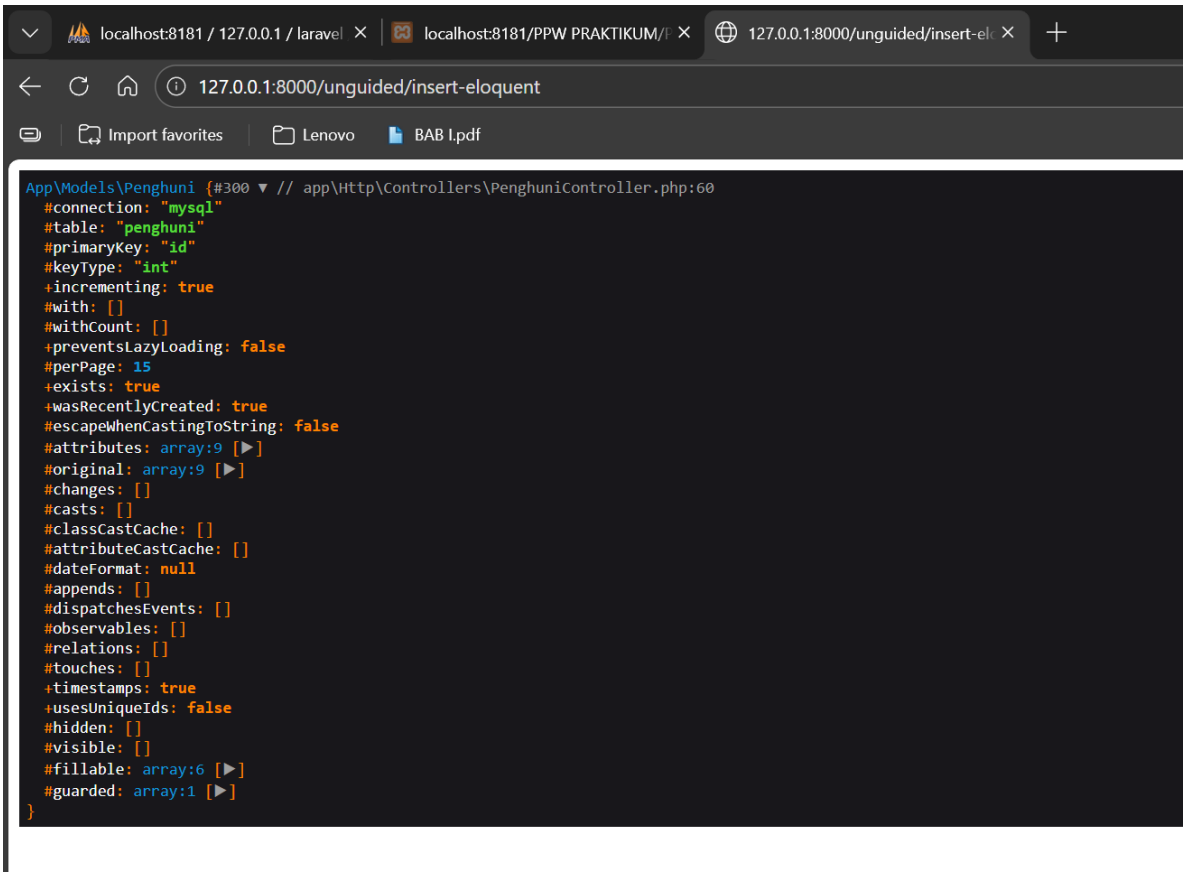
Penjelasan: Hasil true menunjukkan bahwa query INSERT dengan Raw SQL berhasil dijalankan dan data "Siti Aminah" telah tersimpan ke database.

b. Testing Query Builder



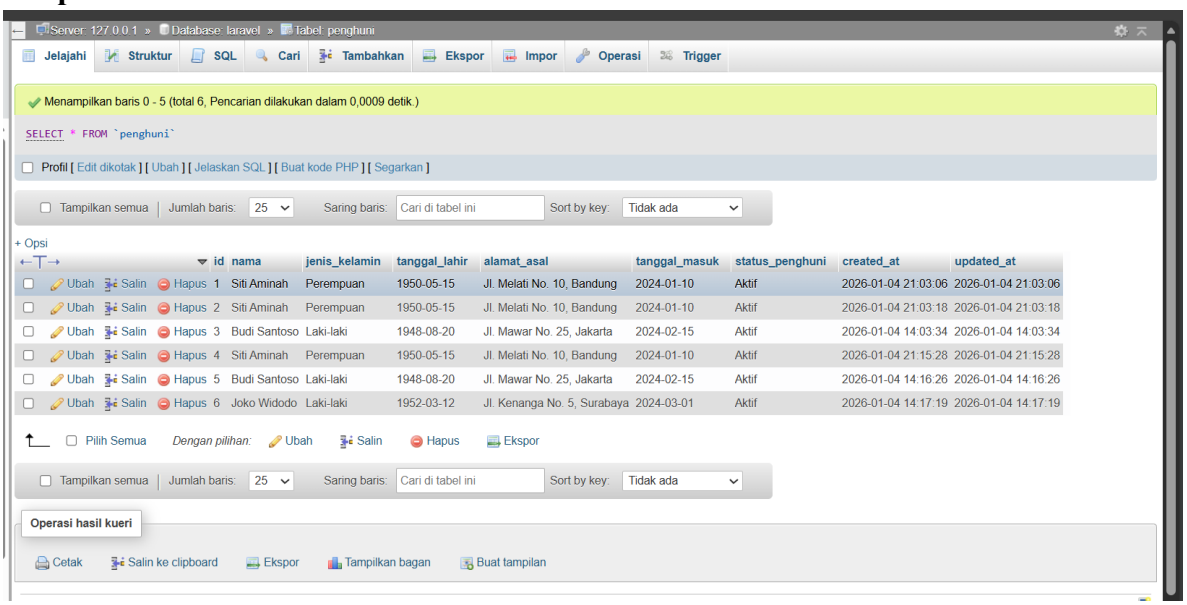
Penjelasan: Hasil true menunjukkan bahwa insert dengan Query Builder berhasil dan data "Budi Santoso" telah tersimpan ke database.

c. Testing Eloquent ORM



Penjelasan: Eloquent ORM mengembalikan object lengkap dari data yang baru diinsert, termasuk id, nama, jenis_kelamin, dan timestamps. Data "Joko Widodo" telah tersimpan ke database.

Output:



Ketiga data dari masing-masing metode insert (Raw SQL, Query Builder, dan Eloquent ORM) berhasil tersimpan di database dengan lengkap. Terlihat kolom created_at dan updated_at juga terisi otomatis.

BAB III

PENUTUP

A. Kesimpulan

Dari praktikum Modul 12 tentang Migration dan Model pada Laravel, dapat disimpulkan beberapa hal penting:

1. **Migration** sangat membantu dalam mengelola struktur database karena semua perubahan dapat dilacak melalui file migration dan dapat di-rollback jika terjadi kesalahan. Migration membuat pengembangan aplikasi menjadi lebih terstruktur dan memudahkan kolaborasi tim.
2. **Model** mempermudah interaksi dengan database karena kita tidak perlu menulis query SQL yang panjang. Dengan model, operasi CRUD menjadi lebih simple dan kode lebih mudah dipahami.
3. Laravel menyediakan **tiga metode** untuk insert data ke database, yaitu:
    - **Raw SQL Queries**: Fleksibel untuk query kompleks tapi rentan SQL injection
    - **Query Builder**: Lebih aman dan sintaks lebih bersih dibanding raw SQL
    - **Eloquent ORM**: Paling mudah dan direkomendasikan untuk CRUD standar
4. Eloquent ORM adalah metode yang paling efisien untuk operasi database standar karena otomatis menghandle timestamps, lebih aman, dan mengembalikan data dalam bentuk object yang mudah dimanipulasi.
5. Pemilihan metode insert data tergantung kebutuhan: gunakan Raw SQL untuk query sangat kompleks, Query Builder untuk query standar dengan kondisi khusus, dan Eloquent ORM untuk operasi CRUD sehari-hari.

Kendala yang dihadapi:

- Pada awalnya terjadi error "can't reach this page" karena belum menjalankan php artisan serve dan MySQL di XAMPP belum di-start
- Setelah memastikan Apache, MySQL di XAMPP running dan menjalankan Laravel development server dengan php artisan serve, semua fungsi berhasil dijalankan dengan baik

Praktikum ini memberikan pemahaman yang baik tentang pengelolaan database di Laravel menggunakan migration dan model, yang akan sangat berguna untuk pengembangan aplikasi web yang lebih kompleks kedepannya.