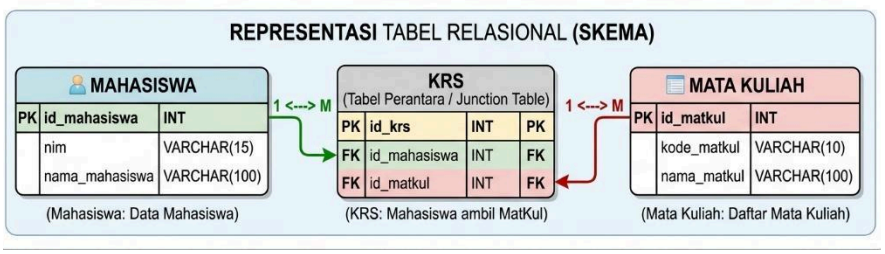
Contoh relasi 3 tabel sederhana untuk sistem database mahasiswa terdiri dari tabel Mahasiswa, Mata Kuliah, dan KRS (Kartu Rencana Studi) sebagai tabel perantara:

Struktur 3 Tabel

- **Mahasiswa**: Menyimpan data pribadi setiap mahasiswa.
    - id_mahasiswa (Primary Key)
    - nim
    - nama_mahasiswa
- **Mata Kuliah**: Menyimpan daftar mata kuliah yang tersedia di kampus.
    - id_matkul (Primary Key)
    - kode_matkul
    - nama_matkul
- **KRS (Kartu Rencana Studi)**: Tabel relasi penghubung (*many-to-many*) antara mahasiswa dan mata kuliah.
    - id_krs (Primary Key)
    - id_mahasiswa (Foreign Key merujuk ke tabel Mahasiswa)
    - id_matkul (Foreign Key merujuk ke tabel Mata Kuliah

Ilustrasi Relasi

- Satu **Mahasiswa** dapat mengambil banyak **Mata Kuliah** melalui tabel **KRS**.
- Satu **Mata Kuliah** dapat diambil oleh banyak **Mahasiswa** melalui tabel **KRS**.




```
-- ========================================

-- 1. PEMBUATAN DATABASE

-- ========================================

CREATE DATABASE IF NOT EXISTS db_akademik;

USE db_akademik;


-- ========================================

-- 2. PEMBUATAN TABEL INDUK (PARENT TABLES)
```

```
-- ========================================

-- Tabel Mahasiswa
CREATE TABLE mahasiswa (
   id_mahasiswa INT AUTO_INCREMENT PRIMARY KEY,
   nim VARCHAR(15) UNIQUE NOT NULL,
   nama_mahasiswa VARCHAR(100) NOT NULL
);

-- Tabel Mata Kuliah
CREATE TABLE mata_kuliah (
   id_matkul INT AUTO_INCREMENT PRIMARY KEY,
   kode_matkul VARCHAR(10) UNIQUE NOT NULL,
   nama_matkul VARCHAR(100) NOT NULL
);

-- ========================================
-- 3. PEMBUATAN TABEL PERANTARA (JUNCTION TABLE)
-- ========================================

-- Tabel KRS (Kartu Rencana Studi) dengan Foreign Key
CREATE TABLE krs (
   id_krs INT AUTO_INCREMENT PRIMARY KEY,
   id_mahasiswa INT NOT NULL,
   id_matkul INT NOT NULL,

   -- Menghubungkan Foreign Key ke tabel mahasiswa dan mata_kuliah
   CONSTRAINT fk_mahasiswa FOREIGN KEY (id_mahasiswa)
      REFERENCES mahasiswa(id_mahasiswa)
      ON DELETE CASCADE ON UPDATE CASCADE,
   CONSTRAINT fk_matkul FOREIGN KEY (id_matkul)
```

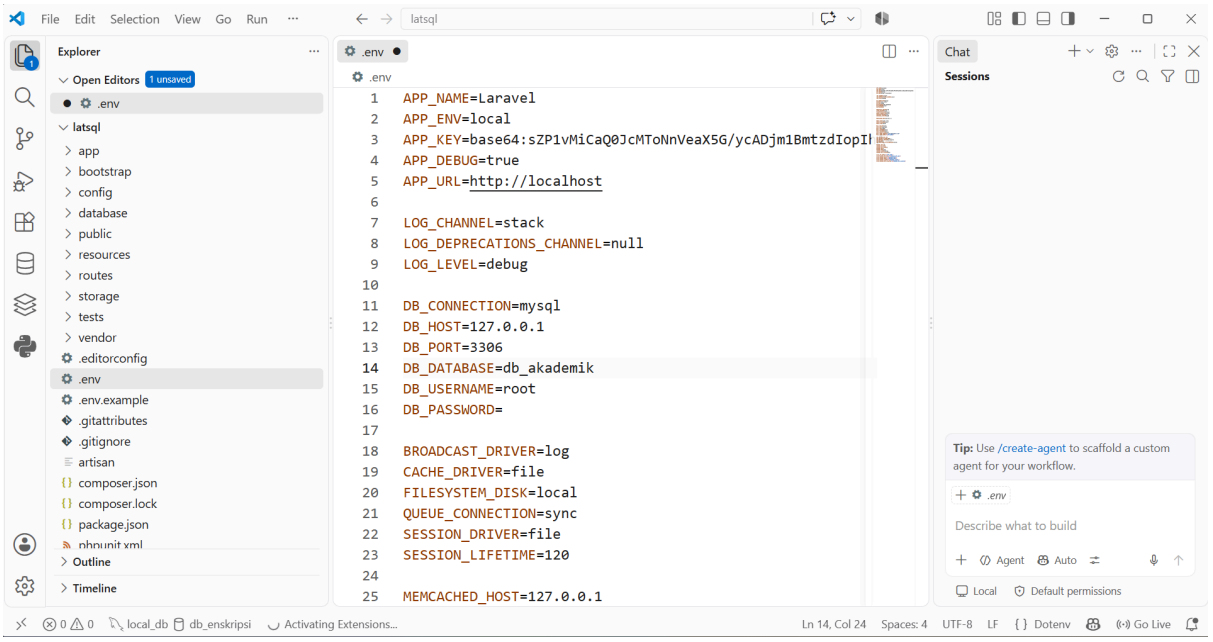
```
    REFERENCES mata_kuliah(id_matkul)
    ON DELETE CASCADE ON UPDATE CASCADE
);

-- ========================================
-- 4. CONTOH PENGISIAN DATA (DML)
-- ========================================

-- Masukkan data mahasiswa
INSERT INTO mahasiswa (nim, nama_mahasiswa) VALUES
('230101', 'Budi Santoso'),
('230102', 'Siti Rahma'),
('230103', 'Ahmad Fauzi');

-- Masukkan data mata kuliah
INSERT INTO mata_kuliah (kode_matkul, nama_matkul) VALUES
('IF101', 'Pemrograman Web'),
('IF102', 'Basis Data'),
('IF103', 'Algoritma Pemrograman');

-- Masukkan data KRS (Relasi Many-to-Many)
INSERT INTO krs (id_mahasiswa, id_matkul) VALUES
(1, 1), -- Budi mengambil Pemrograman Web
(1, 2), -- Budi mengambil Basis Data
(2, 1), -- Siti mengambil Pemrograman Web
(2, 3), -- Siti mengambil Algoritma Pemrograman
(3, 2); -- Ahmad mengambil Basis Data

-- ========================================
-- 5. CONTOH QUERY MENAMPILKAN DATA (JOIN)
-- ========================================
```

```
SELECT
    m.nim,
    m.nama_mahasiswa,
    mk.kode_matkul,
    mk.nama_matkul
FROM krs k
JOIN mahasiswa m ON k.id_mahasiswa = m.id_mahasiswa
JOIN mata_kuliah mk ON k.id_matkul = mk.id_matkul;
```

Langkah 1: Konfigurasi Koneksi Database (.env)

Buka file .env di root project Laravel Anda, lalu sesuaikan bagian konfigurasi database agar terhubung ke database db_akademik:

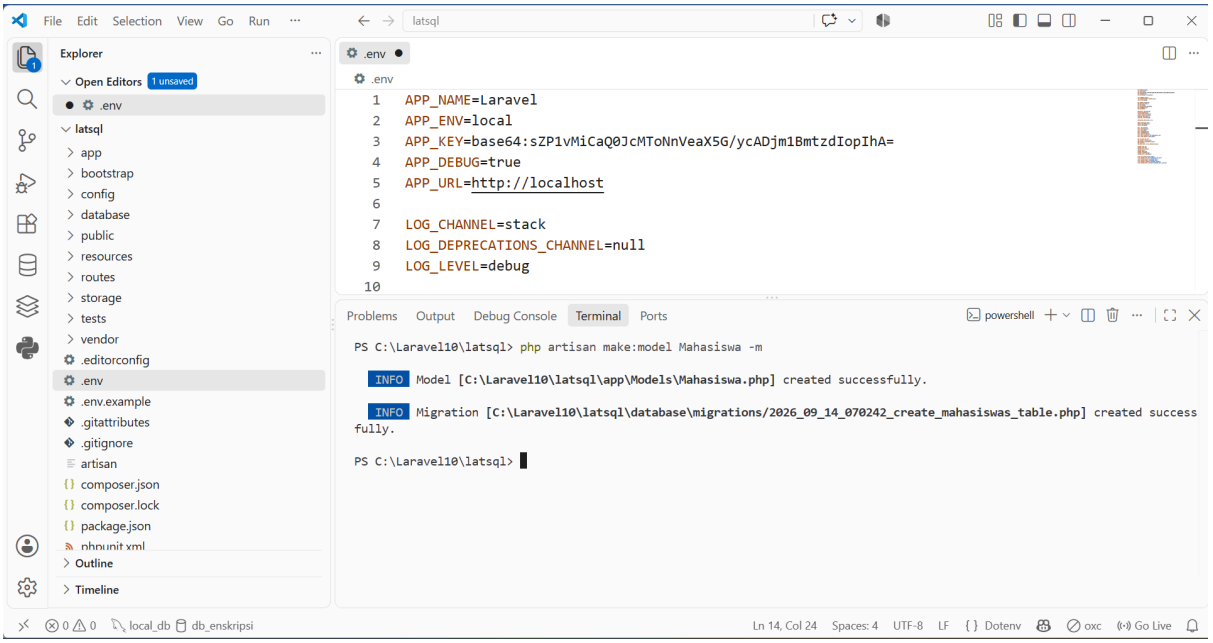
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=db_akademik
DB_USERNAME=root
DB_PASSWORD=
```

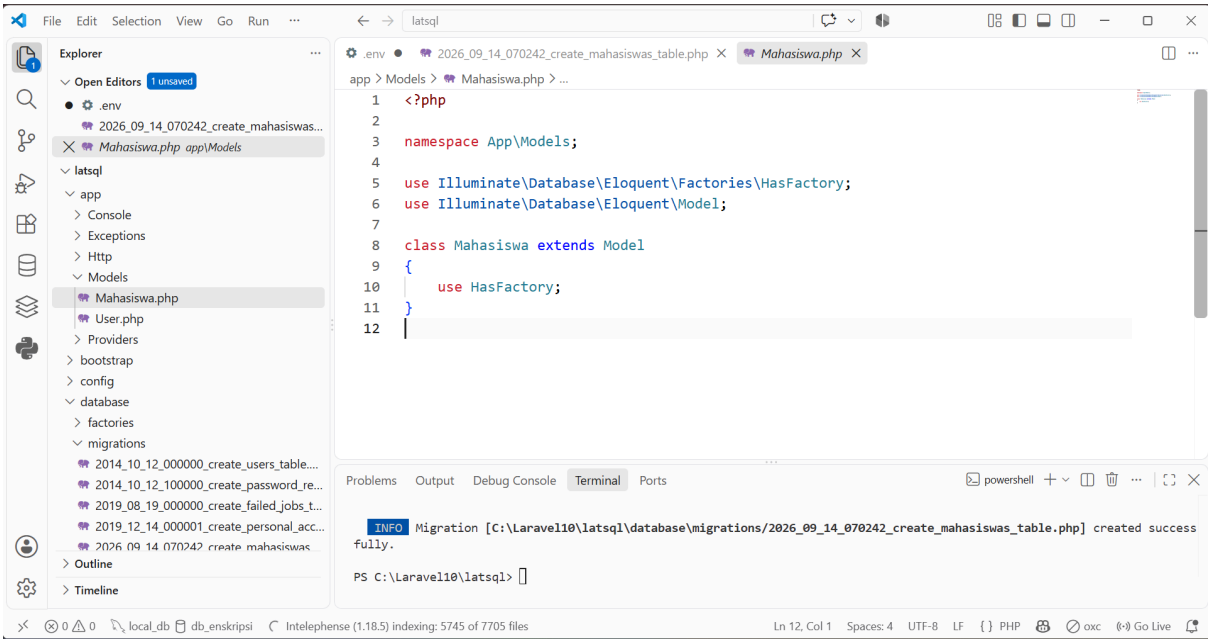


Langkah 2: Membuat Model dan Migration

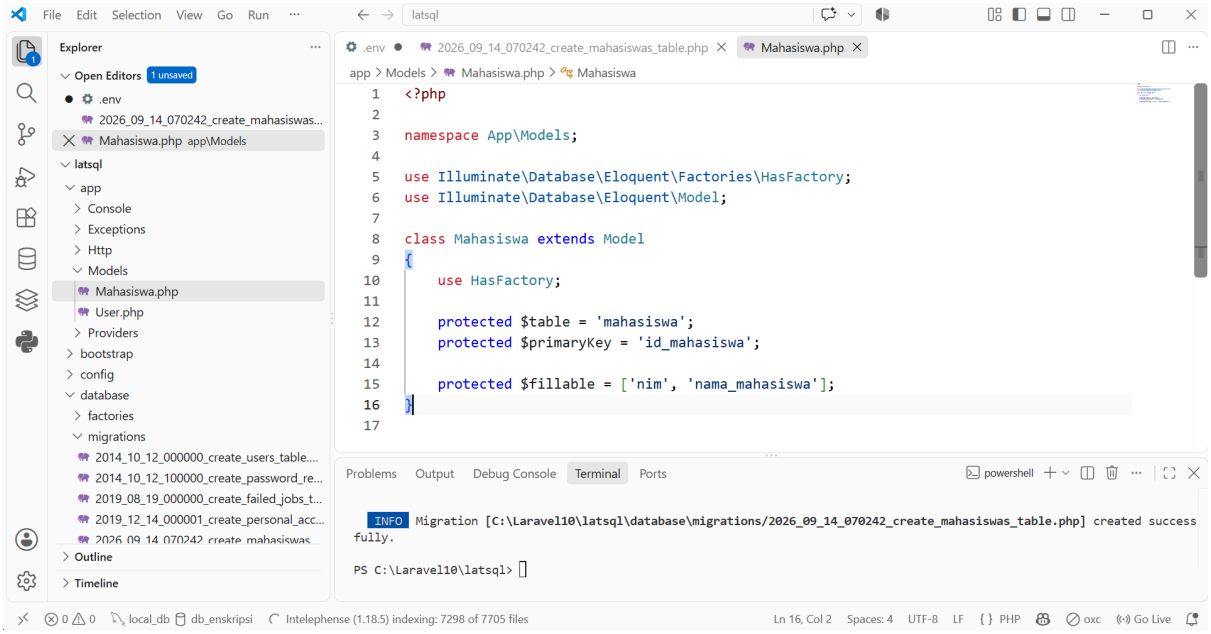
Gunakan Artisan CLI untuk membuat Model Mahasiswa sekaligus file migration-nya:

php artisan make:model Mahasiswa -m





ganti



namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Mahasiswa extends Model

{

use HasFactory;

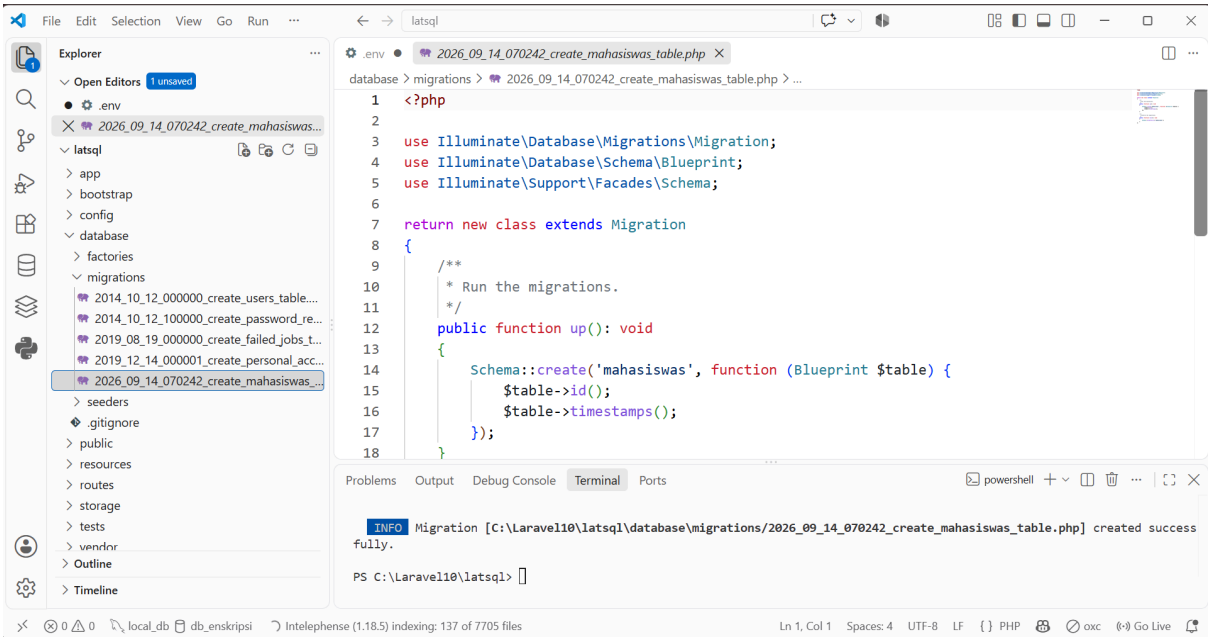
protected $table = 'mahasiswa';

protected $primaryKey = 'id_mahasiswa';

protected $fillable = ['nim', 'nama_mahasiswa'];

}

Buka file migration yang baru saja dibuat di folder database/migrations/..._create_mahasiswas_table.php, lalu sesuaikan kodenya:



public function up(): void

{

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

$table->id('id_mahasiswa');

$table->string('nim', 15)->unique();

$table->string('nama_mahasiswa', 100);

$table->timestamps();

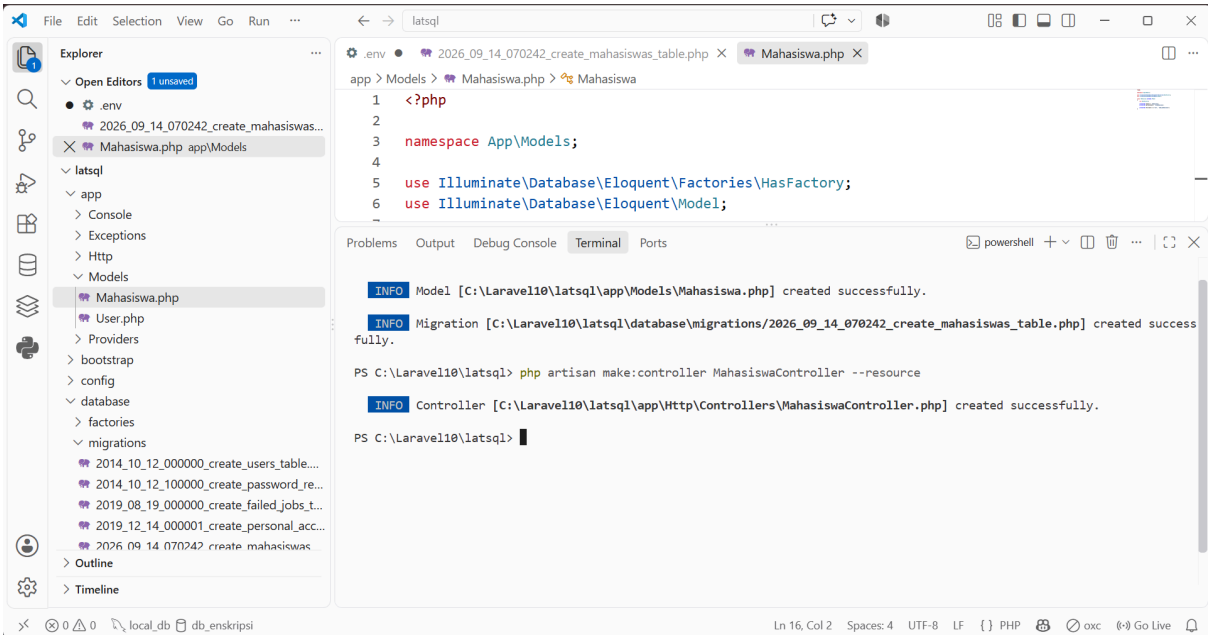
});

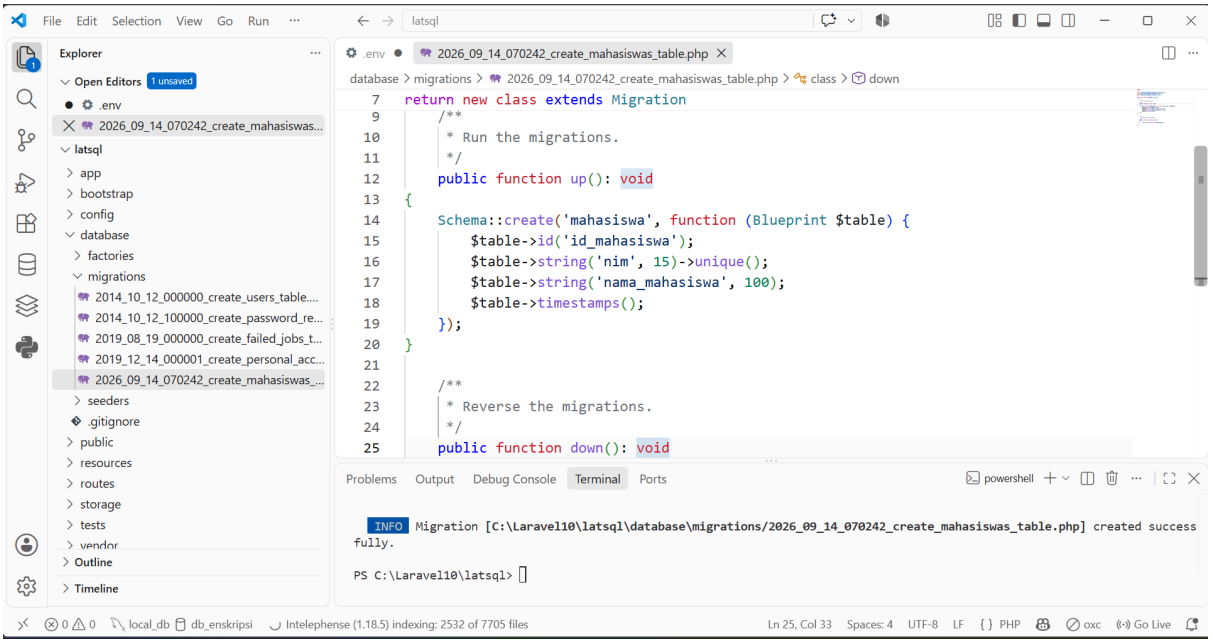
}

Langkah 3: Membuat Controller (Resource Controller)

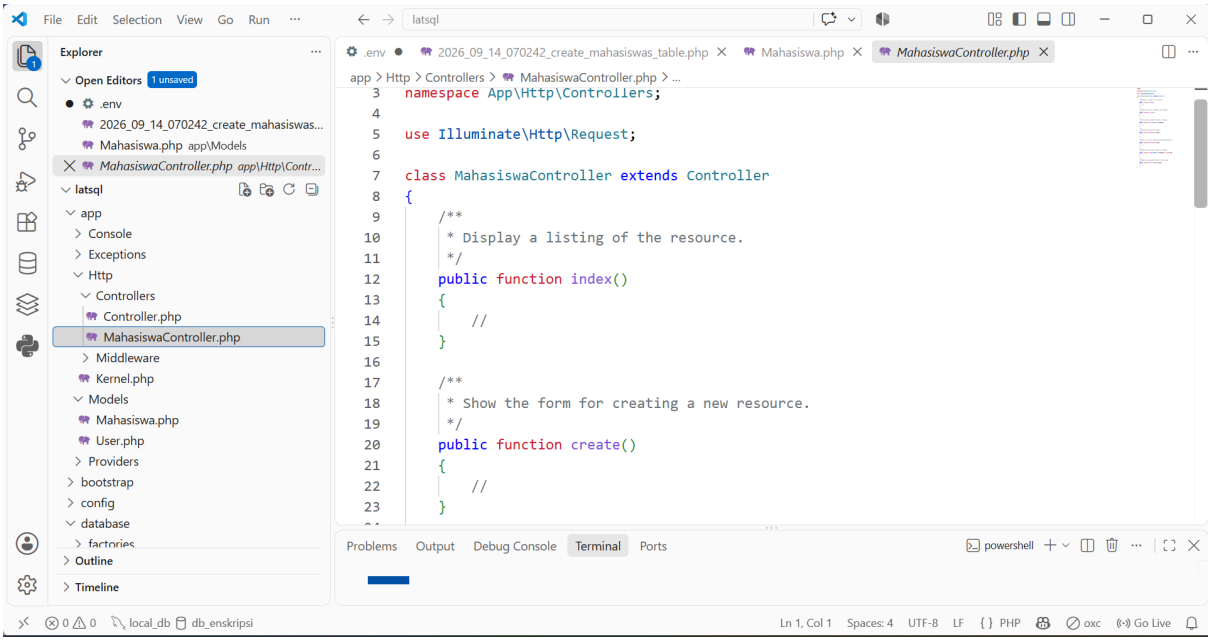
Buat Controller dengan opsi --resource agar otomatis menghasilkan fungsi CRUD (index, create, store, show, edit, update, destroy):

php artisan make:controller MahasiswaController –resource





Buka file app/Http/Controllers/MahasiswaController.php, lalu isi kodenya seperti berikut:



Ubah menjadi

namespace App\Http\Controllers;

use App\Models\Mahasiswa;

use Illuminate\Http\Request;

class MahasiswaController extends Controller

{

// READ: Menampilkan semua data

public function index()

{

$mahasiswa = Mahasiswa::all();

return view('mahasiswa.index', compact('mahasiswa'));

}

// CREATE: Menampilkan form tambah data

public function create()

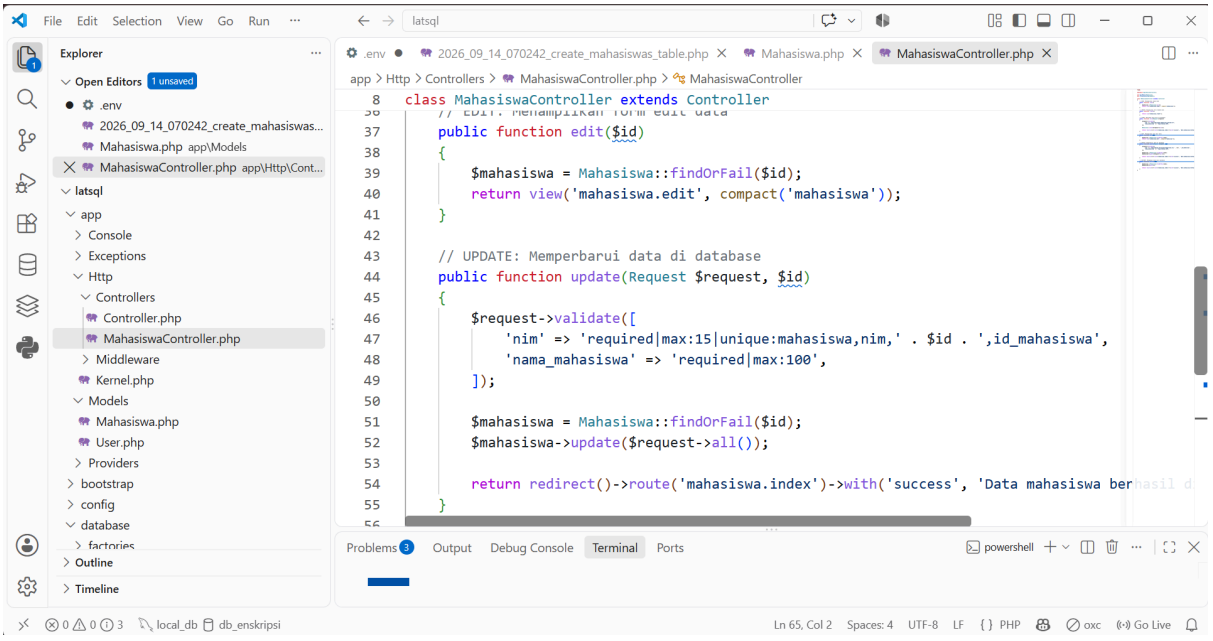
{

return view('mahasiswa.create');

}

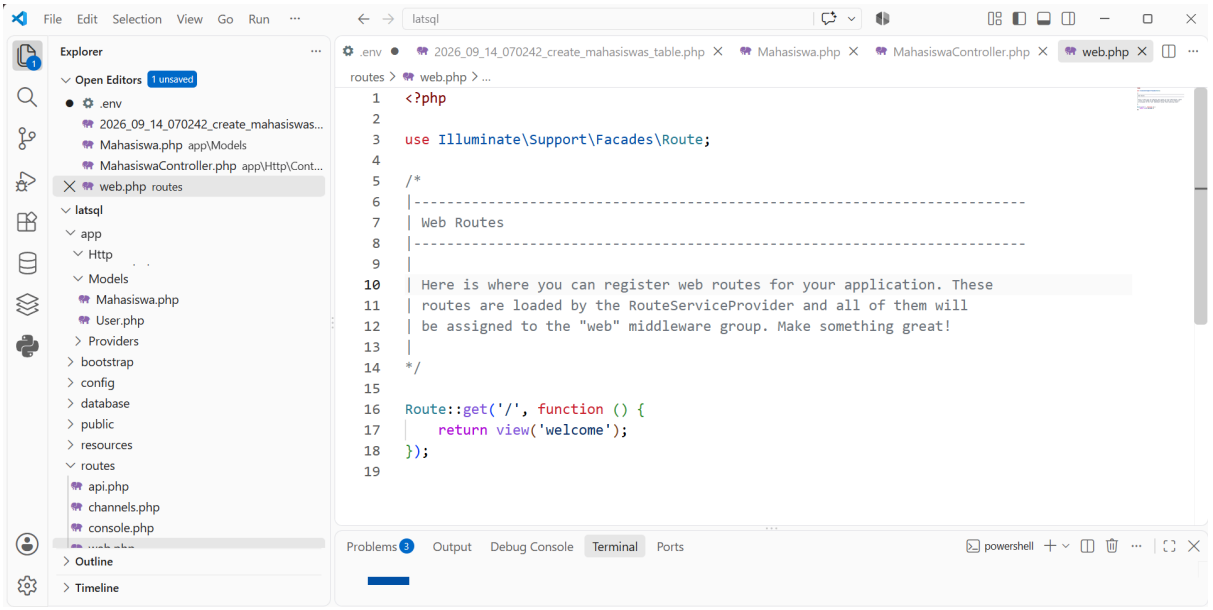
```
    // STORE: Menyimpan data baru ke database
    public function store(Request $request)
    {
        $request->validate([
            'nim' => 'required|unique:mahasiswa,nim|max:15',
            'nama_mahasiswa' => 'required|max:100',
        ]);

        Mahasiswa::create($request->all());

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

    // EDIT: Menampilkan form edit data
    public function edit($id)
    {
        $mahasiswa = Mahasiswa::findOrFail($id);
        return view('mahasiswa.edit', compact('mahasiswa'));
    }

    // UPDATE: Memperbarui data di database
    public function update(Request $request, $id)
    {
        $request->validate([
            'nim' => 'required|max:15|unique:mahasiswa,nim,' . $id . ',id_mahasiswa',
            'nama_mahasiswa' => 'required|max:100',
        ]);

        $mahasiswa = Mahasiswa::findOrFail($id);
```

```
        $mahasiswa->update($request->all());

        return redirect()->route('mahasiswa.index')->with('success', 'Data mahasiswa berhasil diperbarui.');
    }

    // DESTROY: Menghapus data dari database
    public function destroy($id)
    {
        $mahasiswa = Mahasiswa::findOrFail($id);
        $mahasiswa->delete();

        return redirect()->route('mahasiswa.index')->with('success', 'Data mahasiswa berhasil dihapus.');
    }
}
```



Langkah 4: Menambahkan Route

Buka file routes/web.php, lalu daftarkan resource route untuk controller tersebut:



Ganti

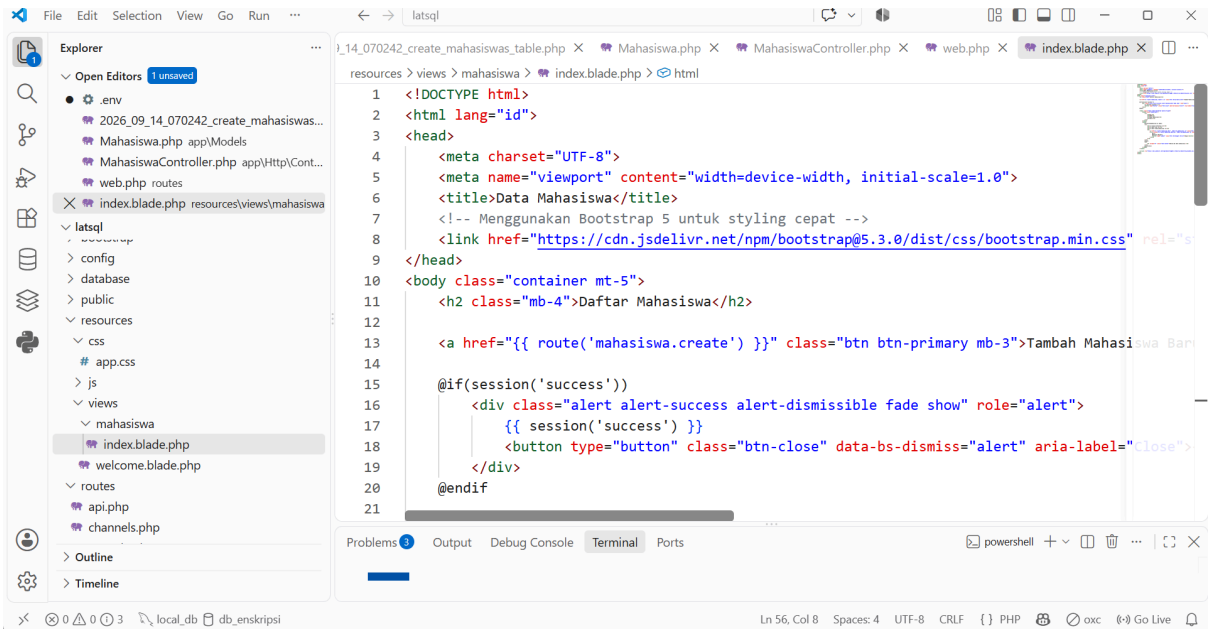
use App\Http\Controllers\MahasiswaController;

Route::resource('mahasiswa', MahasiswaController::class);

Langkah 5: Membuat Tampilan (Views Blade)

Buat folder baru bernama mahasiswa di dalam direktori resources/views/. Di dalam folder tersebut, buat tiga file utama:

1. **resources/views/mahasiswa/index.php** (Menampilkan tabel data & tombol hapus)

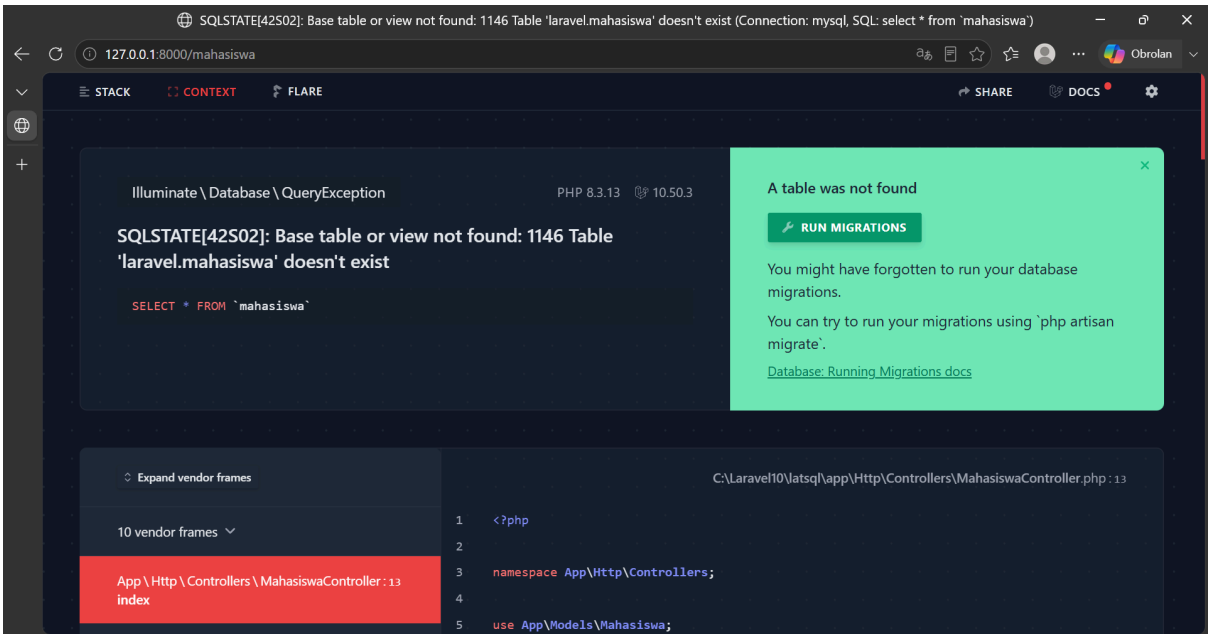


<!DOCTYPE html>

```
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Data Mahasiswa</title>
    <!-- Menggunakan Bootstrap 5 untuk styling cepat -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="container mt-5">
    <h2 class="mb-4">Daftar Mahasiswa</h2>

    <a href="{{ route('mahasiswa.create') }}" class="btn btn-primary mb-3">Tambah Mahasiswa Baru</a>

    @if(session('success'))
        <div class="alert alert-success alert-dismissible fade show" role="alert">
            {{ session('success') }}
            <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
        </div>
    @endif

    <table class="table table-bordered table-striped">
        <thead class="table-dark">
            <tr>
                <th>No</th>
                <th>NIM</th>
                <th>Nama Mahasiswa</th>
                <th>Aksi</th>
            </tr>
        </thead>
```

```
    <tbody>
      @forelse($mahasiswa as $mhs)
      <tr>
        <td>{{ $loop->iteration }}</td>
        <td>{{ $mhs->nim }}</td>
        <td>{{ $mhs->nama_mahasiswa }}</td>
        <td>
          <a href="{{ route('mahasiswa.edit', $mhs->id_mahasiswa) }}" class="btn btn-warning
btn-sm">Edit</a>
          <form action="{{ route('mahasiswa.destroy', $mhs->id_mahasiswa) }}"
method="POST" class="d-inline" onsubmit="return confirm('Apakah Anda yakin ingin
menghapus data ini?')">
            @csrf
            @method('DELETE')
            <button type="submit" class="btn btn-danger btn-sm">Hapus</button>
          </form>
        </td>
      </tr>
      @empty
      <tr>
        <td colspan="4" class="text-center">Belum ada data mahasiswa.</td>
      </tr>
      @endforelse
    </tbody>
  </table>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Jika Error



Langkah perbaikan:

laravel, bukan ke database **db_akademik** yang telah kita buat sebelumnya. Akibatnya, Laravel mencari tabel mahasiswa di dalam database laravel dan tidak menemukannya.

Untuk mengatasi masalah ini, Anda bisa memilih salah satu dari dua cara berikut:

Cara 1: Menghubungkan Laravel ke Database db_akademik (Paling Sesuai)

Jika Anda ingin Laravel menggunakan database db_akademik yang sudah berisi tabel-tabel relasi sebelumnya:

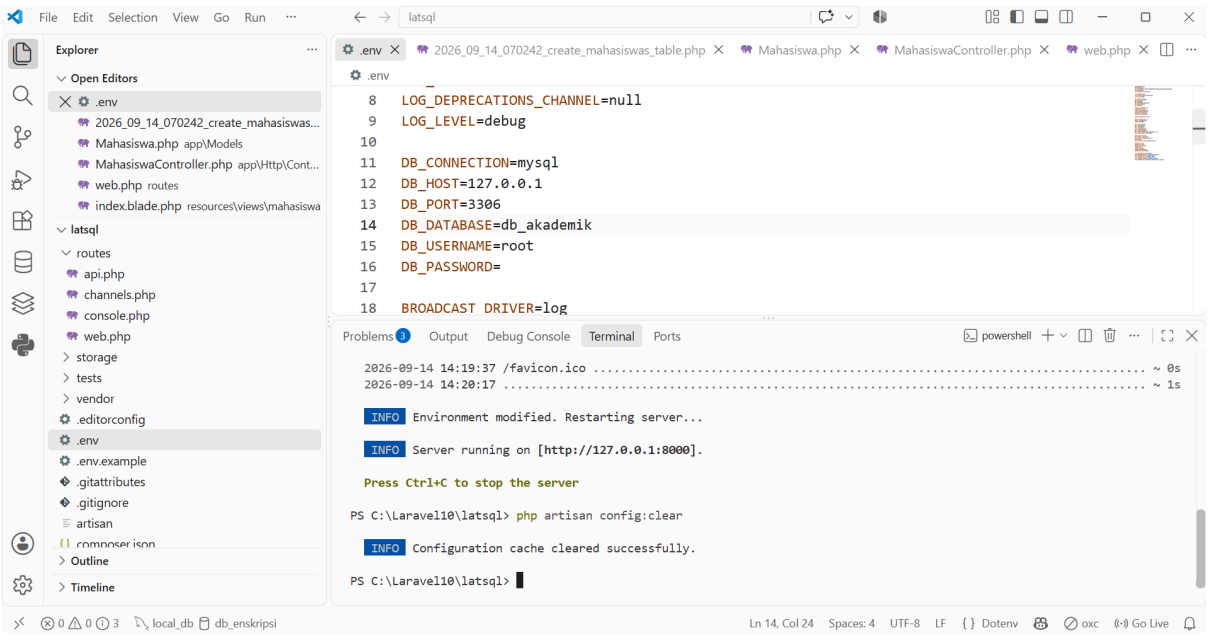
🎬 Buka file **.env** di folder utama project Laravel Anda.

🎬 Ubah bagian DB_DATABASE menjadi db_akademik:

🎬 Simpan file .env.

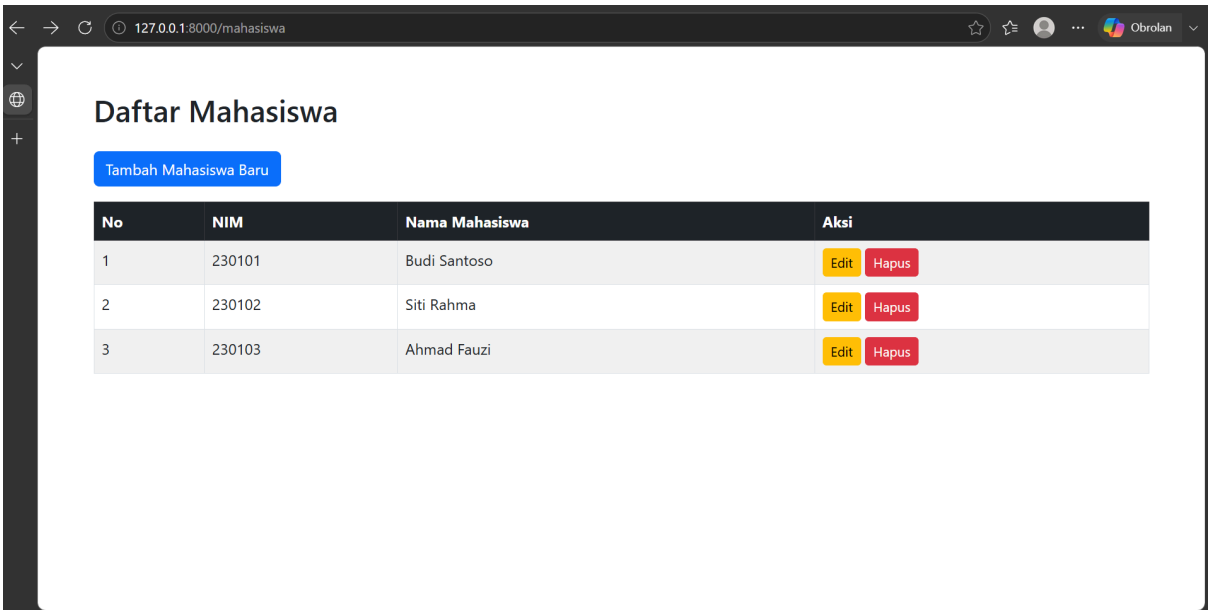
🎬 Buka terminal/command prompt, lalu bersihkan cache konfigurasi Laravel agar perubahan .env terbaca:

php artisan config:clear



Segarkan kembali halaman browser Anda (http://127.0.0.1:8000/mahasiswa).

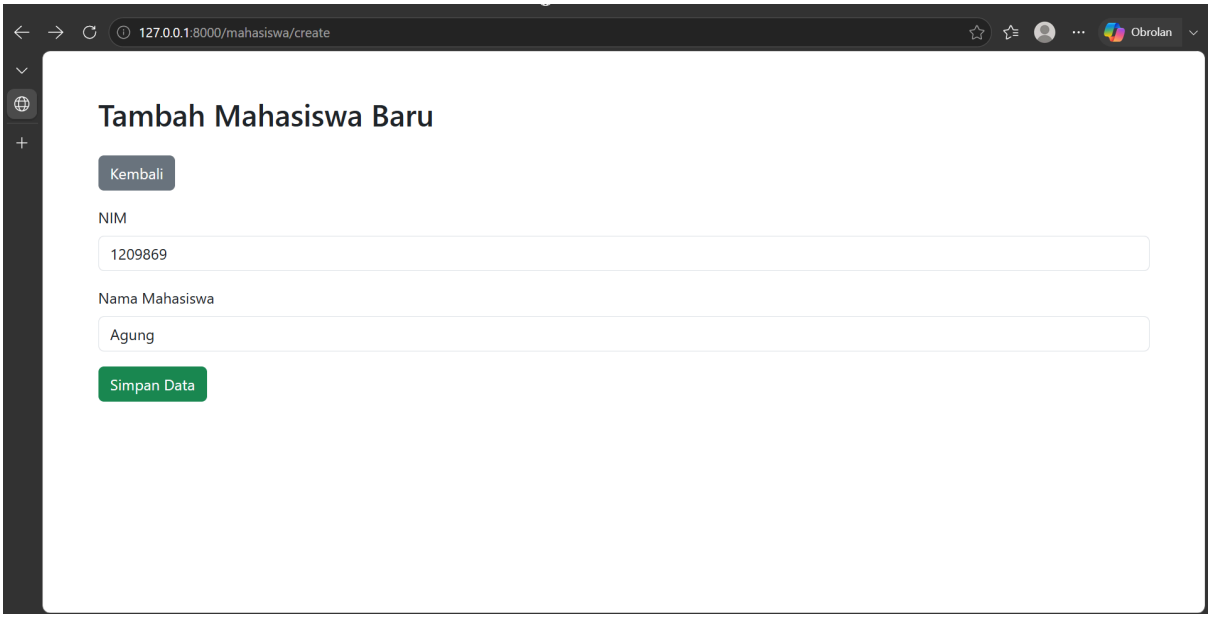
setelah menjalankan perintah php artisan serve

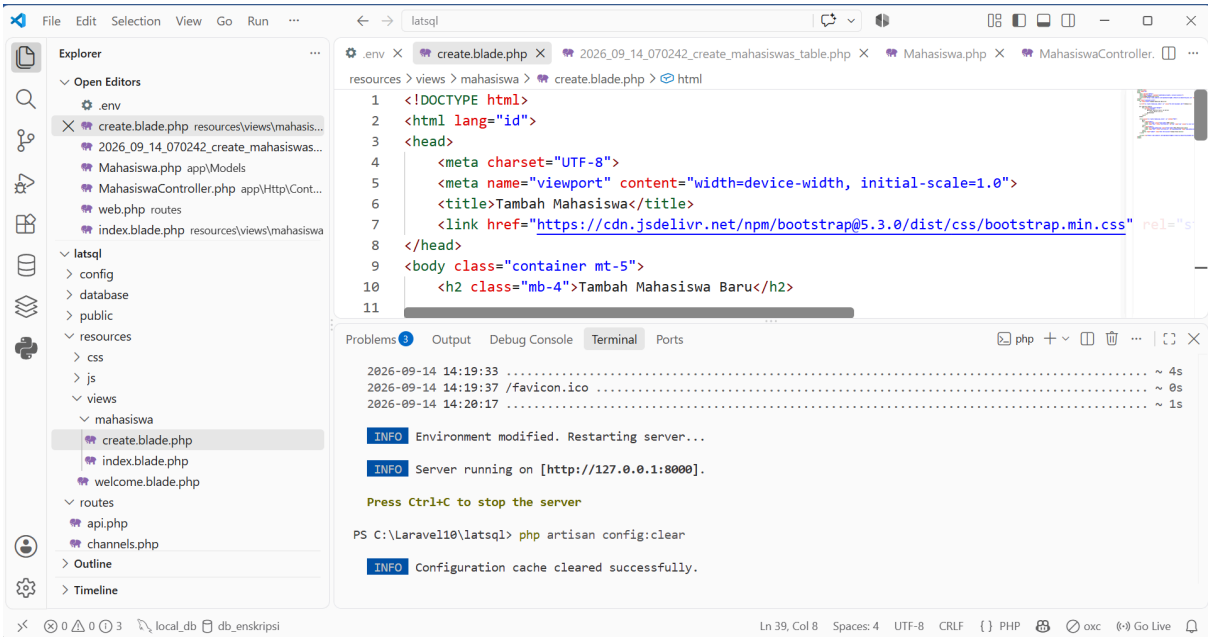


1. **resources/views/mahasiswa/create.php** (Form input data baru)

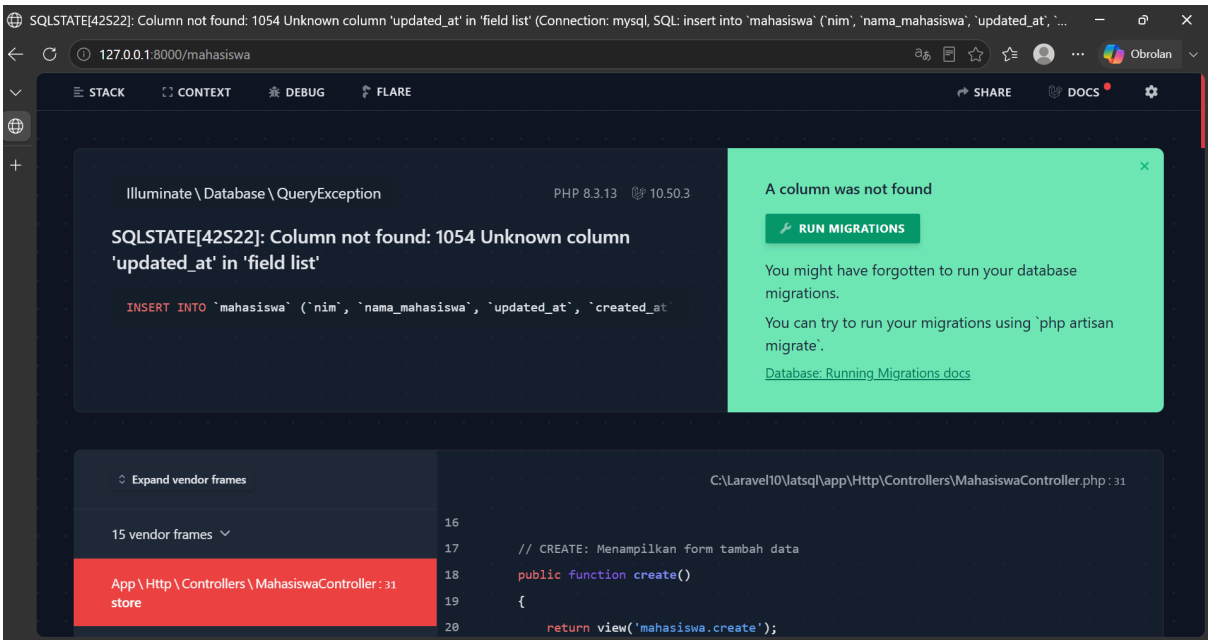
```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tambah Mahasiswa</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="container mt-5">
    <h2 class="mb-4">Tambah Mahasiswa Baru</h2>

    <a href="{{ route('mahasiswa.index') }}" class="btn btn-secondary mb-3">Kembali</a>

    @if ($errors->any())
        <div class="alert alert-danger">
            <ul class="mb-0">
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
    @endif

    <form action="{{ route('mahasiswa.store') }}" method="POST">
        @csrf
        <div class="mb-3">
            <label for="nim" class="form-label">NIM</label>
            <input type="text" class="form-control" id="nim" name="nim" value="{{ old('nim') }}" required>
        </div>
```

```
        <div class="mb-3">
            <label for="nama_mahasiswa" class="form-label">Nama Mahasiswa</label>
            <input type="text" class="form-control" id="nama_mahasiswa"
name="nama_mahasiswa" value="{{ old('nama_mahasiswa') }}" required>
        </div>
        <button type="submit" class="btn btn-success">Simpan Data</button>
    </form>


    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></scrip
t>
</body>
</html>
```





Jika error:



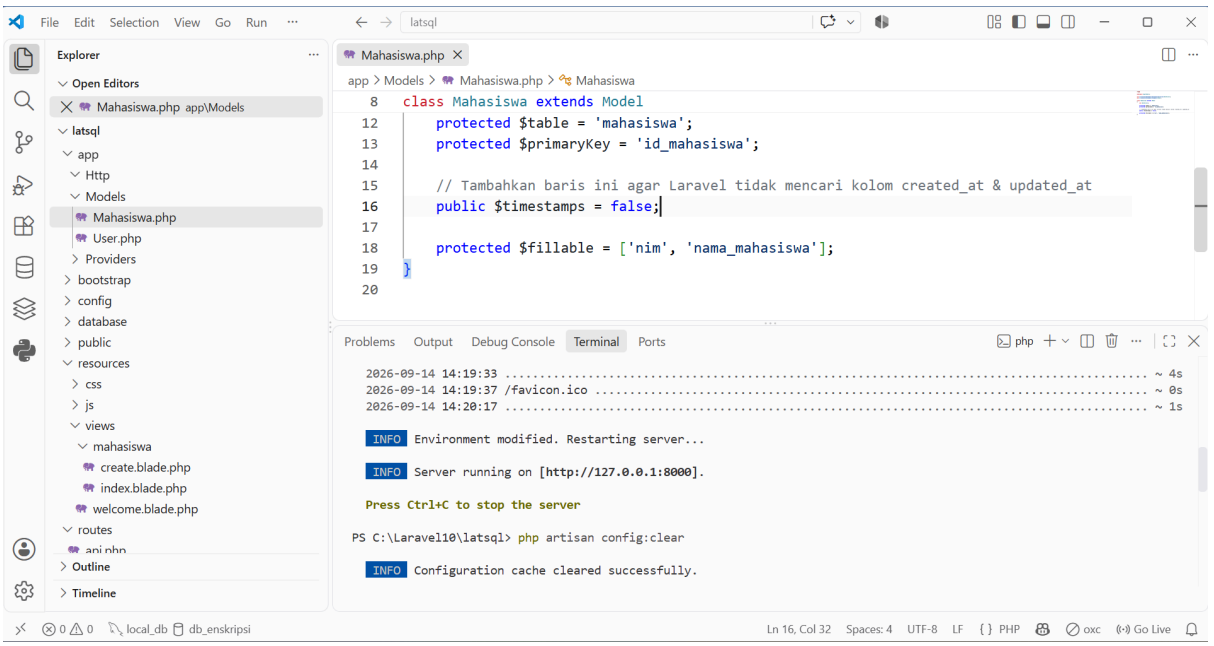
Error tersebut terjadi karena **Laravel secara otomatis mencoba mengisi kolom created_at dan updated_at** saat Anda melakukan proses *insert* data (simpan data baru). Namun, tabel mahasiswa yang kita buat sebelumnya secara manual melalui SQL **tidak memiliki kolom timestamp tersebut**.

Matikan Timestamps di Model (Paling Cepat)

Jika tabel mahasiswa Anda tidak memerlukan kolom waktu pencatatan otomatis (created_at dan updated_at), Anda bisa menonaktifkannya langsung di dalam file Model Laravel.

1. Buka file **app/Models/Mahasiswa.php**.
2. Tambahkan baris public $timestamps = false; di dalam kelasnya:

```php
namespace App\Models;

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

class Mahasiswa extends Model
{
    use HasFactory;

    protected $table = 'mahasiswa';
    protected $primaryKey = 'id_mahasiswa';

    // Tambahkan baris ini agar Laravel tidak mencari kolom created_at & updated_at
    public $timestamps = false;

    protected $fillable = ['nim', 'nama_mahasiswa'];
}
```



Cara 2: Tambahkan Kolom Timestamps ke Database

Jika Anda ingin tetap menggunakan fitur pencatatan waktu otomatis di Laravel, Anda bisa menambahkan kedua kolom tersebut langsung ke database db_akademik Anda melalui SQL Editor (phpMyAdmin/DBeaver/MySQL Workbench):

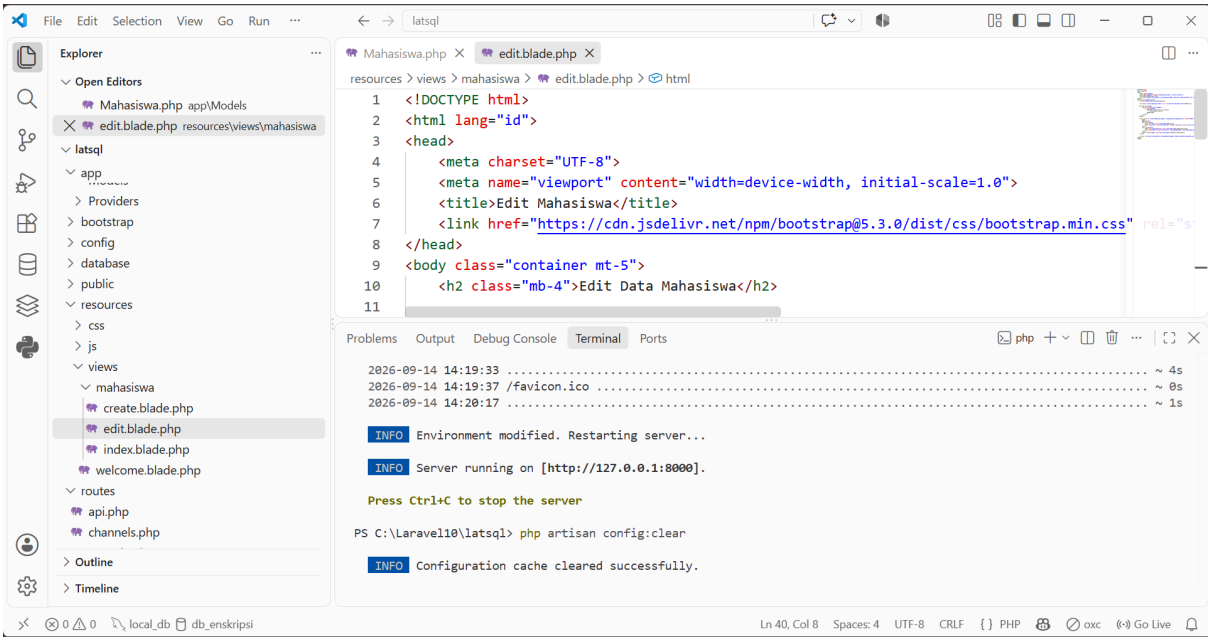
Jalankan query SQL berikut:

ALTER TABLE mahasiswa

ADD COLUMN created_at TIMESTAMP NULL,

ADD COLUMN updated_at TIMESTAMP NULL;

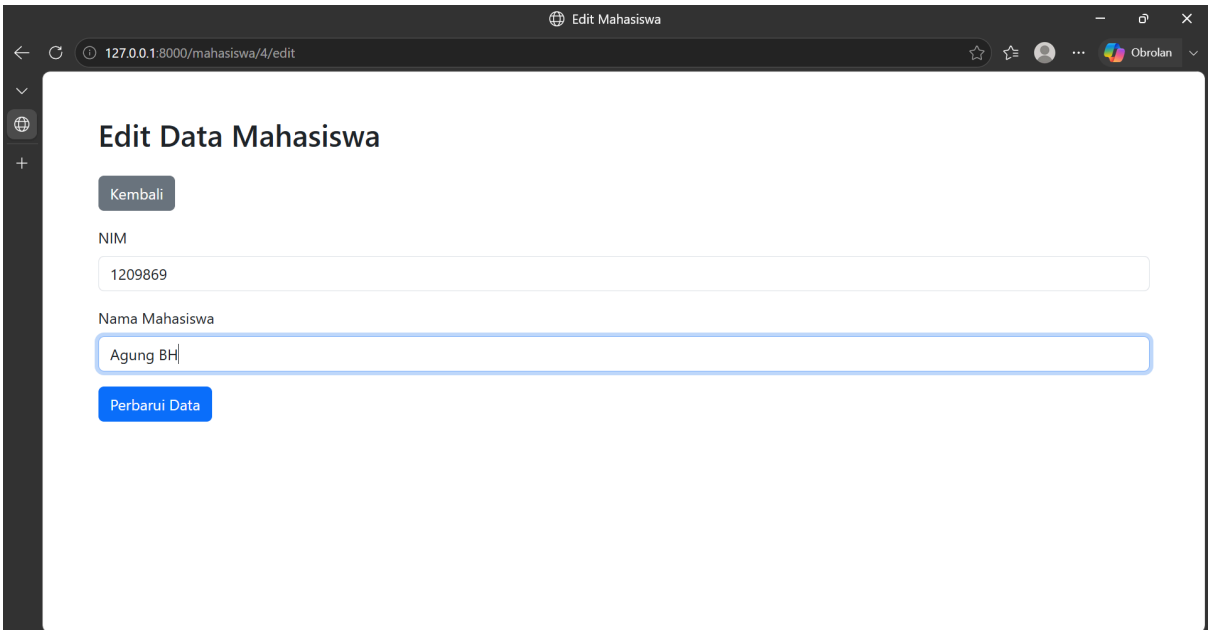
2. **resources/views/mahasiswa/edit.blade.php** (Form ubah data)




```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Mahasiswa</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="container mt-5">
    <h2 class="mb-4">Edit Data Mahasiswa</h2>
```

```
    <a href="{{ route('mahasiswa.index') }}" class="btn btn-secondary mb-3">Kembali</a>

    @if ($errors->any())
        <div class="alert alert-danger">
            <ul class="mb-0">
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
    @endif

    <form action="{{ route('mahasiswa.update', $mahasiswa->id_mahasiswa) }}" method="POST">
        @csrf
        @method('PUT')
        <div class="mb-3">
            <label for="nim" class="form-label">NIM</label>
            <input type="text" class="form-control" id="nim" name="nim" value="{{ old('nim',
$mahasiswa->nim) }}" required>
        </div>
        <div class="mb-3">
            <label for="nama_mahasiswa" class="form-label">Nama Mahasiswa</label>
            <input type="text" class="form-control" id="nama_mahasiswa" name="nama_mahasiswa"
value="{{ old('nama_mahasiswa', $mahasiswa->nama_mahasiswa) }}" required>
        </div>
        <button type="submit" class="btn btn-primary">Perbarui Data</button>
    </form>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
```

</html>



Anda dapat mengakses sistem CRUD ini melalui browser dengan URL: http://127.0.0.1:8000/mahasiswa setelah menjalankan perintah php artisan serve.

https://gemini.google.com/u/2/app/8971f985c2d9290b?pageId=none

- Lanjutkan:
  Mata Kuliah: Menyimpan daftar mata kuliah yang tersedia di kampus.
    - id_matkul (Primary Key)
    - kode_matkul
    - nama_matkul

Karena tabel mata_kuliah di database kita buat secara manual (tanpa kolom created_at dan updated_at), kita juga perlu menambahkan konfigurasi $timestamps = false di Model-nya nanti.

Langkah 1: Membuat Model MataKuliah

Ketik perintah ini di terminal:

php artisan make:model MataKuliah

Buka file **app/Models/MataKuliah.php**, lalu sesuaikan kodenya agar sesuai dengan struktur database mata_kuliah:

```
namespace App\Models;


use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;


class MataKuliah extends Model

{

   use HasFactory;


   protected $table = 'mata_kuliah';

   protected $primaryKey = 'id_matkul';

   public $timestamps = false; // Karena tabel manual tanpa created_at & updated_at


   protected $fillable = ['kode_matkul', 'nama_matkul'];

}
```

Langkah 2: Membuat Controller MataKuliahController

Ketik perintah ini di terminal:

```
php artisan make:controller MataKuliahController –resource
```

```
namespace App\Http\Controllers;

use App\Models\MataKuliah;
use Illuminate\Http\Request;

class MataKuliahController extends Controller
{
    public function index()
    {
        $mataKuliah = MataKuliah::all();
        return view('mata_kuliah.index', compact('mataKuliah'));
    }

    public function create()
    {
        return view('mata_kuliah.create');
    }

    public function store(Request $request)
    {
        $request->validate([
            'kode_matkul' => 'required|unique:mata_kuliah,kode_matkul|max:10',
            'nama_matkul' => 'required|max:100',
        ]);

        MataKuliah::create($request->all());

        return redirect()->route('mata-kuliah.index')->with('success', 'Data mata kuliah berhasil
ditambahkan.');
    }
```

```
    public function edit($id)
    {
        $mataKuliah = MataKuliah::findOrFail($id);
        return view('mata_kuliah.edit', compact('mataKuliah'));
    }

    public function update(Request $request, $id)
    {
        $request->validate([
            'kode_matkul' => 'required|max:10|unique:mata_kuliah,kode_matkul,' . $id . ',id_matkul',
            'nama_matkul' => 'required|max:100',
        ]);

        $mataKuliah = MataKuliah::findOrFail($id);
        $mataKuliah->update($request->all());

        return redirect()->route('mata-kuliah.index')->with('success', 'Data mata kuliah berhasil diperbarui.');
    }

    public function destroy($id)
    {
        $mataKuliah = MataKuliah::findOrFail($id);
        $mataKuliah->delete();

        return redirect()->route('mata-kuliah.index')->with('success', 'Data mata kuliah berhasil dihapus.');
    }
}
```

Langkah 3: Menambahkan Route

Buka file **routes/web.php**, lalu tambahkan route resource untuk mata kuliah:

```
use App\Http\Controllers\MataKuliahController;

Route::resource('mata-kuliah', MataKuliahController::class);
```

Langkah 4: Membuat Tampilan (Views Blade)

Buat folder baru bernama **mata_kuliah** di dalam direktori resources/views/ (resources/views/mata_kuliah/).

Di dalam folder tersebut, buat 3 file berikut:

1. resources/views/mata_kuliah/index.blade.php

```
<!DOCTYPE html>

<html lang="id">

<head>

   <meta charset="UTF-8">

   <meta name="viewport" content="width=device-width, initial-scale=1.0">

   <title>Data Mata Kuliah</title>

   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

</head>

<body class="container mt-5">

   <h2 class="mb-4">Daftar Mata Kuliah</h2>


   <a href="{{ route('mata-kuliah.create') }}" class="btn btn-primary mb-3">Tambah Mata Kuliah Baru</a>

   <a href="{{ route('mahasiswa.index') }}" class="btn btn-secondary mb-3 ms-2">Ke Data Mahasiswa</a>


   @if(session('success'))

      <div class="alert alert-success alert-dismissible fade show" role="alert">

         {{ session('success') }}

         <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>

      </div>

   @endif
```

```
<table class="table table-bordered table-striped">
    <thead class="table-dark">
        <tr>
            <th>No</th>
            <th>Kode Matkul</th>
            <th>Nama Mata Kuliah</th>
            <th>Aksi</th>
        </tr>
    </thead>
    <tbody>
        @forelse($mataKuliah as $mk)
        <tr>
            <td>{{ $loop->iteration }}</td>
            <td>{{ $mk->kode_matkul }}</td>
            <td>{{ $mk->nama_matkul }}</td>
            <td>
                <a href="{{ route('mata-kuliah.edit', $mk->id_matkul) }}" class="btn btn-warning btn-sm">Edit</a>
                <form action="{{ route('mata-kuliah.destroy', $mk->id_matkul) }}" method="POST" class="d-inline" onsubmit="return confirm('Apakah Anda yakin ingin menghapus data ini?')">
                    @csrf
                    @method('DELETE')
                    <button type="submit" class="btn btn-danger btn-sm">Hapus</button>
                </form>
            </td>
        </tr>
        @empty
        <tr>
            <td colspan="4" class="text-center">Belum ada data mata kuliah.</td>
        </tr>
        @endforelse
```

```
    </tbody>
  </table>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

2. resources/views/mata_kuliah/create.blade.php

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tambah Mata Kuliah</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="container mt-5">
  <h2 class="mb-4">Tambah Mata Kuliah Baru</h2>

  <a href="{{ route('mata-kuliah.index') }}" class="btn btn-secondary mb-3">Kembali</a>

  @if ($errors->any())
    <div class="alert alert-danger">
      <ul class="mb-0">
        @foreach ($errors->all() as $error)
          <li>{{ $error }}</li>
        @endforeach
      </ul>
    </div>
  @endif
```

```
    <form action="{{ route('mata-kuliah.store') }}" method="POST">
        @csrf
        <div class="mb-3">
            <label for="kode_matkul" class="form-label">Kode Mata Kuliah</label>
            <input type="text" class="form-control" id="kode_matkul" name="kode_matkul" value="{{ old('kode_matkul') }}" required>
        </div>
        <div class="mb-3">
            <label for="nama_matkul" class="form-label">Nama Mata Kuliah</label>
            <input type="text" class="form-control" id="nama_matkul" name="nama_matkul" value="{{ old('nama_matkul') }}" required>
        </div>
        <button type="submit" class="btn btn-success">Simpan Data</button>
    </form>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

3. resources/views/mata_kuliah/edit.blade.php

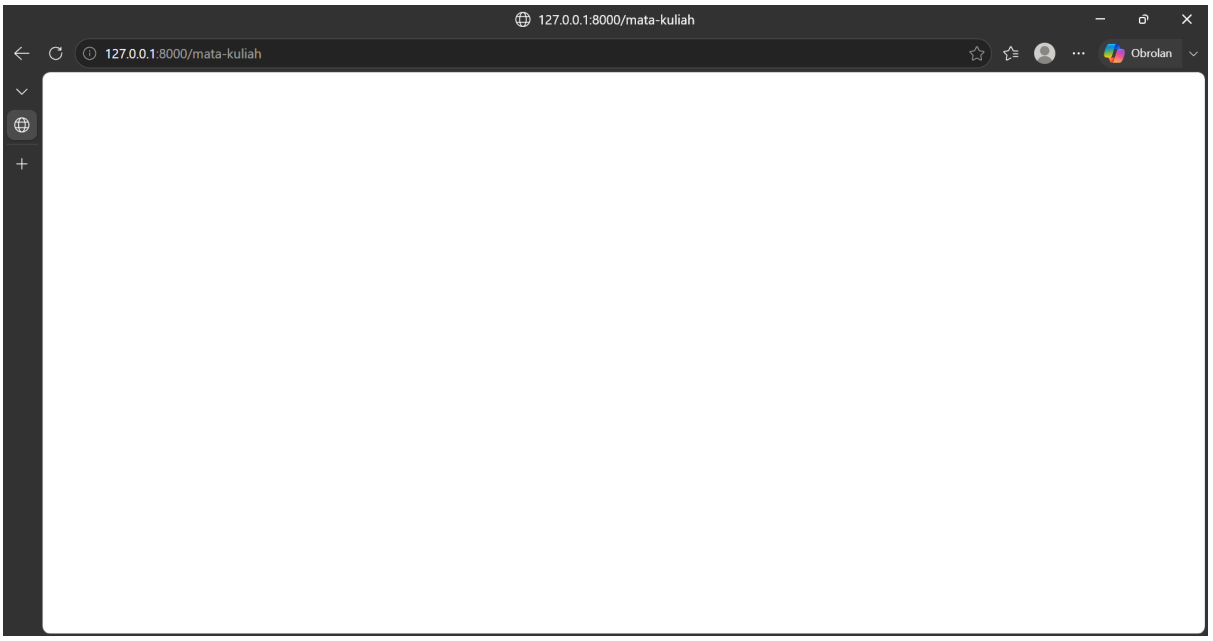
```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Mata Kuliah</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="container mt-5">
    <h2 class="mb-4">Edit Data Mata Kuliah</h2>
```

```
    <a href="{{ route('mata-kuliah.index') }}" class="btn btn-secondary mb-3">Kembali</a>

    @if ($errors->any())
        <div class="alert alert-danger">
            <ul class="mb-0">
                @foreach ($errors->all() as $error)
                    <li>{{ $error }}</li>
                @endforeach
            </ul>
        </div>
    @endif

    <form action="{{ route('mata-kuliah.update', $mataKuliah->id_matkul) }}" method="POST">
        @csrf
        @method('PUT')
        <div class="mb-3">
            <label for="kode_matkul" class="form-label">Kode Mata Kuliah</label>
            <input type="text" class="form-control" id="kode_matkul" name="kode_matkul" value="{{
old('kode_matkul', $mataKuliah->kode_matkul) }}" required>
        </div>
        <div class="mb-3">
            <label for="nama_matkul" class="form-label">Nama Mata Kuliah</label>
            <input type="text" class="form-control" id="nama_matkul" name="nama_matkul" value="{{
old('nama_matkul', $mataKuliah->nama_matkul) }}" required>
        </div>
        <button type="submit" class="btn btn-primary">Perbarui Data</button>
    </form>

    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Cara Menguji Coba:

1. Pastikan server lokal Anda menyala (php artisan serve).
2. Buka browser dan akses URL:
   http://127.0.0.1:8000/mata-kuliah

Kalau Error



2. Terjadi Cache View di Laravel

Laravel terkadang menyimpan *cache* dari tampilan sebelumnya sehingga tidak membaca file view yang baru.

- **Solusi:** Buka terminal Anda, lalu jalankan perintah berikut untuk membersihkan cache tampilan:

php artisan view:clear

php artisan config:clear

3. Periksa Log Error Laravel

Jika setelah dibersihkan halamannya masih tetap putih, coba periksa apakah ada error tersembunyi di dalam file log:

- Buka folder project Anda di storage/logs/laravel.log.
- Lihat bagian baris paling bawah untuk membaca detail error jika ada kendala pada script Controller atau Model.

Mari kita lanjutkan ke bagian terakhir dari sistem relasi database ini, yaitu **CRUD untuk Tabel KRS (Kartu Rencana Studi)** sebagai tabel perantara (*junction table*) yang menghubungkan Mahasiswa dan Mata Kuliah.

Pada tabel KRS ini, proses *Create* dan *Index* akan menggunakan relasi antar model untuk menampilkan pilihan nama mahasiswa dan nama mata kuliah melalui elemen dropdown (<select>).

Langkah 1: Membuat Model Krs

Ketik perintah ini di terminal:

Bash

```
php artisan make:model Krs
```

Buka file **app/Models/Krs.php**, lalu isi kodenya untuk mendefinisikan relasi ke tabel mahasiswa dan mata_kuliah:

PHP

```
namespace App\Models;


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


class Krs extends Model
{
    use HasFactory;


    protected $table = 'krs';
    protected $primaryKey = 'id_krs';
    public $timestamps = false; // Tabel manual tanpa created_at & updated_at


    protected $fillable = ['id_mahasiswa', 'id_matkul'];


    // Relasi ke tabel Mahasiswa
    public function mahasiswa()
    {
        return $this->belongsTo(Mahasiswa::class, 'id_mahasiswa', 'id_mahasiswa');
    }
```

```
    // Relasi ke tabel Mata Kuliah
    public function mataKuliah()
    {
        return $this->belongsTo(MataKuliah::class, 'id_matkul', 'id_matkul');
    }
}
```

Langkah 2: Membuat Controller KrsController

Ketik perintah ini di terminal:

Bash

```
php artisan make:controller KrsController --resource
```

Buka file **app/Http/Controllers/KrsController.php**, lalu sesuaikan kodenya (kita fokuskan pada fungsi index, create, store, dan destroy karena KRS umumnya tidak memerlukan proses edit, cukup tambah atau hapus):

PHP

```
namespace App\Http\Controllers;

use App\Models\Krs;
use App\Models\Mahasiswa;
use App\Models\MataKuliah;
use Illuminate\Http\Request;

class KrsController extends Controller
{
    // Menampilkan daftar KRS beserta data relasinya
    public function index()
    {
        $krs = Krs::with(['mahasiswa', 'mataKuliah'])->get();
        return view('krs.index', compact('krs'));
    }

    // Menampilkan form tambah KRS dengan pilihan dropdown mahasiswa & matkul
```

```
    public function create()
    {
        $mahasiswa = Mahasiswa::all();
        $mataKuliah = MataKuliah::all();
        return view('krs.create', compact('mahasiswa', 'mataKuliah'));
    }

    // Menyimpan data KRS baru
    public function store(Request $request)
    {
        $request->validate([
            'id_mahasiswa' => 'required',
            'id_matkul' => 'required',
        ]);

        Krs::create($request->all());

        return redirect()->route('krs.index')->with('success', 'Mata kuliah berhasil diambil oleh mahasiswa.');
    }

    // Menghapus data KRS
    public function destroy($id)
    {
        $krs = Krs::findOrFail($id);
        $krs->delete();

        return redirect()->route('krs.index')->with('success', 'Data KRS berhasil dihapus.');
    }
}
```

Langkah 3: Menambahkan Route

Buka file **routes/web.php**, lalu tambahkan route resource untuk KRS:

PHP

```
use App\Http\Controllers\KrsController;


Route::resource('krs', KrsController::class);
```

Langkah 4: Membuat Tampilan (Views Blade)

Buat folder baru bernama **krs** di dalam direktori resources/views/ (resources/views/krs/). Di dalamnya, buat 2 file tampilan berikut:

1. resources/views/krs/index.blade.php

HTML

```
<!DOCTYPE html>

<html lang="id">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Data KRS (Kartu Rencana Studi)</title>

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">

</head>

<body class="container mt-5">

    <h2 class="mb-4">Daftar Pengambilan KRS Mahasiswa</h2>


    <a href="{{ route('krs.create') }}" class="btn btn-primary mb-3">Tambah KRS Baru</a>

    <a href="{{ route('mahasiswa.index') }}" class="btn btn-secondary mb-3 ms-2">Menu
Mahasiswa</a>

    <a href="{{ route('mata-kuliah.index') }}" class="btn btn-secondary mb-3 ms-2">Menu Mata
Kuliah</a>


    @if(session('success'))

        <div class="alert alert-success alert-dismissible fade show" role="alert">

            {{ session('success') }}

            <button type="button" class="btn-close" data-bs-dismiss="alert"
aria-label="Close"></button>
```

```
    </div>
  @endif

  <table class="table table-bordered table-striped">
    <thead class="table-dark">
      <tr>
        <th>No</th>
        <th>NIM Mahasiswa</th>
        <th>Nama Mahasiswa</th>
        <th>Kode Matkul</th>
        <th>Nama Mata Kuliah</th>
        <th>Aksi</th>
      </tr>
    </thead>
    <tbody>
      @forelse($krs as $item)
      <tr>
        <td>{{ $loop->iteration }}</td>
        <td>{{ $item->mahasiswa->nim ?? '-' }}</td>
        <td>{{ $item->mahasiswa->nama_mahasiswa ?? '-' }}</td>
        <td>{{ $item->mataKuliah->kode_matkul ?? '-' }}</td>
        <td>{{ $item->mataKuliah->nama_matkul ?? '-' }}</td>
        <td>
          <form action="{{ route('krs.destroy', $item->id_krs) }}" method="POST" class="d-inline"
onsubmit="return confirm('Batalkan pengambilan mata kuliah ini?')">
            @csrf
            @method('DELETE')
            <button type="submit" class="btn btn-danger btn-sm">Hapus</button>
          </form>
        </td>
      </tr>
```

```
        @empty
        <tr>
            <td colspan="6" class="text-center">Belum ada data KRS yang tercatat.</td>
        </tr>
        @endforelse
    </tbody>
  </table>

  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

2. resources/views/krs/create.blade.php

HTML

```
<!DOCTYPE html>
<html lang="id">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tambah KRS</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body class="container mt-5">
  <h2 class="mb-4">Form Pengambilan KRS</h2>

  <a href="{{ route('krs.index') }}" class="btn btn-secondary mb-3">Kembali</a>

  @if ($errors->any())
    <div class="alert alert-danger">
      <ul class="mb-0">
        @foreach ($errors->all() as $error)
```

```
                <li>{{ $error }}</li>
            @endforeach
        </ul>
    </div>
@endif

<form action="{{ route('krs.store') }}" method="POST">
    @csrf
    <div class="mb-3">
        <label for="id_mahasiswa" class="form-label">Pilih Mahasiswa</label>
        <select class="form-select" id="id_mahasiswa" name="id_mahasiswa" required>
            <option value="">-- Pilih Mahasiswa --</option>
            @foreach($mahasiswa as $mhs)
                <option value="{{ $mhs->id_mahasiswa }}">{{ $mhs->nim }} - {{ $mhs->nama_mahasiswa
}}</option>
            @endforeach
        </select>
    </div>

    <div class="mb-3">
        <label for="id_matkul" class="form-label">Pilih Mata Kuliah</label>
        <select class="form-select" id="id_matkul" name="id_matkul" required>
            <option value="">-- Pilih Mata Kuliah --</option>
            @foreach($mataKuliah as $mk)
                <option value="{{ $mk->id_matkul }}">{{ $mk->kode_matkul }} - {{ $mk->nama_matkul
}}</option>
            @endforeach
        </select>
    </div>

    <button type="submit" class="btn btn-success">Simpan KRS</button>
</form>
```

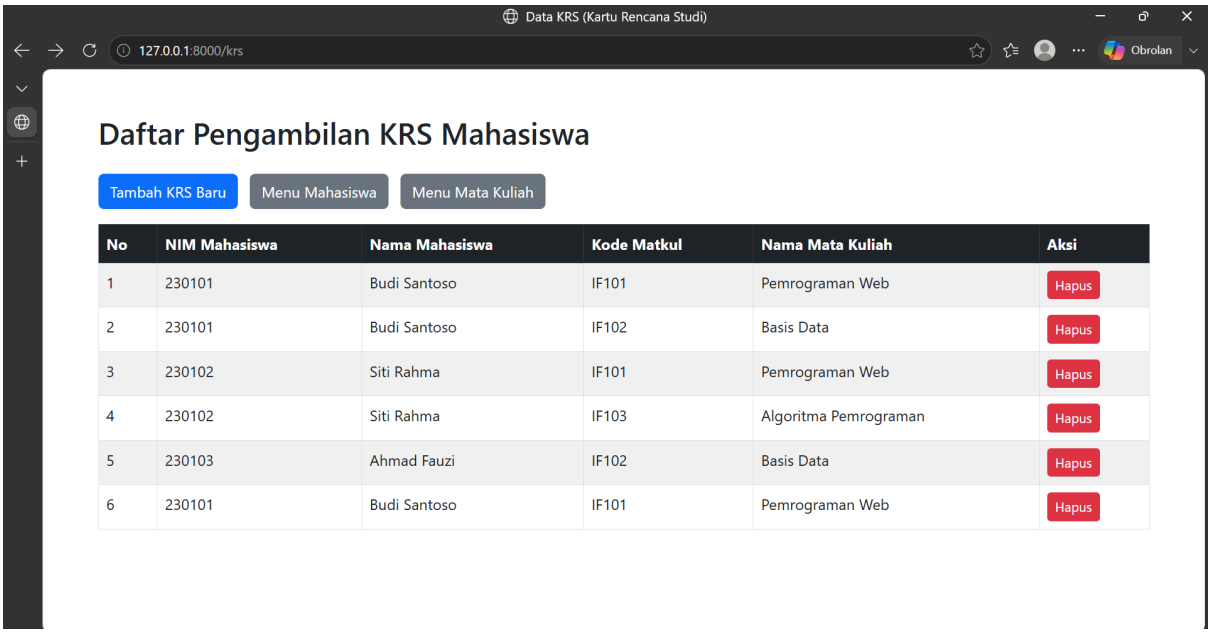
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>

Cara Menguji Coba:

1. Pastikan server lokal menyala (php artisan serve).
2. Buka browser dan akses halaman KRS di URL: http://127.0.0.1:8000/krs
3. Anda sekarang dapat menghubungkan mahasiswa dengan mata kuliah pilihannya secara dinamis menggunakan data dari database!



Untuk mencegah mahasiswa mengambil mata kuliah yang sama lebih dari sekali, kita perlu menambahkan aturan validasi unik gabungan (unique dengan kondisi where) pada KrsController.

Berikut adalah cara menerapkannya:

Perbarui KrsController.php

Buka file **app/Http/Controllers/KrsController.php**, lalu ubah fungsi **store** di dalamnya menjadi seperti ini:

PHP

namespace App\Http\Controllers;

use App\Models\Krs;

use App\Models\Mahasiswa;

```
use App\Models\MataKuliah;
use Illuminate\Http\Request;
use Illuminate\Validation\Rule; // <-- Pastikan class Rule ini diimport di bagian atas

class KrsController extends Controller
{
    public function index()
    {
        $krs = Krs::with(['mahasiswa', 'mataKuliah'])->get();
        return view('krs.index', compact('krs'));
    }

    public function create()
    {
        $mahasiswa = Mahasiswa::all();
        $mataKuliah = MataKuliah::all();
        return view('krs.create', compact('mahasiswa', 'mataKuliah'));
    }

    public function store(Request $request)
    {
        // Validasi dengan pengecekan agar mahasiswa tidak bisa mengambil matkul yang sama
        $request->validate([
            'id_mahasiswa' => 'required',
            'id_matkul' => [
                'required',
                Rule::unique('krs')->where(function ($query) use ($request) {
                    return $query->where('id_mahasiswa', $request->id_mahasiswa);
                }),
            ],
        ], [
```

```
            // Pesan error kustom agar mudah dipahami
            'id_matkul.unique' => 'Mata kuliah ini sudah pernah diambil oleh mahasiswa tersebut!',
        ]);

        Krs::create($request->all());

        return redirect()->route('krs.index')->with('success', 'Mata kuliah berhasil diambil oleh mahasiswa.');
    }

    public function destroy($id)
    {
        $krs = Krs::findOrFail($id);
        $krs->delete();

        return redirect()->route('krs.index')->with('success', 'Data KRS berhasil dihapus.');
    }
}
```

Penjelasan:

- **Rule::unique('krs')->where(...)**: Berfungsi untuk mengecek ke dalam tabel krs, apakah kombinasi antara id_mahasiswa dan id_matkul yang dipilih sudah ada sebelumnya atau belum.
- Jika kombinasi tersebut sudah tercatat di database, Laravel akan menolak penyimpanan dan mengembalikan pesan error yang sudah kita kustomisasi: *"Mata kuliah ini sudah pernah diambil oleh mahasiswa tersebut!"*.

Untuk menyatukan semua menu (**Mahasiswa**, **Mata Kuliah**, dan **KRS**) ke dalam satu halaman utama yang rapi, kita bisa membuat halaman **Beranda (Dashboard)**. Halaman ini akan menjadi halaman pertama yang muncul saat aplikasi diakses.

Berikut adalah langkah-langkah pembuatannya:

Langkah 1: Tambahkan Route untuk Beranda

Buka file **routes/web.php**, lalu ubah route utama (/) agar langsung mengarah ke tampilan beranda:

PHP

```
use Illuminate\Support\Facades\Route;
```

```
// Halaman Beranda (Dashboard Utama)
Route::get('/', function () {
    return view('beranda');
})->name('beranda');
```

Langkah 2: Buat Tampilan Beranda (resources/views/beranda.blade.php)

Buat file baru bernama **beranda.blade.php** di dalam folder resources/views/ (resources/views/beranda.blade.php), lalu isi dengan kode Bootstrap 5 berikut:

HTML

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Beranda - Sistem Akademik</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light">

    <div class="container py-5">
        <!-- Header Judul -->
        <div class="text-center mb-5">
            <h1 class="display-5 fw-bold text-dark">Sistem Informasi Akademik</h1>
            <p class="text-muted">Pilih menu di bawah ini untuk mulai mengelola data akademik.</p>
        </div>

        <!-- Kartu Menu Navigasi -->
        <div class="row justify-content-center g-4">

            <!-- Menu Mahasiswa -->
            <div class="col-md-4">
```

```
            <div class="card shadow-sm h-100 border-0">
                <div class="card-body text-center p-4">
                    <h3 class="card-title fw-bold text-primary mb-3">Mahasiswa</h3>
                    <p class="card-text text-muted">Kelola data mahasiswa, tambah NIM, nama, dan informasi pribadi lainnya.</p>
                    <a href="{{ route('mahasiswa.index') }}" class="btn btn-primary w-100 mt-3">Buka Mahasiswa</a>
                </div>
            </div>
        </div>

        <!-- Menu Mata Kuliah -->
        <div class="col-md-4">
            <div class="card shadow-sm h-100 border-0">
                <div class="card-body text-center p-4">
                    <h3 class="card-title fw-bold text-success mb-3">Mata Kuliah</h3>
                    <p class="card-text text-muted">Kelola daftar mata kuliah, kode matkul, dan daftar pengajaran.</p>
                    <a href="{{ route('mata-kuliah.index') }}" class="btn btn-success w-100 mt-3">Buka Mata Kuliah</a>
                </div>
            </div>
        </div>

        <!-- Menu KRS -->
        <div class="col-md-4">
            <div class="card shadow-sm h-100 border-0">
                <div class="card-body text-center p-4">
                    <h3 class="card-title fw-bold text-warning mb-3">KRS</h3>
                    <p class="card-text text-muted">Atur kartu rencana studi untuk menghubungkan mahasiswa dan mata kuliah.</p>
                    <a href="{{ route('krs.index') }}" class="btn btn-warning text-white w-100 mt-3">Buka KRS</a>
```

```
            </div>
          </div>
        </div>


    </div>
  </div>


  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>

</body>

</html>
```

Langkah 3: Tambahkan Tombol "Beranda" di Menu Lainnya

Agar Anda bisa kembali ke halaman Beranda dari menu Mahasiswa, Mata Kuliah, maupun KRS, tambahkan tombol navigasi Beranda di bagian atas file-file index masing-masing:

- **Di resources/views/mahasiswa/index.blade.php**: Tambahkan tombol ini di sebelah tombol tambah data:

HTML

```
<a href="{{ route('beranda') }}" class="btn btn-secondary mb-3">Beranda Utama</a>
```

- **Di resources/views/mata_kuliah/index.blade.php**:

HTML

```
<a href="{{ route('beranda') }}" class="btn btn-secondary mb-3">Beranda Utama</a>
```

- **Di resources/views/krs/index.blade.php** (yang sudah kita buat sebelumnya, tombolnya sudah terhubung dengan rapi!).

Cara Menguji:

1. Jalankan server Laravel Anda: php artisan serve
2. Buka browser di alamat utama: **http://127.0.0.1:8000/**
3. Anda akan disambut dengan tampilan Beranda modern yang memiliki tombol akses cepat ke menu Mahasiswa, Mata Kuliah, dan KRS!

https://gemini.google.com/u/2/app/8971f985c2d9290b?pageId=none