LAPORAN PRAKTIKUM PEMROGRAMAN WEB

PRAKTIKUM 12 – MIGRATION, SEEDER, FACTORY

Disusun oleh:

Nama : Afif Imam Rahadi

Nim : L0122006

Kelas : A

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI DAN SAINS DATA

UNIVERSITAS SEBELAS MARET

2024

1. **Screenshot Source Code**

Model

```php
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Student extends Model
{
    use HasFactory;
    protected $fillable =
    [
        'nama', 'nim', 'prodi', 'kelas', 'angkatan'
    ];
}
```

Code di atas adalah sebuah model di Laravel yang mewakili tabel students dalam database. Model ini terletak dalam namespace App\Models, yang menunjukkan bahwa model ini berada dalam direktori app/Models dari aplikasi Laravel. Dua kelas dari framework Laravel diimpor, yaitu HasFactory dan Model. HasFactory digunakan untuk mendukung pembuatan instance model menggunakan factory, yang berguna untuk pengujian dan pengisian data tiruan, sementara Model adalah kelas dasar yang menyediakan fungsionalitas ORM (Object-Relational Mapping) Eloquent di Laravel. Kelas Student didefinisikan dan mewarisi semua fungsionalitas dari kelas Model, menjadikannya model Eloquent yang dapat digunakan untuk berinteraksi dengan tabel students dalam database.

Model ini menggunakan trait HasFactory, memberikan kemampuan untuk menggunakan factory dalam pembuatan data tiruan atau pengujian. Atribut fillable didefinisikan sebagai array yang berisi daftar kolom-kolom yang diizinkan untuk pengisian massal (mass assignment), yaitu nama, nim, prodi, kelas, dan angkatan. Ini bertujuan untuk mencegah serangan yang dikenal sebagai Mass Assignment Vulnerability, dengan memastikan hanya kolom-kolom yang diizinkan saja yang dapat diisi melalui metode mass assignment. Secara keseluruhan, model Student ini berfungsi sebagai representasi dari tabel students dalam database, memungkinkan interaksi yang mudah dengan data mahasiswa melalui ORM Eloquent Laravel.

Route

```php
<?php

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

Route::get('/', function () {
    return view('welcome');
}) -> name('welcome');

Route::get('/students', [StudentController::class, 'index']) -> name('students');

Route::get('/students/form', function () {
    return view('student.form');
});

Route::post('/students/add', [StudentController::class,'store']) -> name('add-students');
Route::get('/students/delete/{id}', [StudentController::class,'destroy']);
Route::get('/students/edit/{student}', [StudentController::class,'edit']);
Route::post('/students/update/{student}', [StudentController::class,'update']) -> name('students.update'
```

Code ini adalah konfigurasi rute (routes) untuk aplikasi Laravel. Baris pertama menggunakan use untuk mengimpor StudentController dan Route. Rute pertama menggunakan metode get untuk mendefinisikan rute dasar '/', yang mengembalikan view welcome saat diakses dan menamainya welcome. Rute kedua juga menggunakan metode get untuk mendefinisikan URL /students, yang memetakan ke metode index dari StudentController dan menamainya students. Selanjutnya, rute dengan URL /students/form mengembalikan view student.form tanpa menggunakan controller. Rute post dengan URL /students/add memetakan ke metode store dari StudentController dan menamainya add-students, yang biasanya digunakan untuk menambahkan data baru ke dalam database. Rute get dengan URL /students/delete/{id} memetakan ke metode destroy dari StudentController yang digunakan untuk menghapus data berdasarkan id yang diberikan. Rute berikutnya adalah untuk URL /students/edit/{student}, yang memetakan ke metode edit dari StudentController untuk mengedit data berdasarkan entitas student yang diberikan. Terakhir, rute post dengan URL /students/update/{student} memetakan ke metode update dari StudentController dan menamainya students.update, yang biasanya digunakan untuk memperbarui data yang ada. Secara keseluruhan, rute-rute ini mengatur navigasi aplikasi, menghubungkan URL dengan tindakan yang sesuai di controller dan view, memungkinkan interaksi pengguna dengan data mahasiswa.

Student Controller

```php
<?php

namespace App\Http\Controllers;

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

class StudentController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index()
    {
        $students = DB::table('students')->get();
        return view('student.index', ['students' => $students]);
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request)
    {
        $request->validate([
            'nama'=> 'required',
            'nim'=> 'required',
            'prodi'=> 'required',
            'kelas'=> 'required',
            'angkatan'=> 'required',
        ]);

        $student = new Student;
        // kiri objek model - kanan dari request
        $student->nama = $request->nama;
        $student->nim = $request->nim;
        $student->prodi = $request->prodi;
        $student->kelas = $request->kelas;
        $student->angkatan = $request->angkatan;

        $student->save();

        return redirect()->route('students') ->with('success', 'Data mahasiswa berhasil ditambahkan');
    }

    /**
     * Display the specified resource.
     */
    public function show(Student $student)
    {
        //
    }

    /**
     * Show the form for editing the specified resource.
     */
    public function edit(Student $student)
    {
        return view('student.edit', ['student' => $student]);
    }

    /**
     * Update the specified resource in storage.
     */
    public function update(Request $request, Student $student)
    {
        $request->validate([
            'nama'=> 'required',
            'nim'=> 'required',
            'prodi'=> 'required',
            'kelas'=> 'required',
            'angkatan'=> 'required',
        ]);

        $student->update($request->all());
        return redirect()->route('students')-> with('success', 'Data berhasil diupdate');
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy($id)
    {
        $student = DB::table('students') -> where('id', $id)->delete();
        return redirect() -> route('students');
    }
}
```

Code ini adalah definisi dari StudentController di Laravel yang mengatur logika aplikasi terkait operasi CRUD (Create, Read, Update, Delete) untuk model Student. Controller ini berada di dalam namespace App\Http\Controllers yang menunjukkan bahwa file ini berada di direktori

app/Http/Controllers. Beberapa kelas diimpor di bagian awal, termasuk Student dari App\Models, Request dari Illuminate\Http\Request, dan DB dari Illuminate\Support\Facades.

Metode index dalam controller ini mengambil semua data dari tabel students menggunakan query builder DB::table('students')->get(), dan mengirimkannya ke view student.index. Metode create didefinisikan tetapi kosong, menunjukkan bahwa form untuk membuat resource baru belum diimplementasikan. Metode store menerima Request dari form, memvalidasi input untuk memastikan semua kolom wajib diisi (nama, nim, prodi, kelas, dan angkatan), kemudian membuat instance baru dari Student dan menyimpan data yang valid ke dalam database. Setelah data berhasil disimpan, pengguna akan dialihkan ke route students dengan pesan sukses.

Metode show didefinisikan tetapi kosong, biasanya digunakan untuk menampilkan detail dari satu resource Student. Metode edit menerima instance Student sebagai parameter, mengembalikan view student.edit dengan data mahasiswa yang akan diedit. Metode update menerima Request dan instance Student, memvalidasi input, kemudian memperbarui data mahasiswa yang ada dengan data baru dari request. Setelah berhasil memperbarui data, pengguna akan dialihkan kembali ke route students dengan pesan sukses.

Metode destroy menerima parameter $id, menghapus data mahasiswa dari tabel students berdasarkan id yang diberikan, kemudian mengarahkan pengguna kembali ke route students. Metode ini menggunakan query builder untuk operasi penghapusan. Secara keseluruhan, controller ini menangani semua operasi dasar CRUD untuk model Student, memastikan data dapat dibuat, dibaca, diperbarui, dan dihapus sesuai kebutuhan aplikasi.

Migration

```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('students', function (Blueprint $table) {
            $table->id();
            $table->string('nama');
            $table->string('nim');
            $table->string('prodi');
            $table->char('kelas');
            $table->integer('angkatan');
            $table->timestamps();
        });
    }
```

Kode ini adalah definisi migrasi database di Laravel yang digunakan untuk membuat dan menghapus tabel students. Migrasi ini terletak dalam namespace Illuminate\ Database\Migrations dan menggunakan beberapa kelas dari Laravel termasuk Blueprint dan Schema. Kode ini didefinisikan dalam sebuah anonymous class yang mewarisi Migration.

Metode up didefinisikan untuk menjalankan migrasi, di mana Schema::create digunakan untuk membuat tabel students. Di dalam fungsi ini, sebuah instance Blueprint diberikan sebagai parameter untuk mendefinisikan struktur tabel. Tabel students memiliki beberapa kolom: id yang merupakan primary key dengan auto-increment, nama dan nim yang keduanya adalah string, prodi yang juga merupakan string, kelas yang merupakan karakter (char), angkatan yang merupakan integer, dan dua kolom timestamp (created_at dan updated_at) yang secara otomatis dikelola oleh Laravel. Metode down didefinisikan untuk membalikkan migrasi, yang dalam hal ini menghapus tabel students jika ada. Metode ini menggunakan Schema:: dropIfExists untuk memastikan bahwa tabel students dihapus dengan aman.

Database Seeder

```php
<?php

namespace Database\Seeders;

use App\Models\User;
// use Illuminate\Database\Console\Seeds\WithoutModelEvents;
use App\Models\Student;
use Database\Factories\StudentFactory;
use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder
{
    /**
     * Seed the application's database.
     */
    public function run(): void
    {

        $this->call([
            StudentSeeder::class
        ]);
    }
}
```

Code di atas merupakan file DatabaseSeeder.php. Kelas ini menggunakan namespace Database\Seeders untuk mengorganisir kode dan menghindari konflik nama kelas. Bagian atas terdapat use yang mengimpor beberapa kelas yang diperlukan App\Models\User, App\Models\Student, Database\Factories\StudentFactory, dan Illuminate\Database\Seeder. User dan Student adalah model Eloquent yang mewakili tabel users dan students di database, sementara StudentFactory digunakan untuk menghasilkan data tiruan untuk model Student, dan Seeder adalah kelas dasar yang disediakan oleh Laravel untuk membuat seeders. Selanjutnya, kelas DatabaseSeeder didefinisikan dengan memperluas (extends) kelas Seeder, yang bertanggung jawab untuk menjalankan semua seeder lainnya yang diperlukan untuk mengisi database dengan data awal atau data tiruan. Metode run, yang akan dipanggil saat seeder dijalankan, dijelaskan dengan komentar docblock bahwa metode ini digunakan untuk mengisi database aplikasi. Dalam metode run, this->call([StudentSeeder::class]) memanggil seeder lain, yaitu StudentSeeder, yang dijalankan dengan metode call untuk mengisi tabel students dengan data awal atau data tiruan. Kode ini ditutup dengan tanda kurung kurawal yang menutup metode run dan kelas DatabaseSeeder.

Student Seeder

```php
<?php

namespace Database\Seeders;

use App\Models\Student;
use Illuminate\Database\Seeder;
use Illuminate\Database\Console\Seeds\WithoutModelEvents;

class StudentSeeder extends Seeder
{
    /**
     * Run the database seeds.
     */
    public function run(): void
    {
        Student::factory()->count(10)->create();
    }
}
```

Code di atas merupakan code dari StudentSeeder.php. File ini berada dalam namespace Database\Seeders, yang mengorganisir kode dan mencegah konflik nama. Pada bagian awal terdapat syntax use yang digunakan untuk mengimpor kelas Student dari App\Models dan kelas Seeder dari Illuminate\Database\Seeder. Kelas StudentSeeder didefinisikan dengan memperluas (extends) kelas Seeder. Kelas ini memiliki satu metode, run, yang akan dipanggil saat seeder dijalankan. Metode run mengisi tabel students di database dengan data tiruan menggunakan Student::factory()->count(10)->create(), yang memanggil pabrik (factory) untuk membuat 10 instance model Student dan menyimpannya ke dalam database. Pabrik ini menghasilkan data tiruan yang sesuai dengan definisi dalam StudentFactory.

Student Factory

```php
<?php

namespace Database\Factories;

use App\Models\Student;
use Illuminate\Database\Eloquent\Factories\Factory;

/**
 * @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Student>
 */
class StudentFactory extends Factory
{
    /**
     * Define the model's default state.
     *
     * @return array<string, mixed>
     */

    public function definition()
    {
        return [
            'nama'=> $this->faker->name,
            'nim' => 'L0122' . $this->faker->unique()->numerify('###'),
            'prodi'=> $this->faker->randomElement(['Informatika', 'Sains Data']),
            'kelas'=> $this->faker->randomElement(['A', 'B', 'C', 'D', 'E']),
            'angkatan'=> $this->faker->randomElement(['2020', '2021', '2022', '2023', '2024']),
        ];
    }
}
```

Code di atas merupakan code dari StudentFactory.php. File ini berada dalam namespace Database\Factories, yang mengorganisir kode dan mencegah konflik nama. Kelas StudentFactory menggunakan App\Models\Student dan Illuminate\Database\Eloquent \Factories\Factory, diimpor menggunakan pernyataan use. Kelas ini memperluas (extends) kelas Factory, yang disediakan oleh Laravel untuk mendefinisikan pabrik pembuatan data tiruan untuk model Eloquent. Komentar docblock di atas kelas menunjukkan bahwa kelas ini merupakan ekstensi dari Factory untuk model Student. Metode definition dalam kelas ini bertanggung jawab untuk mendefinisikan keadaan default model Student. Metode ini mengembalikan sebuah array yang berisi nilai-nilai atribut untuk mengisi kolom-kolom dalam tabel students. Atribut nama diisi dengan nama acak yang dihasilkan oleh $this->faker->name. Atribut nim diisi dengan string yang diawali dengan 'L0122' diikuti oleh tiga digit angka acak yang unik, dihasilkan oleh $this->faker->unique()->numerify('###'). Atribut prodi diisi dengan salah satu dari dua nilai acak, yaitu 'Informatika' atau 'Sains Data', dipilih menggunakan $this->faker->randomElement(['Informatika', 'Sains Data']). Atribut kelas diisi dengan salah satu dari lima nilai acak, yaitu 'A', 'B', 'C', 'D', atau 'E', dipilih menggunakan $this->faker->randomElement(['A', 'B', 'C', 'D', 'E']). Atribut angkatan diisi dengan salah satu dari lima nilai acak, yaitu '2020', '2021', '2022', '2023', atau '2024', dipilih menggunakan $this->faker->randomElement(['2020', '2021', '2022', '2023', '2024']).

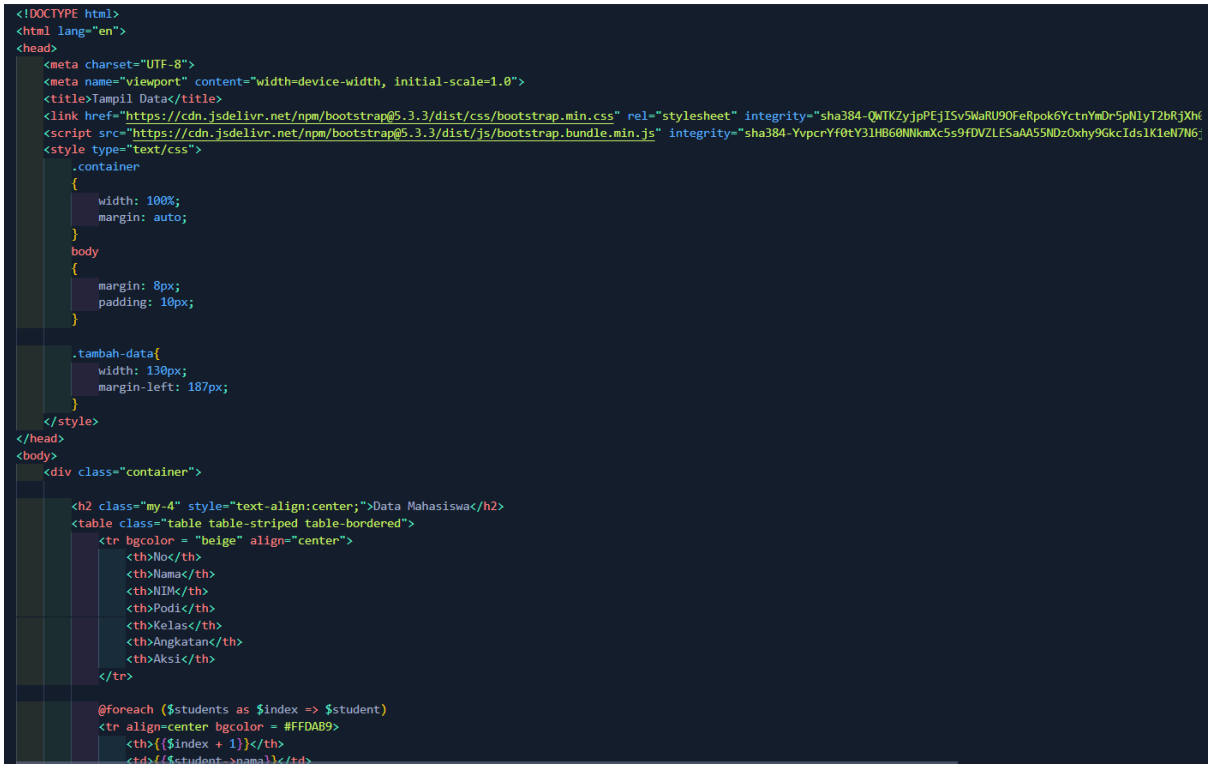
Welcome Views

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>welcome</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRj
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7
    <style>
        body
        {
            background-color: azure;
        }
        .centered {
            height: 100vh;
            display: flex;
            flex-direction: column;
            justify-content: center;
            align-items: center;
            text-align: center;
        }
    </style>
</head>
<body>
    <div class="container centered">
        <h1>Selamat Datang Dalam Web Pendataan Mahasiswa</h1>
        <div class="mt-4">
            <button  class="btn btn-primary mx-2">
                <a href="/students" class="text-light" style="text-decoration: none;">List Daftar Mahasiswa</a>
            </button>

            <button  class="btn btn-primary mx-2">
                <a href="/students/form" class="text-light" style="text-decoration: none;">Tambah Data Mahasiswa</a>
            </button>
        </div>
    </div>
</body>
</html>
```

Pada bagian welcome ini, merupakan teks html biasa yang menggunakan bootsrap sebagai bantuan styling. Pada halaman ini hanya sebagai halaman utama sebelum user dapat melihat dan megisi data mahasiswa. Di halaman ini, terdapat 2 button yang mengarah ke halaman yang berbeda.

Student/index view



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tampil Data</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6
    <style type="text/css">
        .container
        {
            width: 100%;
            margin: auto;
        }
        body
        {
            margin: 8px;
            padding: 10px;
        }

        .tambah-data{
            width: 130px;
            margin-left: 187px;
        }
    </style>
</head>
<body>
    <div class="container">

        <h2 class="my-4" style="text-align:center;">Data Mahasiswa</h2>
        <table class="table table-striped table-bordered">
            <tr bgcolor = "beige" align="center">
                <th>No</th>
                <th>Nama</th>
                <th>NIM</th>
                <th>Podi</th>
                <th>Kelas</th>
                <th>Angkatan</th>
                <th>Aksi</th>
            </tr>

            @foreach ($students as $index => $student)
            <tr align=center bgcolor = #FFDAB9>
                <th>{{$index + 1}}</th>
                <td>{{$student->nama}}</td>
```

Ini adalah halaman utama dari students. Halaman utama ini akan menampilkan data mahasiswa yang berbentuk tabel. Di dalam tabel ini terdapat data-data yang telah diinputkan oleh user. Pada setiap data terdapat aksi yang dapat dilakukan oleh user, yaitu edit dan delete. Oleh karena itu, user dapat mengatur data sesuai yang diinginkan.

Student/form View

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh
    <title style="text-align: center;">Form Data Mahasiswa</title>
</head>
<body>
    <h1 class="my-4 mx-3">Form Data Mahasiswa</h1>
    <div class="container w-75 m-4 mx-5"  >
        <form method="POST" action={{ route('add-students')}}>
            @csrf
            <div>
                <div>
                    <label class="input-group-text w-25">Nama</label>
                </div>
                <div>
                    <input class="input-group-text w-25" type= "text" name="nama" size="30" placeholder="Masukkan nama anda" >
                </div>
            </div>

            <div>
                <div>
                    <label class="input-group-text w-25 mt-3">NIM</label>
                </div>
                <div>
                    <input class="input-group-text w-25" type="text" name="nim" size="30" placeholder="Masukkan nim anda"></input>
                </div>
            </div>

            <div>
                <div>
                    <label class="input-group-text w-25 mt-3">Prodi</label>
                </div>
                <div>
                    <input class="input-group-text w-25" type= "text" name="prodi" size="30" placeholder="Masukkan prodi anda">
                </div>
            </div>

            <div>
                <div>
                    <label class="input-group-text w-25 mt-3" >Kelas</label>
                </div>
```

Ini adalah code halaman dari form data mahasiswa. Di sini user dapat mengisi data sesuai dengan temaptanya. Terdapat tag form agar lebih terstruktur untuk user input. Setiap form sudah memiliki name masing-masing yang nantinya akan diproses untuk dimasukkan ke dalam database.

Student/edit view

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh
    <title style="text-align: center;">Edit Form Data Mahasiswa</title>
</head>
<body>
    <h1 class="my-4 mx-3">Edit Form Data Mahasiswa</h1>
    <div id="mainContainer" class="container w-75 m-4 mx-5" >
        <form method="POST" action={{ route('students.update', $student -> id )}}>
            @csrf
            <div>
                <div>
                    <label class="input-group-text w-25">Nama</label>
                </div>
                <div>
                    <input class="input-group-text w-25" type= "text" name="nama" value="{{$student->nama}}" size="30" >
                </div>
            </div>

            <div>
                <div>
                    <label class="input-group-text w-25 mt-3">NIM</label>
                </div>
                <div>
                    <input class="input-group-text w-25" type="text" name="nim" value="{{$student->nim}}" size="30"></input>
                </div>
            </div>

            <div>
                <div>
                    <label class="input-group-text w-25 mt-3">Prodi</label>
                </div>
                <div>
                    <input class="input-group-text w-25" type= "text" name="prodi" value="{{$student->prodi}}" size="30">
                </div>
            </div>

            <div>
                <div>
                    <label class="input-group-text w-25 mt-3" >Kelas</label>
                </div>
```

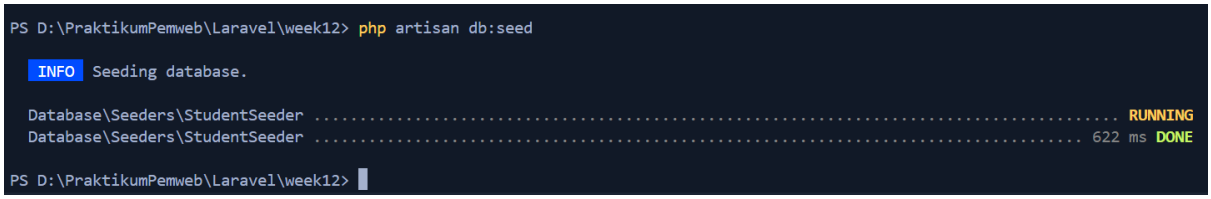
Tidak berbeda jauh dengan views form. Yang membedakannya adalah pada halaman edit ini, pada bagian form sudah memiliki value masing-masing yang diambil dari databasenya. Sehingga saat user ingin melakukan edit, sudah ada data sebelumnya sehingga tidak bingung untuk mengganti yang bagian mana.

2. Screenshot Ouput

Home

Ini adalah halaman home. Halaman home sangat sederhana yang berisi tulisan selamat datang. Di bawahnya, terdapat button list daftar mahasiswa dan tambah data. Jika button ini di klik, maka program akan mengarahkannya ke halaman yang lain.

Menjalankan db:seed



Untuk menggunakan seeder dan factory yang sudah kita buat, maka user perlu melakukan perintah Laravel berupa php artisan db:seed agar seeder dapat dilakukan. Setelah itu, maka dapat melakukan serve agar web dapat dibuka.

Student/Index

Data Mahasiswa

No	Nama	NIM	Podi	Kelas	Angkatan	Aksi
1	Nova Abbott	L0122129	Sains Data	D	2021	Edit Delete
2	Leanna Schiller	L0122277	Informatika	D	2024	Edit Delete
3	Maggie Hintz	L0122243	Informatika	A	2023	Edit Delete
4	Estel Hayes	L0122596	Sains Data	B	2022	Edit Delete
5	Annalise Goyette	L0122753	Informatika	D	2022	Edit Delete
6	Kellen Hoppe	L0122373	Sains Data	C	2023	Edit Delete
7	Tyrel Koepp	L0122923	Sains Data	E	2022	Edit Delete
8	Dr. Adolfo Johns Jr.	L0122631	Informatika	D	2023	Edit Delete
9	Lorenzo Bednar	L0122652	Informatika	A	2023	Edit Delete
10	Rylan Stark II	L0122501	Informatika	B	2023	Edit Delete

Tambah Data Home

Dapat terlihat bahwa setalah dijalankan, isi tabel terdapat 10 nama random dengan properties lainnya yang sudah terisi. Nim diawal dengan L0122 dan 3 digit angka random secara unique. Lalu prodi hanya ada dua pilihan saja yaitu sains data atau informatika. Lalu kelas dari A hingga E dan Angkatan antara 2020-2024.

Edit

Edit Form Data Mahasiswa

Nama
Nova Abbott

NIM
L0122129

Prodi
Sains Data

Kelas
D

Angkatan
2021

Edit Data

Untuk melakukan edit dan delete ini sama seperti sebelumnya sehingga tidak ada proses yang berbeda. Hanya saja data ini terisi secara random menggunakan seeder yang memudahkan user agar tidak mengisi data secara manual.