

LAPORAN PRAKTIKUM
PERANCANGAN DAN PEMROGRAMAN WEB

MODUL 13
(MIGRATION)



Oleh:

Farhan Kurniawan 2311104073

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK
DIREKTORAT KAMPUS PURWOKERTO
UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

A. Dasar Teori

Migration pada framework Laravel merupakan fitur yang digunakan untuk mengelola struktur basis data secara terprogram melalui kode. Dengan migration, pengembang dapat membuat, mengubah, dan menghapus tabel maupun kolom pada database tanpa harus menuliskan perintah SQL secara langsung. Setiap perubahan struktur database disimpan dalam bentuk file migration sehingga memudahkan proses pengelolaan database secara sistematis, konsisten, dan terdokumentasi dengan baik.

Selain itu, migration berperan sebagai *version control* untuk database karena setiap perubahan dicatat berdasarkan urutan waktu (timestamp). Hal ini memungkinkan pengembang melakukan rollback atau mengembalikan struktur database ke kondisi sebelumnya apabila terjadi kesalahan. Dengan kemampuan tersebut, migration sangat membantu dalam pengembangan aplikasi yang dinamis, terutama ketika terjadi perubahan kebutuhan sistem, tanpa harus menghapus data yang sudah ada di dalam database.

B. Tujuan

Tujuan dari praktikum ini adalah untuk memahami konsep dan fungsi migration pada framework Laravel serta mampu menerapkannya dalam pengelolaan struktur basis data aplikasi. Melalui praktikum ini, mahasiswa diharapkan dapat membuat, memodifikasi, dan mengelola tabel database secara terstruktur menggunakan migration, sehingga proses pengembangan aplikasi menjadi lebih efisien, terkontrol, dan mudah dikembangkan sesuai dengan kebutuhan sistem.

BAB II

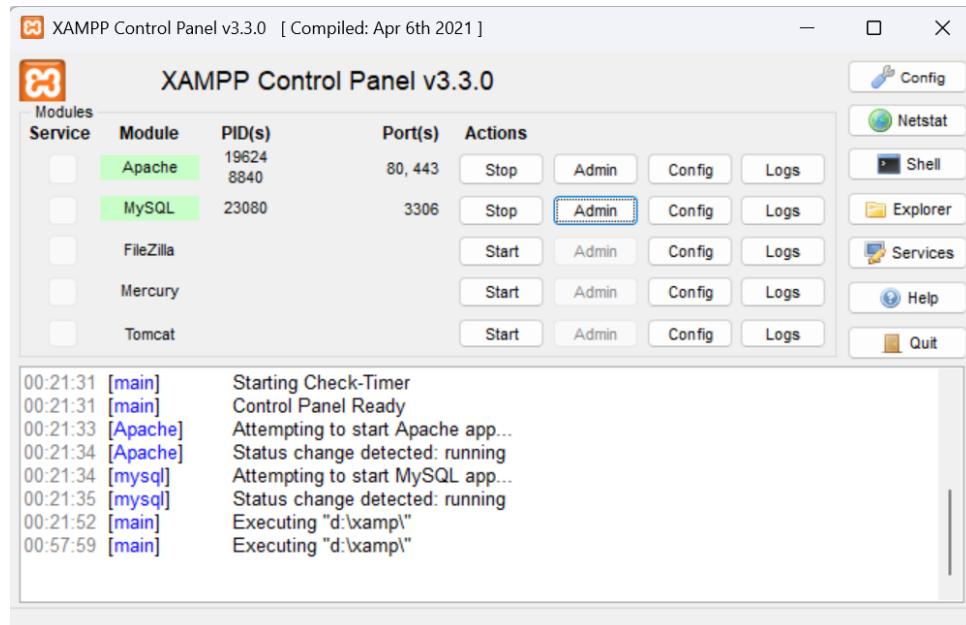
HASIL

A. GUIDED

1. Migration

a. Konfigurasi Database

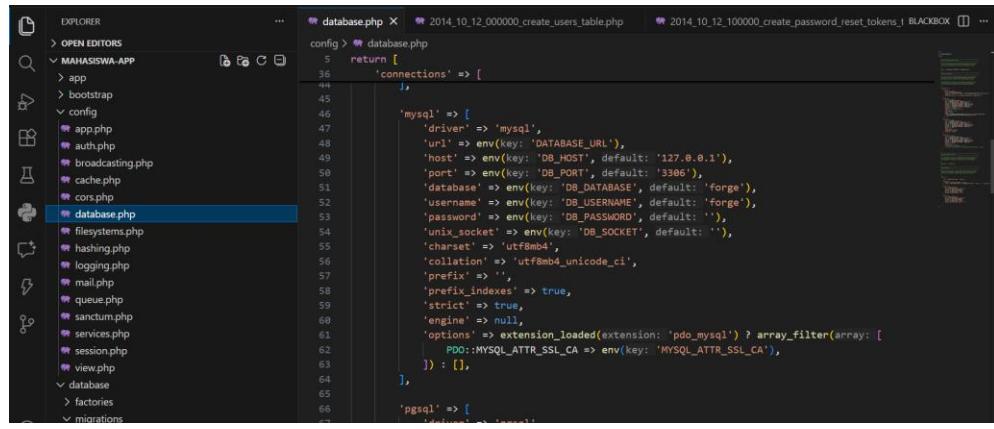
Sebelum menjalankan migration, layanan Apache dan MySQL harus diaktifkan melalui XAMPP.



Selanjutnya, Laravel memerlukan pengaturan koneksi database yang dilakukan pada dua berkas konfigurasi, yaitu file .env sebagai konfigurasi lokal dan file config/database.php sebagai konfigurasi global.

A screenshot of a file explorer interface. On the left, there is a tree view of a project structure: MAHASISWA-APP, app, bootstrap, config, database, public, resources, routes, storage, tests, vendor, .editorconfig, .env (which is selected and highlighted in blue), .env.example, and .gitattributes. On the right, the contents of the .env file are displayed in a code editor:

```
4 APP_DEBUG=true
5 APP_URL=http://localhost
6
7 LOG_CHANNEL=stack
8 LOG_DEPRECATIONS_CHANNEL=null
9 LOG_LEVEL=debug
10
11 DB_CONNECTION=mysql
12 DB_HOST=127.0.0.1
13 DB_PORT=3306
14 DB_DATABASE=laravel
15 DB_USERNAME=root
16 DB_PASSWORD=
17
18 BROADCAST_DRIVER=log
19 CACHE_DRIVER=file
20 FILESYSTEM_DISK=local
21 QUEUE_CONNECTION=sync
```

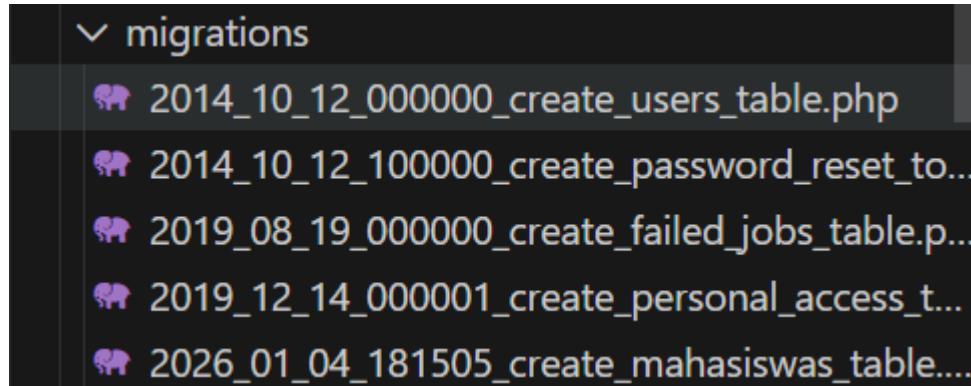


```
database.php
config > database.php
  5  return [
  6    'connections' => [
  7      'mysql' => [
  8        'driver' => 'mysql',
  9        'url' => env(key: 'DATABASE_URL'),
 10        'host' => env(key: 'DB_HOST', default: '127.0.0.1'),
 11        'port' => env(key: 'DB_PORT', default: '3306'),
 12        'database' => env(key: 'DB_DATABASE', default: 'forge'),
 13        'username' => env(key: 'DB_USERNAME', default: 'forge'),
 14        'password' => env(key: 'DB_PASSWORD', default: ''),
 15        'unix_socket' => env(key: 'DB_SOCKET', default: ''),
 16        'charset' => 'utf8mb4',
 17        'collation' => 'utf8mb4_unicode_ci',
 18        'prefix' => '',
 19        'prefix_indexes' => true,
 20        'strict' => true,
 21        'engine' => null,
 22        'options' => extension_loaded(extension: 'pdo_mysql') ? array_filter(array: [
 23          PDO::MYSQL_ATTR_SSL_CA => env(key: 'MYSQL_ATTR_SSL_CA'),
 24        ]) : []
 25      ],
 26      'pgsql' => [
 27        ...
 28      ]
 29    ]
 30  ]
 31
```

Pengaturan pada file .env digunakan untuk menentukan nama database, username, dan password yang akan digunakan oleh aplikasi. Sementara itu, file database.php berfungsi untuk memastikan pengaturan database sesuai dengan konfigurasi yang ditetapkan secara global pada framework Laravel.

b. File Migration Bawaan Laravel

Pada beberapa versi Laravel, telah tersedia file migration bawaan yang terletak pada direktori database/migrations.



Setiap file migration digunakan untuk membentuk satu tabel pada database. Oleh karena itu, beberapa file migration bawaan akan menghasilkan beberapa tabel secara otomatis. Nama file migration diawali dengan timestamp yang menunjukkan waktu pembuatan file tersebut. Urutan timestamp ini menentukan urutan eksekusi migration ketika perintah dijalankan. Untuk menjalankan migration bawaan Laravel, digunakan perintah:

```

PS D:\xampp\htdocs\PPW\Pertemuan 13\Guided\mahasiswa-app> php artisan migrate
>>

[INFO] Preparing database.

Creating migration table ..... 14ms DON
E

[INFO] Running migrations.

2014_10_12_000000_create_users_table ..... 29ms DON
E
2014_10_12_100000_create_password_reset_tokens_table ..... 10ms DON
E

Commit Message Ln 1, Col 1 Spaces: 4 UTF-8 LF { } PHP Go Live BLACKBOXAI: Open Chat 8.1 Prettier

```

Setelah perintah dijalankan, hasil pembuatan tabel dapat diperiksa melalui phpMyAdmin.

Table	Action	Rows	Type	Collation	Size	Overhead
failed_jobs	<input type="checkbox"/> Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
migrations	<input type="checkbox"/> Browse Structure Search Insert Empty Drop	4	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
password_reset_tokens	<input type="checkbox"/> Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	16.0 KiB	-
personal_access_tokens	<input type="checkbox"/> Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	48.0 KiB	-
users	<input type="checkbox"/> Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 KiB	-
5 tables	Sum	4	InnoDB	utf8mb4_general_ci	144.0 KiB	0 B

c. Migration Rollback

Laravel menyediakan fitur rollback untuk membatalkan perubahan struktur database yang telah dijalankan. Migration dijalankan berdasarkan urutan timestamp, sehingga rollback dapat digunakan untuk kembali ke kondisi database sebelumnya. Untuk melihat status migration yang telah dijalankan, digunakan perintah:

```

PS D:\xampp\htdocs\PPW\Pertemuan 13\Guided\mahasiswa-app> php artisan migrate:status
Migration name ..... Batch / Status
2014_10_12_000000_create_users_table ..... [1] Ran
2014_10_12_100000_create_password_reset_tokens_table ..... [1] Ran
2019_08_19_000000_create_failed_jobs_table ..... [1] Ran
2019_12_14_000001_create_personal_access_tokens_table ..... [1] Ran
2026_01_04_181505_create_mahasiswas_table ..... [2] Ran

```

Sedangkan untuk membatalkan satu migration terakhir, digunakan perintah:

```

PS D:\xampp\htdocs\PPW\Pertemuan 13\Guided\mahasiswa-app> php artisan migrate:rollback --step=1
[INFO] Rolling back migrations.
Open folder in new window (ctrl + click)
2026_01_04_181505_create_mahasiswas_table ..... 13ms DONE

```

Perintah tersebut akan menghapus perubahan yang dihasilkan oleh satu file migration terakhir.

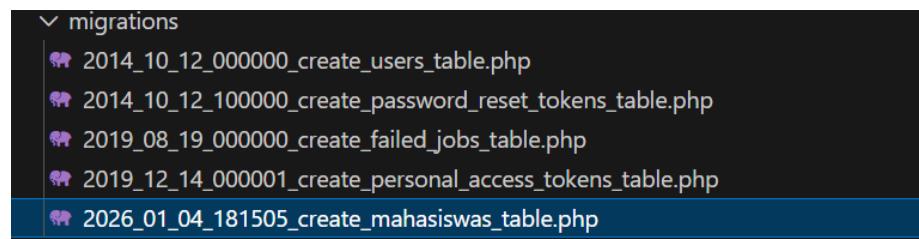
d. Pembuatan Migration Baru

File migration dapat dibuat secara manual, namun cara yang lebih efisien adalah menggunakan perintah Artisan. Format penamaan migration sebaiknya mengikuti pola <aksi>_<nama_tabel>_table agar mudah dipahami.

Contoh pembuatan migration tabel mahasiswa:

```
PS D:\xampp\htdocs\PPW\Pertemuan 13\Guided\mahasiswa-app> php artisan make:migration create_mahasiswa_table --create=mahasiswa
>>
[INFO] Migration [D:\xampp\htdocs\PPW\Pertemuan 13\Guided\mahasiswa-app\database\migrations\2026_01_04_181505_create_mahasiswa_table.php]
created successfully.
```

Perintah tersebut akan menghasilkan file migration baru pada folder database/migrations dengan nama yang disertai timestamp.



Setelah file migration dibuat, struktur tabel didefinisikan pada method up() menggunakan Schema Builder. Method down() digunakan untuk menghapus tabel apabila dilakukan rollback.

Source Code

```
<?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('mahasiswa', 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();
});

}

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

Struktur Tabel

The screenshot shows the phpMyAdmin interface with the following details:

Server: 127.0.0.1 » Database: laravel » Table: mahasiswa

Structure View (selected)

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	id 🛡	bigint(20)		UNSIGNED	No	None		AUTO_INCREMENT	Change Drop More
2	nim 💬	char(8)	utf8mb4_unicode_ci		No	None			Change Drop More
3	nama	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
4	tempat_lahir	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
5	tanggal_lahir	date			No	None			Change Drop More
6	fakultas	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
7	jurusan	varchar(255)	utf8mb4_unicode_ci		No	None			Change Drop More
8	ipk	decimal(3,2)			No	1.00			Change Drop More
9	created_at	timestamp			Yes	NULL			Change Drop More
	updated_at	timestamp			Yes	NULL			Change Drop More

e. Modifikasi dan Rollback Migration

Struktur tabel yang telah dibuat dapat disesuaikan kembali dengan kebutuhan aplikasi. Perubahan pada file migration mengharuskan proses rollback terlebih dahulu sebelum menjalankan migration ulang agar perubahan dapat diterapkan ke database.

Source code yang di ubah

```
public function up(){
    Schema::create('mahasiswa', 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();
    });
}
```

B. UNGUIDED

Soal:

1. Buatlah file migration untuk database yang akan digunakan pada tugas besar di kelas teori.
2. Buatlah fungsi-fungsi berikut untuk menyelesaikan tugas besar di matakuliah teori:
 - a. Buat fungsi untuk insert data menggunakan raw SQL Queries
 - b. Buat fungsi untuk insert data menggunakan Query Builder
 - c. Buat fungsi untuk insert data menggunakan Eloquent ORM

Jawaban:

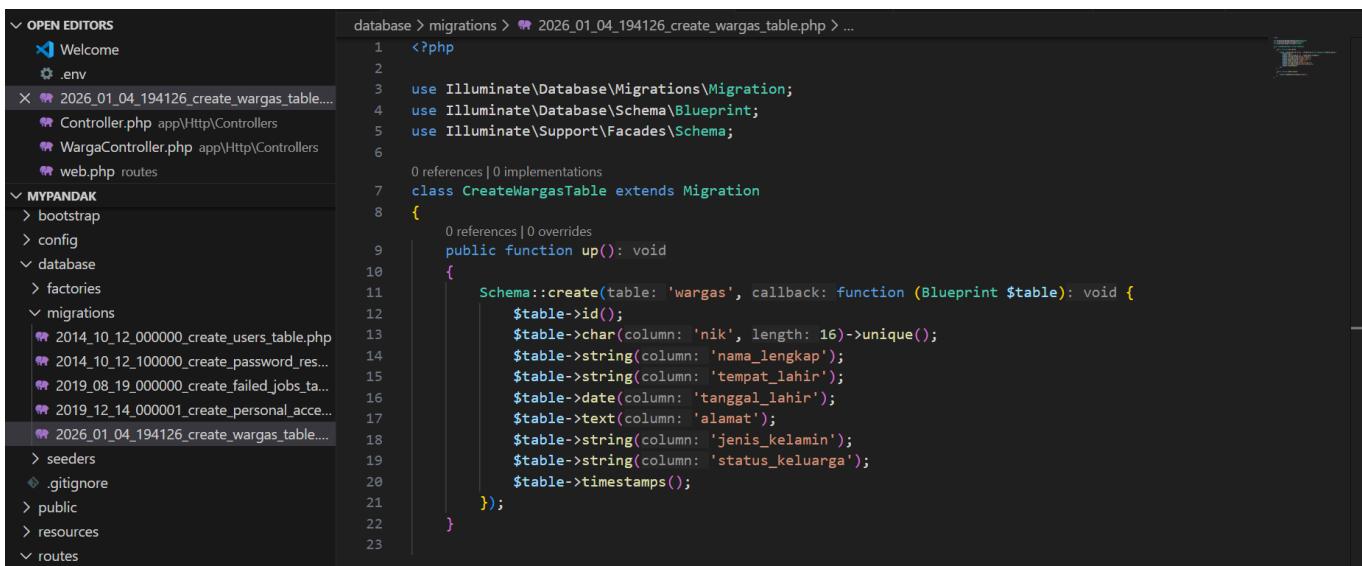
1. Berikut:

Buat file migration:

```
PS D:\xampp\htdocs\PPW\Pertemuan 13\Unguided\mypandak> php artisan make:migration create_wargas_table --create=wargas
• >>

[INFO] Migration [D:\xampp\htdocs\PPW\Pertemuan 13\Unguided\mypandak\database\migrations\2026_01_04_191747_create_wargas_
table.php] created successfully.
```

Edit File Migration



The screenshot shows a code editor with the following details:

- Left Sidebar:** Shows the project structure with the following files and folders:
 - OPEN EDITORS: Welcome, .env
 - MYPANDAK: bootstrap, config, database (with factories and migrations), factories, migrations (containing 2014.10.12_000000_create_users_table.php, 2014.10.12_100000_create_password_res..., 2019.08.19_000000_create_failed_jobs_ta..., 2019.12.14_000001_create_personal_acce..., 2026.01.04_194126_create_wargas_table.php), seeders, .gitignore, public, resources, routes.
- Right Panel:** Displays the content of the selected file, `2026_01_04_194126_create_wargas_table.php`. The code defines a `CreateWargasTable` class extending `Migration`. The `up()` method uses the `Schema::create` method to create a table named `wargas` with the following columns:
 - `$table->id();`
 - `$table->char(column: 'nik', length: 16)->unique();`
 - `$table->string(column: 'nama_lengkap');`
 - `$table->string(column: 'tempat_lahir');`
 - `$table->date(column: 'tanggal_lahir');`
 - `$table->text(column: 'alamat');`
 - `$table->string(column: 'jenis_kelamin');`
 - `$table->string(column: 'status_keluarga');`
 - `$table->timestamps();`

Kemudian

```
PS D:\xampp\htdocs\PPW\Pertemuan 13\Unguided\mypandak> php artisan migrate
● >>
    INFO  Preparing database.

    Creating migration table ..... 18ms DONE

    INFO  Running migrations.

    2014_10_12_000000_create_users_table ..... 26ms DONE
    2014_10_12_100000_create_password_reset_tokens_table ..... 9ms DONE
    2019_08_19_000000_create_failed_jobs_table ..... 28ms DONE
    2019_12_14_000001_create_personal_access_tokens_table ..... 33ms DONE
    2026_01_04_194126_create_wargas_table ..... 24ms DONE
```

2. Membuat Controller

A. INSERT DATA MENGGUNAKAN RAW SQL

edit WargaController.php

```
use Illuminate\Support\Facades\DB;

public function insertRaw()
{
    $result = DB::insert(
        "INSERT INTO wargas
        (nik, nama_lengkap, tempat_lahir, tanggal_lahir, alamat, jenis_kelamin,
        status_keluarga)
        VALUES
        ('3305123456780001',
        'Budi Santoso',
        'Purwokerto',
        '2001-05-12',
        'Jl. Merdeka No. 10',
        'Laki-laki',
        'Kepala Keluarga')
    );
    dump($result);
}
```

```
}
```

Tambahkan Route

```
use App\Http\Controllers\WargaController;  
  
Route::get('/insert-raw', [WargaController::class, 'insertRaw']);
```

B. INSERT DATA MENGGUNAKAN QUERY BUILDER

```
public function insertQB()  
{  
    $result = DB::table('wargas')->insert([  
        'nik' => '3305123456780002',  
        'nama_lengkap' => 'Siti Aminah',  
        'tempat_lahir' => 'Banyumas',  
        'tanggal_lahir' => '2002-03-21',  
        'alamat' => 'Jl. Melati No. 5',  
        'jenis_kelamin' => 'Perempuan',  
        'status_keluarga' => 'Istri',  
    ]);  
  
    dump($result);  
}
```

C. INSERT DATA MENGGUNAKAN ELOQUENT ORM

```
<?php  
  
namespace App\Models;  
  
use Illuminate\Database\Eloquent\Model;  
  
class Warga extends Model  
{  
    protected $table = 'wargas';
```

```

protected $fillable = [
    'nik',
    'nama_lengkap',
    'tempat_lahir',
    'tanggal_lahir',
    'alamat',
    'jenis_kelamin',
    'status_keluarga'
];
}

```

Maka Hasil setelah di migrasi:

The screenshot shows the phpMyAdmin interface with the following details:

- Database:** mypadak_db
- Table:** wargas
- Rows:** 3 total
- Query took:** 0.0004 seconds
- SQL Query:** SELECT * FROM `wargas`
- Table Structure:**

	ID	Nik	Nama Lengkap	Tempat Lahir	Tanggal Lahir	Alamat	Jenis Kelamin	Status Keluarga	Created At	Updated At
<input type="checkbox"/>	1	3305123456780001	Budi Santoso	Purwokerto	2001-05-12	Jl. Merdeka No. 10	Laki-laki	Kepala Keluarga	NULL	NULL
<input type="checkbox"/>	2	3305123456780002	Siti Aminah	Banyumas	2002-03-21	Jl. Melati No. 5	Perempuan	Istri	NULL	NULL
<input type="checkbox"/>	3	3305123456780003	Ahmad Fauzi	Cilacap	2000-11-10	Jl. Kenanga No. 7	Laki-laki	Anak	2020-01-04 20:04:10	2020-01-04 20:04:10

BAB III

PENUTUP

A. Kesimpulan

Modul 12 membahas penggunaan Migration pada Laravel sebagai cara untuk mengelola struktur database secara teratur melalui kode program. Dengan migration, pembuatan, perubahan, dan penghapusan tabel dapat dilakukan dengan lebih mudah dan aman tanpa harus mengubah database secara langsung. Praktikum ini juga menunjukkan bahwa Laravel menyediakan tiga cara untuk mengelola data, yaitu Raw SQL, Query Builder, dan Eloquent ORM. Ketiga metode tersebut dapat digunakan untuk melakukan proses input data ke dalam database sesuai dengan kebutuhan aplikasi. Melalui penerapan migration dan berbagai metode pengolahan data tersebut, pengelolaan database menjadi lebih rapi, terkontrol, dan mudah dikembangkan dalam pembuatan aplikasi berbasis Laravel.