

Tabel Progres

Topik	Sub Topik	Status
Menginstal Laravel 11		
Konfigurasi Proyek Laravel		
Membuat Model dan Migration		
Menambahkan Kolom di Dalam Migration		
	1. Migration Kasus	
	2. Migration Siswa	
	3. Migration Kelas	
	4. Migration Walikelas	
Menambahkan Mass Assignment		
	1. Model Kasus	
	2. Model Siswa	
	3. Model Kelas	
	4. Model Walikelas	
Menjalankan Proses Migrate		
Menerapkan Autentikasi		
Membuat Controller Auth		
Menambahkan Fungsi di AuthController		
	Menampilkan Halaman Jenis Login	
	Menampilkan Form Login Berdasarkan Jenis	
	Proses Login	
	Proses Logout	
	Membuat Routing untuk Autentikasi	
Membuat View untuk Menampilkan Halaman Login		
	Halaman Pilihan Login	
	Membuat Form Login	
Membuat Middleware		
	Membuat Middleware untuk Autentikasi	

Topik	Sub Topik	Status
	Membuat Middleware untuk Role	
	Registrasi Middleware	
Membuat Blade Layout		
Membuat Tampilan Homepage		
Membuat Tampilan Dashboard Setelah Login		
Membuat Fitur Kasus	<ul style="list-style-type: none"> Menambahkan Routing untuk Kasus Membuat Controller Kasus Menampilkan Daftar Kasus (Method index) Membuat View untuk Menampilkan Kasus (View index) Menambahkan Kasus Baru ke Dalam Database (Method create dan store) Membuat Form untuk Menambah Kasus (View create) Edit dan Update Kasus ke Database Membuat View Form Edit Kasus (View edit) Menambahkan Method Destroy di Controller Menambahkan Tombol Hapus di View 	
Membuat Fitur Siswa	<ul style="list-style-type: none"> Menambahkan Routing untuk Siswa Membuat Controller Siswa Menampilkan Daftar Siswa (Method index) Membuat View untuk Menampilkan Siswa (View index) Menambahkan Siswa Baru ke Dalam Database (Method create dan store) Membuat Form untuk Menambah Siswa (View create) Menambahkan Method destroy di Controller Menambahkan Tombol Hapus di View 	
Membuat Fitur Walikelas	<ul style="list-style-type: none"> Menambahkan Routing untuk Walikelas Membuat Controller Walikelas 	

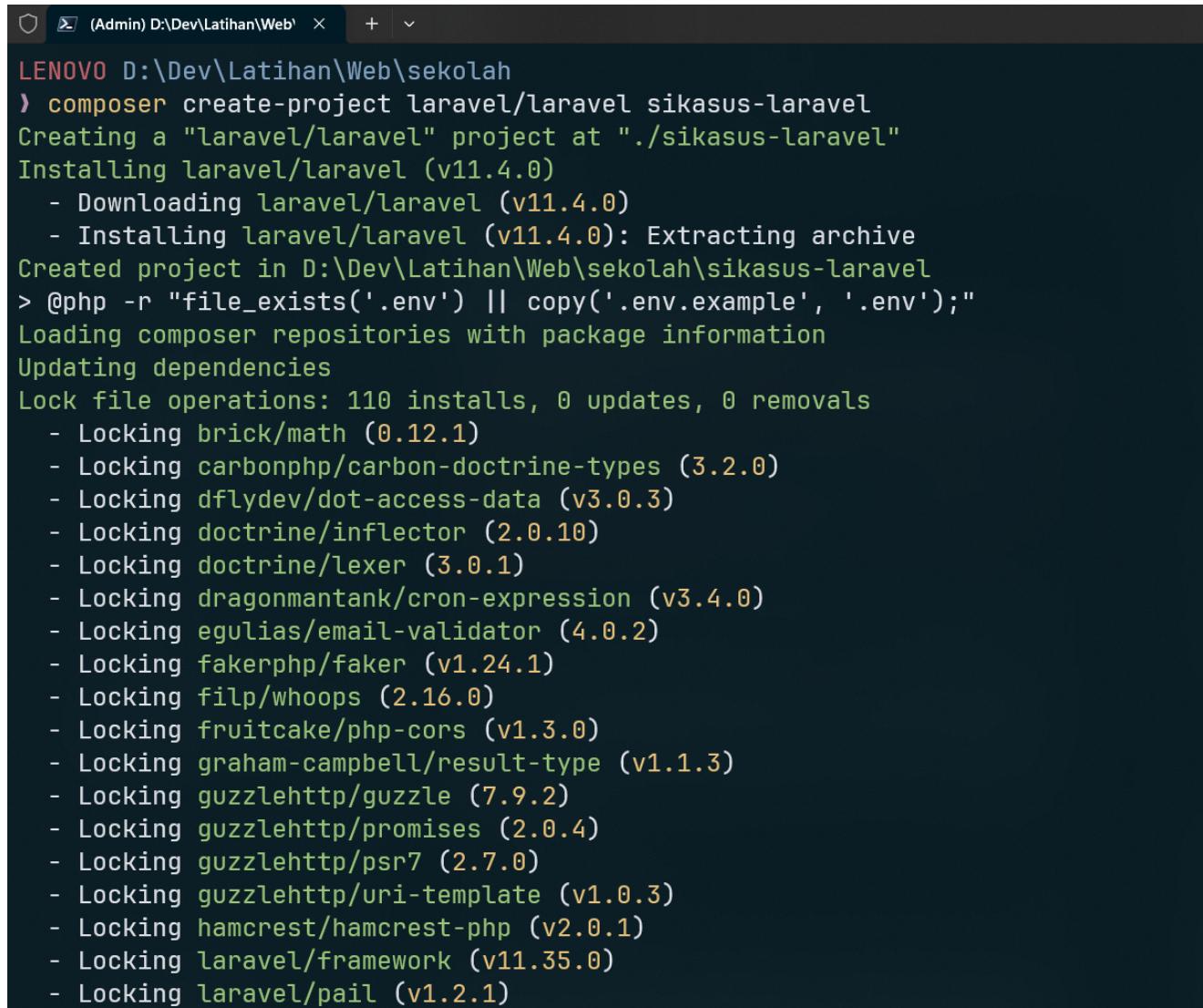
Topik	Sub Topik	Status
	Menampilkan Daftar Walikelas (Method index)	
	Membuat View untuk Menampilkan Walikelas (View index)	
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk `Siswa	
	Membuat Controller `Siswa	

Menginstal Laravel 11

Langkah pertama adalah memastikan bahwa **Composer** telah terinstal di komputer. **Composer** digunakan untuk mengelola dependensi dalam proyek **Laravel**.

Kemudian, buka terminal atau command prompt, lalu jalankan perintah berikut untuk membuat proyek Laravel baru:

```
composer create-project laravel/laravel sikasus-laravel
```



```
LENOVO D:\Dev\Latihan\Web\sekolah
> composer create-project laravel/laravel sikasus-laravel
Creating a "laravel/laravel" project at "./sikasus-laravel"
Installing laravel/laravel (v11.4.0)
- Downloading laravel/laravel (v11.4.0)
- Installing laravel/laravel (v11.4.0): Extracting archive
Created project in D:\Dev\Latihan\Web\sekolah\sikasus-laravel
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
Lock file operations: 110 installs, 0 updates, 0 removals
- Locking brick/math (0.12.1)
- Locking carbonphp/carbon-doctrine-types (3.2.0)
- Locking dflydev/dot-access-data (v3.0.3)
- Locking doctrine/inflector (2.0.10)
- Locking doctrine/lexer (3.0.1)
- Locking dragonmantank/cron-expression (v3.4.0)
- Locking egulias/email-validator (4.0.2)
- Locking fakerphp/faker (v1.24.1)
- Locking filp/whoops (2.16.0)
- Locking fruitcake/php-cors (v1.3.0)
- Locking graham-campbell/result-type (v1.1.3)
- Locking guzzlehttp/guzzle (7.9.2)
- Locking guzzlehttp/promises (2.0.4)
- Locking guzzlehttp/psr7 (2.7.0)
- Locking guzzlehttp/uri-template (v1.0.3)
- Locking hamcrest/hamcrest-php (v2.0.1)
- Locking laravel/framework (v11.35.0)
- Locking laravel/pail (v1.2.1)
```

Perintah ini akan membuat folder proyek baru bernama **sikasus-laravel** yang berisi instalasi **Laravel 11**. Tunggu hingga proses instalasi selesai.

Setelah instalasi selesai, pindah ke dalam folder proyek menggunakan perintah berikut:

```
cd sikasus-laravel
```

```
(Admin) D:\Dev\Latihan\Web' > + ▾
> @php artisan key:generate --ansi
[INFO] Application key set successfully.

> @php -r "file_exists('database/database.sqlite') || touch('database/database.sqlite');"
> @php artisan migrate --graceful --ansi
[INFO] Preparing database.

Creating migration table ...
[INFO] Running migrations.

0001_01_01_000000_create_users_table
0001_01_01_000001_create_cache_table
0001_01_01_000002_create_jobs_table

LENOVO D:\Dev\Latihan\Web\sekolah 50.225s
> cd .\sikasus-laravel\
LENOVO D:\Dev\Latihan\Web\sekolah\sikasus-laravel
> ls

Directory: D:\Dev\Latihan\Web\sekolah\sikasus-laravel
```

Langkah berikutnya adalah menjalankan server pengembangan bawaan **Laravel** untuk memastikan instalasi berhasil. Jalankan perintah berikut:

```
php artisan serve
```

```
----- 12/3/2024 6:11 AM 263 JS vit
LENOVO D:\Dev\Latihan\Web\sekolah\sikasus-laravel
> php artisan serve
forking is not supported on this platform

[INFO] Server running on [http://127.0.0.1:8000].
Press Ctrl+C to stop the server
```

Jika berhasil, Anda akan mendapatkan URL seperti `http://127.0.0.1:8000` yang dapat dibuka di browser untuk melihat tampilan awal proyek **Laravel 11**.

The screenshot shows the official Laravel documentation website at <https://laravel.com/docs>. The page features a dark theme with red highlights. On the left, there's a sidebar with a search bar and a navigation menu. The main content area displays several cards: one for Laracasts (video tutorials), one for Laravel News (community news), and one for the Vibrant Ecosystem (third-party tools). A large red button labeled "Documentation" is visible at the bottom of the sidebar.

Topik	Sub Topik	Status
Menginstal Laravel 11		<input checked="" type="checkbox"/>
Konfigurasi Proyek Laravel		
Membuat Model dan Migration		
Menambahkan Kolom di Dalam Migration	<ol style="list-style-type: none">1. Migration Kasus2. Migration Siswa3. Migration Kelas4. Migration Walikelas	
Menambahkan Mass Assignment	<ol style="list-style-type: none">1. Model Kasus2. Model Siswa3. Model Kelas4. Model Walikelas	
Menjalankan Proses Migrate		
Menerapkan Autentikasi		
Membuat Controller Auth		
Menambahkan Fungsi di AuthController		
	Menampilkan Halaman Jenis Login	
	Menampilkan Form Login Berdasarkan Jenis	

Topik	Sub Topik	Status
	Proses Login	
	Proses Logout	
	Membuat Routing untuk Autentikasi	
Membuat View untuk Menampilkan Halaman Login		
	Halaman Pilihan Login	
	Membuat Form Login	
Membuat Middleware		
	Membuat Middleware untuk Autentikasi	
	Membuat Middleware untuk Role	
	Registrasi Middleware	
Membuat Blade Layout		
Membuat Tampilan Homepage		
Membuat Tampilan Dashboard Setelah Login		
Membuat Fitur Kasus		
	Menambahkan Routing untuk Kasus	
	Membuat Controller Kasus	
	Menampilkan Daftar Kasus (Method index)	
	Membuat View untuk Menampilkan Kasus (View index)	
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kasus (View create)	
	Edit dan Update Kasus ke Database	
	Membuat View Form Edit Kasus (View edit)	
	Menambahkan Method Destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Siswa		
	Menambahkan Routing untuk Siswa	
	Membuat Controller Siswa	
	Menampilkan Daftar Siswa (Method index)	
	Membuat View untuk Menampilkan Siswa (View index)	

Topik	Sub Topik	Status
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Siswa (View create)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Walikelas		
	Menambahkan Routing untuk Walikelas	
	Membuat Controller Walikelas	
	Menampilkan Daftar Walikelas (Method index)	
	Membuat View untuk Menampilkan Walikelas (View index)	
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	

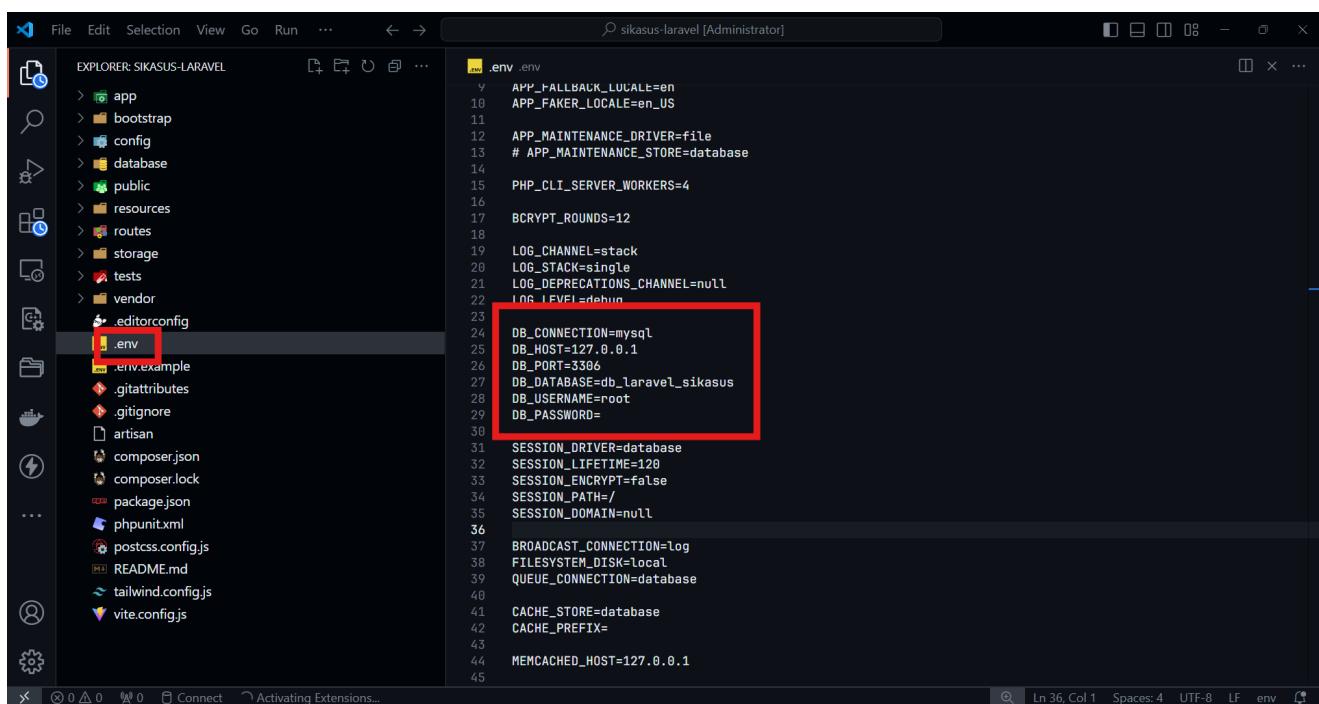
Topik	Sub Topik	Status
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk 'Siswa'	
	Membuat Controller 'Siswa'	

Konfigurasi Proyek Laravel

Langkah selanjutnya adalah mengatur konfigurasi database proyek. Buka file `.env` yang terdapat di dalam folder proyek menggunakan editor teks pilihan Anda.

Cari bagian pengaturan database, kemudian sesuaikan konfigurasi menjadi seperti berikut:

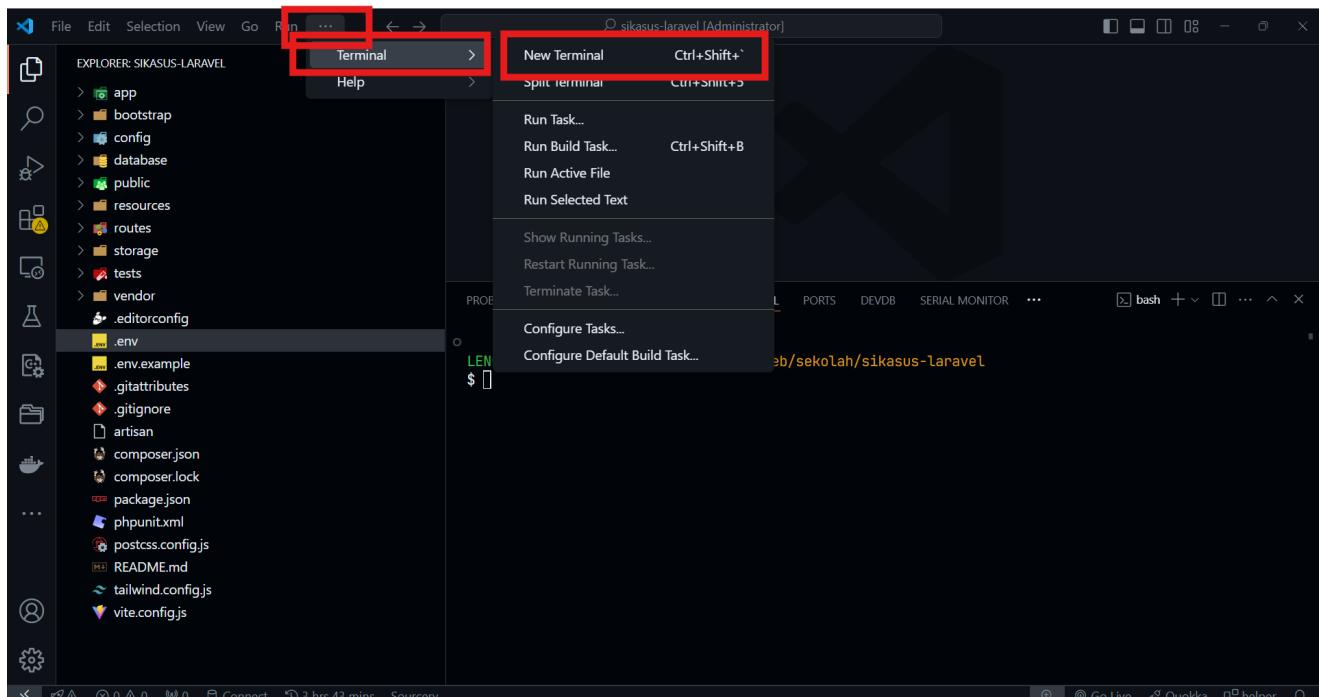
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=db_laravel_sikasus
DB_USERNAME=root
DB_PASSWORD=
```



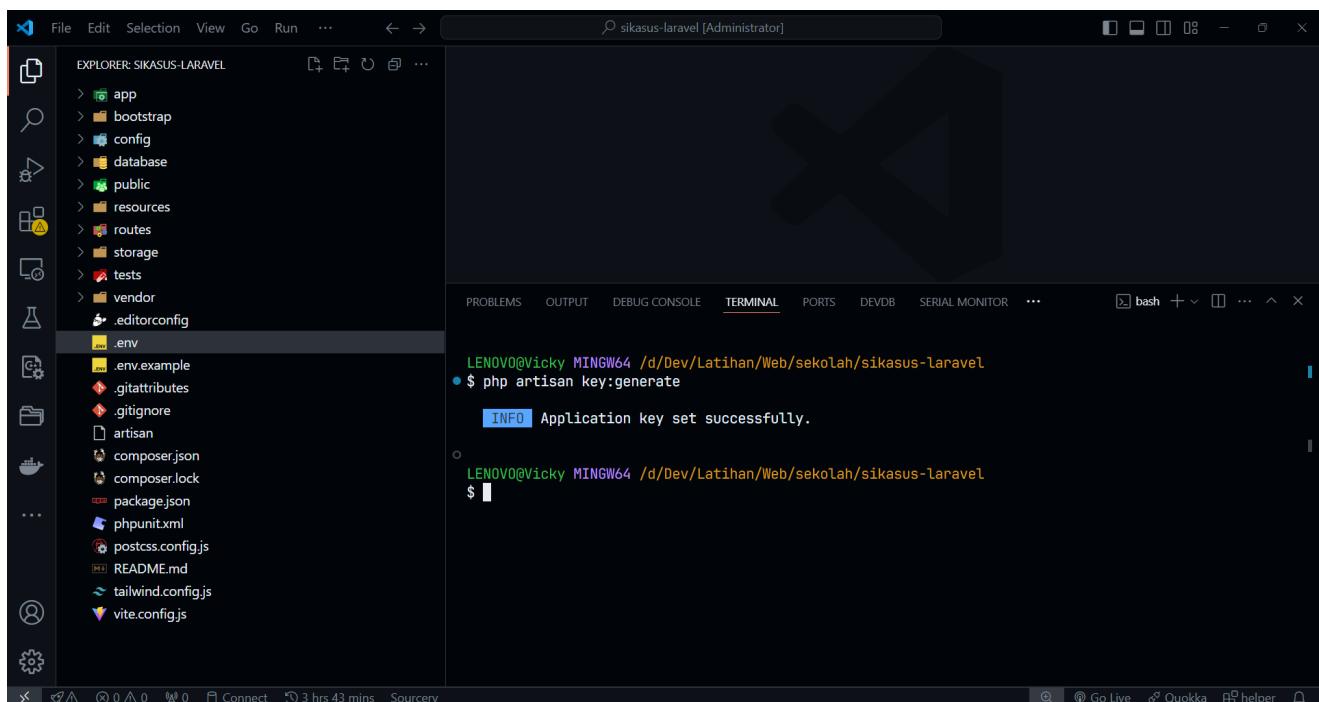
```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=db_laravel_sikasus
DB_USERNAME=root
DB_PASSWORD=
```

Penyesuaian ini memastikan proyek **Laravel** terhubung ke database **MySQL** dengan nama `db_laravel_sikasus`. Pastikan database tersebut sudah dibuat sebelumnya di **phpMyAdmin** atau **MySQL CLI**.

Setelah selesai mengatur file `.env`, simpan perubahan. Kemudian, jalankan perintah berikut untuk menghasilkan kunci aplikasi:



```
php artisan key:generate
```



Perintah ini akan membuat kunci enkripsi aplikasi dan menyimpannya di file `.env`. Kunci ini digunakan untuk menjaga keamanan data aplikasi.

Topik	Sub Topik	Status
Menginstal Laravel 11		<input checked="" type="checkbox"/>
Konfigurasi Proyek Laravel		<input checked="" type="checkbox"/>

Topik	Sub Topik	Status
Membuat Model dan Migration		
Menambahkan Kolom di Dalam Migration		
	1. Migration Kasus 2. Migration Siswa 3. Migration Kelas 4. Migration Walikelas	
Menambahkan Mass Assignment		
	1. Model Kasus 2. Model Siswa 3. Model Kelas 4. Model Walikelas	
Menjalankan Proses Migrate		
Menerapkan Autentikasi		
Membuat Controller Auth		
Menambahkan Fungsi di AuthController		
	Menampilkan Halaman Jenis Login Menampilkan Form Login Berdasarkan Jenis Proses Login Proses Logout Membuat Routing untuk Autentikasi	
Membuat View untuk Menampilkan Halaman Login		
	Halaman Pilihan Login Membuat Form Login	
Membuat Middleware		
	Membuat Middleware untuk Autentikasi Membuat Middleware untuk Role Registrasi Middleware	
Membuat Blade Layout		
Membuat Tampilan Homepage		
Membuat Tampilan Dashboard Setelah Login		

Topik	Sub Topik	Status
Membuat Fitur Kasus	<ul style="list-style-type: none"> Menambahkan Routing untuk Kasus Membuat Controller Kasus Menampilkan Daftar Kasus (Method index) Membuat View untuk Menampilkan Kasus (View index) Menambahkan Kasus Baru ke Dalam Database (Method create dan store) Membuat Form untuk Menambah Kasus (View create) Edit dan Update Kasus ke Database Membuat View Form Edit Kasus (View edit) Menambahkan Method Destroy di Controller Menambahkan Tombol Hapus di View 	
Membuat Fitur Siswa	<ul style="list-style-type: none"> Menambahkan Routing untuk Siswa Membuat Controller Siswa Menampilkan Daftar Siswa (Method index) Membuat View untuk Menampilkan Siswa (View index) Menambahkan Siswa Baru ke Dalam Database (Method create dan store) Membuat Form untuk Menambah Siswa (View create) Menambahkan Method destroy di Controller Menambahkan Tombol Hapus di View 	
Membuat Fitur Walikelas	<ul style="list-style-type: none"> Menambahkan Routing untuk Walikelas Membuat Controller Walikelas Menampilkan Daftar Walikelas (Method index) Membuat View untuk Menampilkan Walikelas (View index) Menambahkan Walikelas Baru ke Dalam Database (Method create dan store) Membuat Form untuk Menambah Walikelas (View create) 	

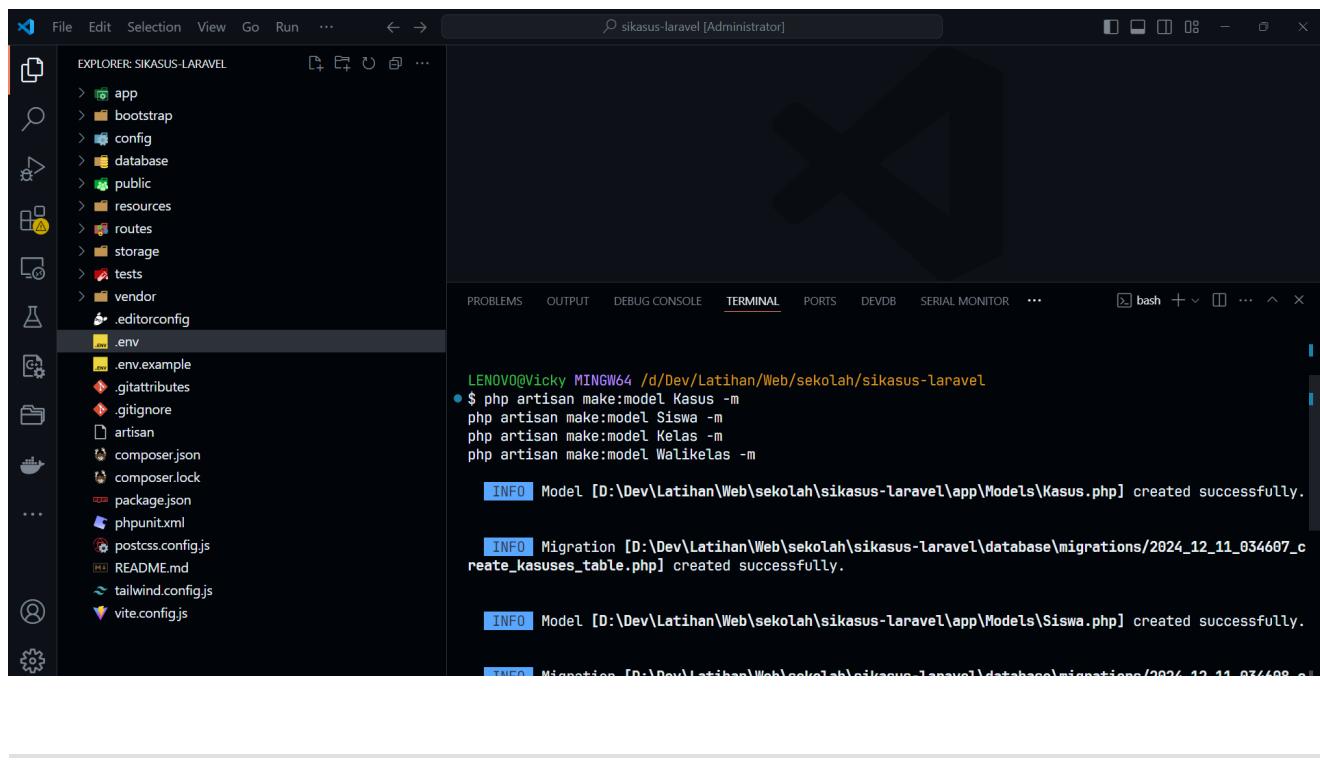
Topik	Sub Topik	Status
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk 'Siswa	
	Membuat Controller 'Siswa	

Membuat Model dan Migration

Langkah berikutnya adalah membuat **Model** beserta **Migration** untuk entitas `kasus` , `siswa` , `kelas` , dan `walikelas` . Jalankan perintah berikut di terminal untuk masing-masing entitas:

```
php artisan make:model Walikelas -m
php artisan make:model Kelas -m
php artisan make:model Siswa -m
php artisan make:model Kasus -m
```

Perintah di atas akan membuat file **Model** di folder `app/Models` dan file **Migration** di folder `database/migrations`.



The screenshot shows the VS Code interface with the terminal tab selected. The terminal window displays the command-line output of running Laravel migrations:

```
LENVOV0@Vicky MINGW64 /d/Dev/Latihan/Web/sekolah/sikasus-laravel
● $ php artisan make:model Kasus -m
php artisan make:model Siswa -m
php artisan make:model Kelas -m
php artisan make:model Walikelas -m

[INFO] Model [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\app\Models\Kasus.php] created successfully.

[INFO] Migration [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\database\migrations\2024_12_11_034607_create_kasuses_table.php] created successfully.

[INFO] Model [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\app\Models\Siswa.php] created successfully.

[INFO] Migration [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\database\migrations\2024_12_11_034607_create_kelas_table.php] created successfully.
```

Menambahkan Kolom di Dalam Migration

Laravel menyediakan berbagai fungsi untuk mendefinisikan struktur kolom di database melalui migration. Fungsi-fungsi utama yang digunakan meliputi:

1. `id()`

Membuat kolom primary key dengan auto-increment.

Contoh:

```
$table->id('id_kasus');
```

2. `foreignId()`

Membuat kolom foreign key dengan tipe data `BIGINT` dan relasi ke tabel lain.

Contoh:

```
$table->foreignId('siswa_id')
    ->constrained('siswa', 'id')
    ->cascadeOnDelete()
    ->cascadeOnUpdate();
```

3. `string()`

Membuat kolom dengan tipe data `VARCHAR`.

Contoh:

```
$table->string('nama_lengkap', 100);
```

4. `text()`

Membuat kolom untuk teks panjang.

Contoh:

```
$table->text('alamat');
```

5. `timestamps()`

Menambahkan kolom otomatis `created_at` dan `updated_at`.

Contoh:

```
$table->timestamps();
```

untuk detailnya silahkan bukan halaman berikut : <https://github.com/xi-rpl-1/Laravel/blob/main/Laravel%20Dasar.md#migration>

Setelah memahami fungsi-fungsi tersebut, berikut adalah implementasi pada setiap file migration:

1. Migration Kasus

```
public function up()
{
    Schema::create('kasus', function (Blueprint $table) {
        $table->id('id_kasus');
        $table->foreignId('siswa_id')->constrained('siswa', 'id')-
>cascadeOnDelete()->cascadeOnUpdate();
        $table->text('deskripsi_kasus');
        $table->date('tanggal_kasus');
        $table->timestamps();
    });
}
```

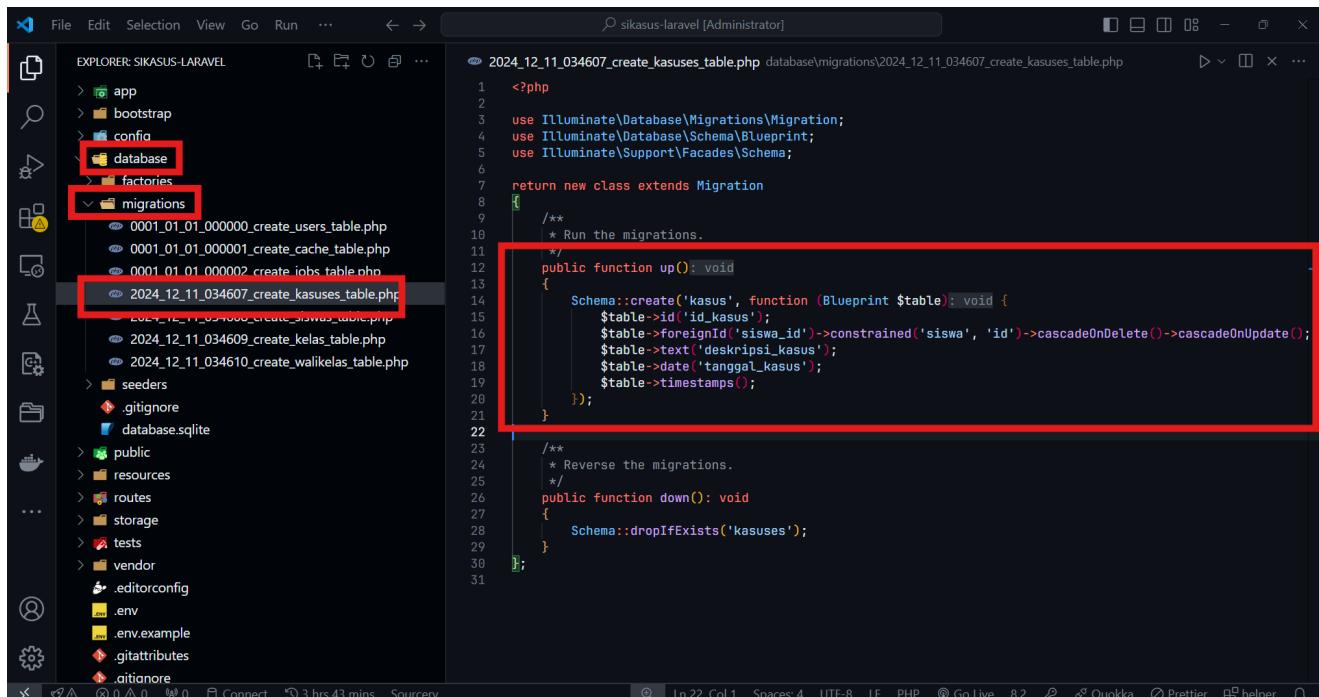
Penjelasan:

Pada kode di atas, fungsi `up()` digunakan untuk mendefinisikan struktur tabel `kasus` dalam database menggunakan migration di Laravel.

1. `$table->id('id_kasus')`: Membuat kolom `id_kasus` sebagai kolom primary key yang otomatis bertipe `BIGINT` dan auto increment.
2. `$table->foreignId('siswa_id')->constrained('siswa', 'id')->cascadeOnDelete()-
>cascadeOnUpdate()`: Menambahkan kolom `siswa_id` yang berfungsi sebagai foreign key yang mengacu pada kolom `id` di tabel `siswa`. Dengan menggunakan

`constrained()`, Laravel secara otomatis menetapkan referensi yang tepat ke tabel `siswa`. `cascadeOnDelete()` dan `cascadeOnUpdate()` memastikan bahwa jika data siswa dihapus atau diperbarui, perubahan tersebut akan diterapkan secara otomatis ke data terkait di tabel `kasus`.

3. `$table->text('deskripsi_kasus')`: Menambahkan kolom `deskripsi_kasus` yang bertipe `TEXT` untuk menyimpan deskripsi dari kasus.
4. `$table->date('tanggal_kasus')`: Menambahkan kolom `tanggal_kasus` yang bertipe `DATE` untuk menyimpan tanggal terjadinya kasus.
5. `$table->timestamps()`: Menambahkan kolom `created_at` dan `updated_at` secara otomatis untuk melacak kapan data kasus dibuat dan diperbarui.



```
<?php  
use Illuminate\Database\Migrations\Migration;  
use Illuminate\Database\Schema\Blueprint;  
use Illuminate\Support\Facades\Schema;  
  
return new class extends Migration  
{  
    public function up(): void  
    {  
        Schema::create('kasus', function (Blueprint $table): void {  
            $table->id('id_kasus');  
            $table->foreignId('siswa_id')->constrained('siswa', 'id')->cascadeOnDelete()->cascadeOnUpdate();  
            $table->text('deskripsi_kasus');  
            $table->date('tanggal_kasus');  
            $table->timestamps();  
        });  
    }  
  
    public function down(): void  
    {  
        Schema::dropIfExists('kasuses');  
    }  
};
```

2. Migration Siswa

```
public function up()  
{  
    Schema::create('siswa', function (Blueprint $table) {  
        $table->id();  
        $table->string('nama_lengkap', 100);  
        $table->string('nisn', 20)->unique();  
        $table->enum('jenis_kelamin', ['Laki-Laki', 'Perempuan']);  
        $table->date('tanggal_lahir');  
        $table->text('alamat');  
        $table->foreignId('kelas_id')->constrained('kelas', 'id_kelas')-  
        >cascadeOnDelete()->cascadeOnUpdate();  
        $table->timestamps();  
    });  
}
```

Penjelasan:

Pada kode di atas, fungsi `up()` digunakan untuk mendefinisikan struktur tabel `siswa` dalam database menggunakan migration di Laravel.

1. `$table->id()`: Membuat kolom `id` sebagai kolom primary key yang otomatis bertipe `BIGINT` dan auto increment.
2. `$table->string('nama_lengkap', 100)`: Menambahkan kolom `nama_lengkap` dengan tipe data `string` dan panjang maksimal 100 karakter untuk menyimpan nama lengkap siswa.
3. `$table->string('nisn', 20)->unique()`: Menambahkan kolom `nisp` dengan tipe data `string` dan panjang maksimal 20 karakter. Kolom ini juga diatur agar memiliki nilai unik, artinya tidak ada dua siswa yang dapat memiliki `nisp` yang sama.
4. `$table->enum('jenis_kelamin', ['Laki-Laki', 'Perempuan'])`: Menambahkan kolom `jenis_kelamin` dengan tipe data `enum`, yang hanya bisa berisi dua nilai: 'Laki-Laki' atau 'Perempuan'.
5. `$table->date('tanggal_lahir')`: Menambahkan kolom `tanggal_lahir` dengan tipe data `DATE` untuk menyimpan tanggal lahir siswa.
6. `$table->text('alamat')`: Menambahkan kolom `alamat` dengan tipe data `TEXT` untuk menyimpan alamat siswa.
7. `$table->foreignId('kelas_id')->constrained('kelas', 'id_kelas')->cascadeOnDelete()->cascadeOnUpdate()`: Menambahkan kolom `kelas_id` yang berfungsi sebagai foreign key yang mengacu pada kolom `id_kelas` di tabel `kelas`. Dengan menggunakan `constrained()`, Laravel secara otomatis menetapkan referensi yang tepat ke tabel `kelas`. `cascadeOnDelete()` dan `cascadeOnUpdate()` memastikan bahwa jika data `kelas` dihapus atau diperbarui, perubahan tersebut akan diterapkan secara otomatis ke data siswa yang terkait.
8. `$table->timestamps()`: Menambahkan kolom `created_at` dan `updated_at` secara otomatis untuk melacak kapan data siswa dibuat dan diperbarui.

```
<?php

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

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('siswas', function (Blueprint $table) {
            $table->id();
            $table->string('nama_lengkap', 100);
            $table->string('nisp', 20)->unique();
            $table->enum('jenis_kelamin', ['Laki-Laki', 'Perempuan']);
            $table->date('tanggal_lahir');
            $table->text('alamat');
            $table->foreignId('kelas_id')->constrained('kelas', 'id_kelas')->cascadeOnDelete()->cascadeOnUpdate();
            $table->timestamps();
        });
    }

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

3. Migration Kelas

```
public function up()
{
    Schema::create('kelas', function (Blueprint $table) {
        $table->id('id_kelas');
        $table->string('nama_kelas', 50);
        $table->foreignId('walikelas_id')-
>constrained('walikelas', 'id_walikelas')->unique()->cascadeOnDelete()-
>cascadeOnUpdate();
        $table->timestamps();
    });
}
```

Penjelasan:

Pada kode di atas, fungsi `up()` digunakan untuk mendefinisikan struktur tabel `kelas` dalam database menggunakan migration di Laravel.

1. `$table->id('id_kelas')`: Membuat kolom `id_kelas` sebagai primary key yang bertipe `BIGINT` dan auto increment, yang akan digunakan sebagai identifikasi unik untuk setiap kelas.
2. `$table->string('nama_kelas', 50)`: Menambahkan kolom `nama_kelas` dengan tipe data `string` dan panjang maksimal 50 karakter untuk menyimpan nama kelas, misalnya "X A", "X B", dan seterusnya.
3. `$table->foreignId('walikelas_id')->constrained('walikelas', 'id_walikelas')->unique()->cascadeOnDelete()>cascadeOnUpdate()`: Menambahkan kolom `walikelas_id` sebagai foreign key yang mengacu pada kolom `id_walikelas` di tabel `walikelas`. Dengan menggunakan `constrained()`, Laravel secara otomatis menetapkan referensi yang tepat ke tabel `walikelas`. `unique()` memastikan bahwa setiap kelas hanya bisa memiliki satu wali kelas, sedangkan `cascadeOnDelete()` dan `cascadeOnUpdate()` memastikan bahwa jika data wali kelas dihapus atau diperbarui, perubahan tersebut akan diterapkan secara otomatis ke data kelas yang terkait.
4. `$table->timestamps()`: Menambahkan kolom `created_at` dan `updated_at` secara otomatis untuk melacak kapan data kelas dibuat dan diperbarui.

```

2024_12_11_034609_create_kelas_table.php database\migrations\2024_12_11_034609_create_kelas_table.php
1  hp
2
3  • Illuminate\Database\Migrations\Migration;
4  • Illuminate\Database\Schema\Blueprint;
5  • Illuminate\Support\Facades\Schema;
6
7  urn new class extends Migration {
8  /**
9   * Run the migrations.
10  */
11  public function up(): void
12  {
13      Schema::create('kelas', callback: function (Blueprint $table): void {
14          $table->id();
15          $table->string('nama_kelas', 50);
16          $table->foreignId('walikelas_id')->constrained(table: 'walikelas', column: 'id_walikelas')->
17          $table->timestamps();
18      });
19  }
20
21  /**
22   * Reverse the migrations.
23  */
24  public function down(): void
25  {
26      Schema::dropIfExists(table: 'kelas');
27  }
28
29

```

4. Migration Walikelas

```

public function up()
{
    Schema::create('walikelas', function (Blueprint $table) {
        $table->id('id_walikelas');
        $table->string('nama_walikelas', 100);
        $table->string('nip', 20)->unique()->nullable();
        $table->enum('jenis_kelamin', ['Laki-laki', 'Perempuan']);
        $table->text('alamat');
        $table->timestamps();
    });
}

```

```

2024_12_11_034610_create_walikelas_table.php database\migrations\2024_12_11_034610_create_walikelas_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10     * Run the migrations.
11    */
12    public function up(): void
13    {
14        Schema::create('walikelas', callback: function (Blueprint $table): void {
15            $table->id('id_walikelas');
16            $table->string('nama_walikelas', 100);
17            $table->string('nip', 20)->unique()->nullable();
18            $table->enum('jenis_kelamin', allowed: ['Laki-laki', 'Perempuan']);
19            $table->text('alamat');
20            $table->timestamps();
21        });
22    }
23
24    /**
25     * Reverse the migrations.
26    */
27    public function down(): void
28    {
29        Schema::dropIfExists('walikelas');
30    }
31
32

```

Penjelasan:

Pada kode di atas, fungsi `up()` digunakan untuk mendefinisikan struktur tabel `walikelas` dalam database menggunakan migration di Laravel.

1. `$table->id('id_walikelas')`: Membuat kolom `id_walikelas` sebagai primary key yang bertipe `BIGINT` dan auto increment. Kolom ini digunakan untuk memberikan identifikasi unik pada setiap wali kelas.
2. `$table->string('nama_walikelas', 100)`: Menambahkan kolom `nama_walikelas` dengan tipe data `string` dan panjang maksimal 100 karakter untuk menyimpan nama wali kelas.
3. `$table->string('nip', 20)->unique()->nullable()`: Menambahkan kolom `nip` dengan tipe data `string` dan panjang maksimal 20 karakter untuk menyimpan Nomor Induk Pegawai (NIP) wali kelas. Kolom ini diberi aturan `unique()` untuk memastikan tidak ada NIP yang duplikat. `nullable()` berarti kolom ini boleh kosong jika tidak ada NIP yang tercatat.
4. `$table->enum('jenis_kelamin', ['Laki-laki', 'Perempuan'])`: Menambahkan kolom `jenis_kelamin` dengan tipe data `enum`, yang hanya bisa berisi dua nilai: 'Laki-laki' atau 'Perempuan'. Kolom ini digunakan untuk menyimpan informasi jenis kelamin wali kelas.
5. `$table->text('alamat')`: Menambahkan kolom `alamat` dengan tipe data `text`, yang digunakan untuk menyimpan alamat lengkap wali kelas.
6. `$table->timestamps()`: Menambahkan kolom `created_at` dan `updated_at` secara otomatis untuk melacak waktu pembuatan dan pembaruan data wali kelas.

Pengenalan Properti dan Fungsi di Model Laravel

Dalam Laravel, model digunakan untuk berinteraksi dengan tabel di database. Berikut adalah properti dan fungsi penting dalam sebuah model:

1. `$table`

Properti ini menentukan nama tabel yang terkait dengan model.

Contoh:

```
protected $table = 'kasus';
```

2. `$primaryKey`

Properti ini menentukan primary key dari tabel.

Contoh:

```
protected $primaryKey = 'id_kasus';
```

3. \$fillable

Properti ini digunakan untuk menentukan kolom mana saja yang dapat diisi secara massal melalui Mass Assignment.

Contoh:

```
protected $fillable = ['deskripsi_kasus', 'tanggal_kasus', 'siswa_id'];
```

4. Relasi Antar Model

Laravel menyediakan berbagai fungsi untuk mendefinisikan relasi antar tabel:

- `belongsTo` : Relasi ke model lain sebagai foreign key.
- `hasMany` : Relasi dari satu model ke banyak data di model lain.
- `hasOne` : Relasi satu ke satu.

Contoh relasi:

```
public function siswa() {
    return $this->belongsTo(Siswa::class, 'siswa_id', 'id');
}
```

5. \$casts

Properti ini digunakan untuk mengubah tipe data kolom saat diakses dari model.

Contoh:

```
protected $casts = [
    'tanggal_kasus' => 'date',
];
```

Menambahkan Mass Assignment

Untuk setiap **Model**, tambahkan properti `$fillable` agar mendukung **Mass Assignment**.

Contoh:

1. Model Kasus

```
protected $table = 'kasus';
protected $primaryKey = 'id_kasus';
protected $fillable = ['deskripsi_kasus', 'tanggal_kasus', 'siswa_id'];
```

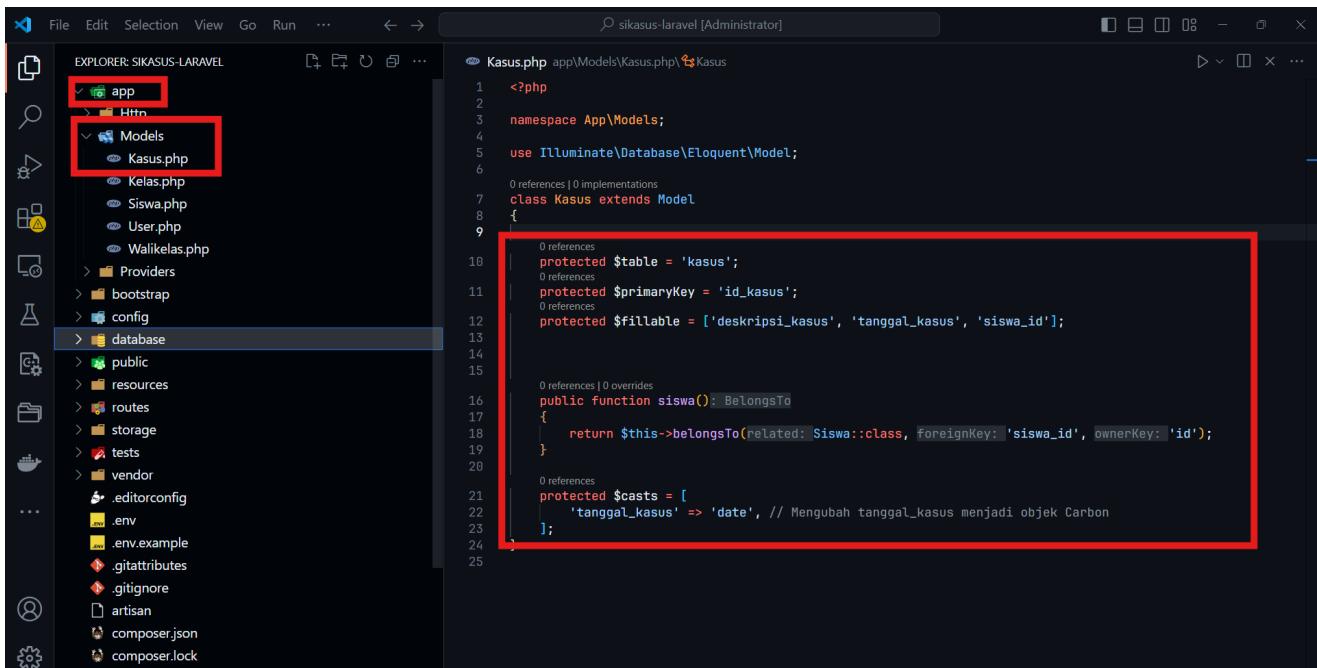
```
public function siswa(){
    return $this->belongsTo(Siswa::class, 'siswa_id', 'id');
}

protected $casts = [
    'tanggal_kasus' => 'date', // Mengubah tanggal_kasus menjadi objek
    Carbon
];
```

Penjelasan:

Pada kode di atas, kita mendefinisikan properti dan relasi untuk model Kasus di Laravel:

1. **\$table = 'kasus'**: Properti ini menentukan nama tabel yang digunakan oleh model ini di database. Dalam hal ini, model Kasus akan berhubungan dengan tabel kasus .
2. **\$primaryKey = 'id_kasus'**: Properti ini digunakan untuk menentukan nama kolom yang berfungsi sebagai primary key untuk tabel kasus . Secara default, Laravel menggunakan kolom id sebagai primary key, tetapi di sini kita menyebutkan id_kasus untuk menyesuaikan dengan struktur tabel.
3. **\$fillable = ['deskripsi_kasus', 'tanggal_kasus', 'siswa_id']**: Properti \$fillable mendefinisikan kolom-kolom yang dapat diisi melalui metode Mass Assignment. Dengan menambahkan kolom deskripsi_kasus , tanggal_kasus , dan siswa_id ke dalam array ini, kita memastikan hanya kolom-kolom tersebut yang dapat diisi saat menggunakan metode seperti create() atau update() .
4. **public function siswa()**: Fungsi ini mendefinisikan relasi antara model Kasus dan model Siswa . Di sini, kita menggunakan metode belongsTo() yang menunjukkan bahwa setiap kasus terkait dengan satu siswa. Relasi ini menjelaskan bahwa kasus memiliki foreign key siswa_id yang merujuk ke primary key id pada tabel siswa .
5. **\$casts = ['tanggal_kasus' => 'date']**: Properti \$casts digunakan untuk memaksa Laravel mengonversi kolom tanggal_kasus menjadi tipe data date saat mengaksesnya. Biasanya, Laravel secara otomatis mengonversi tanggal ke objek Carbon jika tipe data kolom adalah date atau datetime , tetapi menggunakan \$casts ini memastikan bahwa data tanggal_kasus akan selalu dalam format yang sesuai.



```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Kasus extends Model
{
    protected $table = 'kasus';
    protected $primaryKey = 'id_kasus';
    protected $fillable = ['deskripsi_kasus', 'tanggal_kasus', 'siswa_id'];
}

public function siswa(): BelongsTo
{
    return $this->belongsTo(Siswa::class, 'siswa_id', 'id');
}

protected $casts = [
    'tanggal_kasus' => 'date', // Mengubah tanggal_kasus menjadi objek Carbon
];
```

2. Model Siswa

```
protected $table = 'siswa';
protected $primaryKey = 'id';
protected $fillable = [
    'nama_lengkap',
    'nisn',
    'jenis_kelamin',
    'tanggal_lahir',
    'alamat',
    'kelas_id'
];

// Relasi dengan kelas
public function kelas() {
    return $this->belongsTo(Kelas::class, 'kelas_id', 'id_kelas');
}

public function kasus() {
    return $this->hasMany(Kasus::class, 'siswa_id'); // Foreign key harus
'siswa_id'
}
```

Penjelasan:

Pada kode di atas, kita mendefinisikan properti dan relasi untuk model `Siswa` di Laravel:

1. **\$table = 'siswa'**: Properti ini menentukan nama tabel yang digunakan oleh model `Siswa` di database. Dalam hal ini, model `Siswa` akan berhubungan dengan tabel `siswa`.

2. **\$primaryKey = 'id'**: Properti ini menentukan kolom yang digunakan sebagai primary key dalam tabel siswa . Secara default, Laravel menggunakan kolom id sebagai primary key, sehingga kita tidak perlu mengubahnya.
3. **\$fillable = ['nama_lengkap', 'nisn', 'jenis_kelamin', 'tanggal_lahir', 'alamat', 'kelas_id']**: Properti \$fillable mendefinisikan kolom-kolom yang dapat diisi menggunakan Mass Assignment. Dengan menyebutkan kolom-kolom ini, kita memungkinkan pengguna untuk mengisi data seperti nama_lengkap , nisn , jenis_kelamin , dan lainnya melalui metode create() atau update() .
4. **public function kelas()**: Fungsi ini mendefinisikan relasi antara model Siswa dan model Kelas . Di sini, kita menggunakan metode belongsTo() untuk menunjukkan bahwa setiap siswa terkait dengan satu kelas. Relasi ini menjelaskan bahwa kolom kelas_id pada tabel siswa merujuk ke kolom id_kelas pada tabel kelas .
5. **public function kasus()**: Fungsi ini mendefinisikan relasi antara model Siswa dan model Kasus . Kita menggunakan metode hasMany() untuk menunjukkan bahwa satu siswa bisa memiliki banyak kasus. Relasi ini menjelaskan bahwa kolom siswa_id pada tabel kasus merujuk ke primary key id pada tabel siswa .

```

File Edit Selection View Go Run ... < >
Sikasus-laravel [Administrator]
EXPLORER: SIKASUS-LARAVEL
app
> Http
Models
  Kasus.php
  Kelas.php
  Siswa.php
  User.php
  Walikelas.php
  Providers
  bootstrap
  config
  database
  public
  resources
  routes
  storage
  tests
  vendor
  .editorconfig
  .env
  .env.example
  .gitattributes
  .gitignore
  artisan
  composer.json
  composer.lock
Siswa.php app\Models\Siswa.php Sikasus\Kelas
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Siswa extends Model
{
    protected $table = 'siswa';
    protected $primaryKey = 'id';
    protected $fillable = [
        'nama_lengkap',
        'nisn',
        'jenis_kelamin',
        'tanggal_lahir',
        'alamat',
        'kelas_id'
    ];
    // Relasi dengan kelas
    public function kelas(): BelongsTo
    {
        return $this->belongsTo(related: Kelas::class, foreignKey: 'kelas_id', ownerKey: 'id_kelas');
    }
    // Relasi dengan kasus
    public function kasus(): HasMany
    {
        return $this->hasMany(related: Kasus::class, foreignKey: 'siswa_id'); // Foreign key harus 'siswa_id'
    }
}

```

3. Model Kelas

```

protected $table = 'kelas';
protected $primaryKey = 'id_kelas';
protected $fillable = [
    'nama_kelas',
    'walikelas_id',
];
public function walikelas() {
    return $this->belongsTo(Walikelas::class, 'walikelas_id');
}

```

Penjelasan:

Pada kode di atas, kita mendefinisikan properti dan relasi untuk model Kelas di Laravel:

1. **\$table = 'kelas'**: Properti ini menentukan nama tabel yang digunakan oleh model Kelas di database. Dalam hal ini, model Kelas akan berhubungan dengan tabel kelas .
2. **\$primaryKey = 'id_kelas'**: Properti ini menentukan kolom yang digunakan sebagai primary key dalam tabel kelas . Secara default, Laravel menggunakan kolom id sebagai primary key, tetapi di sini kita mengubahnya menjadi id_kelas .
3. **\$fillable = ['nama_kelas', 'walikelas_id']**: Properti \$fillable mendefinisikan kolom-kolom yang dapat diisi menggunakan Mass Assignment. Kolom-kolom yang disebutkan di sini, seperti nama_kelas dan walikelas_id , dapat diisi melalui metode create() atau update() pada model Kelas .
4. **public function walikelas()**: Fungsi ini mendefinisikan relasi antara model Kelas dan model Walikelas . Di sini, kita menggunakan metode belongsTo() untuk menunjukkan bahwa setiap kelas terkait dengan satu wali kelas. Relasi ini menjelaskan bahwa kolom walikelas_id pada tabel kelas merujuk ke kolom id_walikelas pada tabel walikelas .

```
File Edit Selection View Go Run ... ← → ⌂ siskus-laravel [Administrator] ⌂ Kelas.php app\Models\Kelas.php Kelas walikelas()
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Model;
6
7 1 reference | 0 implementations
8 class Kelas extends Model
9 {
10     0 references
11     protected $table = 'kelas';
12     0 references
13     protected $primaryKey = 'id_kelas';
14     0 references
15     protected $fillable = [
16         'nama_kelas',
17         'walikelas_id',
18     ];
19
20     0 references | 0 overrides
21     public function walikelas(): BelongsTo
22     {
23         return $this->belongsTo(related: Walikelas::class, foreignKey: 'walikelas_id');
24     }
25 }
```

4. Model Walikelas

```
protected $table = 'walikelas'; // Pastikan tabel ini sesuai dengan migrasi
protected $primaryKey = 'id_walikelas'; // Pastikan primary key adalah
id_walikelas
protected $fillable = [
    'nama_walikelas',
    'nip',
    'jenis_kelamin',
```

The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "EXPLORER: SIKASUS-LARAVEL". The "Models" folder is expanded, showing files: Kasus.php, Kelas.php, Siswa.php, User.php, and Walikelas.php. Walikelas.php is currently selected.
- Editor (Right):** Displays the code for `Walikelas.php` located at `app\Models\Walikelas.php`. The code defines a Model named `Walikelas` extending `Model`. It includes protected properties for `$table`, `$primaryKey`, and `$fillable`, which includes the fields: `'nama_walikelas'`, `'nip'`, `'jenis_kelamin'`, and `'alamat'`.
- Status Bar:** Shows the path `sikasus-laravel [Administrator]` and various system icons.

Penjelasan:

Pada kode di atas, kita mendefinisikan properti dan pengaturan untuk model `Walikelas` di Laravel:

1. **\$table = 'walikelas'**: Properti ini menentukan nama tabel yang digunakan oleh model Walikelas di database. Dalam hal ini, model Walikelas berhubungan dengan tabel walikelas .
 2. **\$primaryKey = 'id_walikelas'**: Properti ini menentukan kolom yang digunakan sebagai primary key dalam tabel walikelas . Secara default, Laravel menggunakan kolom id sebagai primary key, tetapi di sini kita mengubahnya menjadi id_walikelas .
 3. **\$fillable = ['nama_walikelas', 'nip', 'jenis_kelamin', 'alamat']**: Properti \$fillable mendefinisikan kolom-kolom yang dapat diisi menggunakan Mass Assignment. Kolom-kolom yang disebutkan di sini, seperti nama_walikelas , nip , jenis_kelamin , dan alamat , dapat diisi melalui metode create() atau update() pada model Walikelas . Dengan cara ini, Laravel melindungi dari pengisian kolom yang tidak diinginkan secara otomatis.

Properti dan metode ini membantu Laravel dalam pengelolaan data dan relasi antar tabel dengan lebih mudah dan aman.

Menjalankan Proses Migrate

Setelah semua *migration* selesai dibuat, jalankan perintah berikut untuk membuat tabel di database:

```
php artisan migrate
```

Jika berhasil, tabel `kasus`, `siswa`, `kelas`, dan `walikelas` akan dibuat di database.

The screenshot shows the SourceTree interface with the project "SIKASUS-LARAVEL". The left sidebar shows the file structure, including the "migrations" folder which contains several migration files like 0001_01_01_000000_create_users_table.php and 2024_12_11_0346010_create_kasus_table.php. The right pane shows the terminal output of running `php artisan migrate`. The output shows the creation of the database connection and the execution of various migrations, with a total time of 74.19ms.

```
2024_12_11_0346010_create_kasus_table.php database/migrations/2024_12_11_0346010_create_kasus_table.php
7     return new class extends Migration
9     /**
10    public function up(): void
11    {
12        Schema::create('kasus', callback: function (Blueprint $table): void {
13            $table->id('id_kasus');
14            $table->foreignId('siswa_id')->constrained('table: 'siswa', column: 'id')->cascadeOnDelete();
15            $table->text('deskripsi_kasus');
16            $table->date('tanggal_kasus');
17            $table->timestamps();
18        });
19    }
20}
21

LENovo@Vicky MINGW64 /d/Dev/Latihan/Web/sekolah/sikasus-laravel
$ php artisan migrate
WARN | The database 'db_laravel_sikasus' does not exist on the 'mysql' connection.
Would you like to create it? (yes/no) [yes]
>

INFO | Preparing database.
Creating migration table ..... 74.19ms DONE
INFO | Running migrations.

0001_01_01_000000_create_users_table ..... 145.19ms DONE
0001_01_01_000001_create_cache_table ..... 37.29ms DONE
0001_01_01_000002_create_jobs_table ..... 134.45ms DONE
2024_10_11_034607_create_walikelas_table ..... 36.55ms DONE
```

The screenshot shows the phpMyAdmin interface connected to the database "db_laravel_sikasus". The left sidebar shows the database structure with tables like cache, cache_locks, failed_jobs, jobs, job_batches, kasus, kelas, migrations, password_reset_tokens, sessions, siswa, users, and walikelas. The main panel displays the "Tabel" (Tables) page, listing all these tables with their respective details such as Baris (Rows), Jenis (Type), Penyortiran (Sorting), Ukuran (Size), and Beban (Load).

Tabel	Tindakan	Baris	Jenis	Penyortiran	Ukuran	Beban
cache	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
cache_locks	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
failed_jobs	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
jobs	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
job_batches	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
kasus	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
kelas	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
migrations	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	7	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
password_reset_tokens	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
sessions	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
siswa	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	32.0 KB	-
users	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
walikelas	Jelajahi Struktur Cari Tambahkan Kosongkan Hapus	0	InnoDB	utf8mb4_unicode_ci	16.0 KB	-
13 tabel	Jumlah	7	InnoDB	utf8mb4_0900_ai_ci	256.0 KB	0 B

Fitur Lain Selain Migrate di Laravel

Laravel menyediakan berbagai fitur selain **migrate** untuk mempermudah pengelolaan database:

1. Rollback Migration

Untuk membatalkan migrasi terakhir yang dijalankan, gunakan perintah **migrate:rollback**. Perintah ini akan mengembalikan perubahan yang dilakukan oleh migrasi terakhir.

```
php artisan migrate:rollback
```

2. Reset Migration

Dengan **migrate:reset**, Anda dapat menghapus seluruh tabel yang dibuat oleh migrasi dan mengembalikan database ke keadaan awal.

```
php artisan migrate:reset
```

3. Refresh Migration

migrate:refresh akan menghapus tabel yang ada dan langsung menjalankan ulang semua migrasi untuk memperbarui struktur database.

```
php artisan migrate:refresh
```

4. Seed Database

Setelah migrasi, gunakan **db:seed** untuk mengisi tabel dengan data awal yang telah ditentukan di seeder.

```
php artisan db:seed
```

5. Create Migration dan Seeder

Gunakan **make:migration** untuk membuat migrasi baru, dan **make:seeder** untuk membuat seeder baru.

```
php artisan make:migration create_nama_tabel
php artisan make:seeder NamaSeeder
```

6. Database Factories

Factories memungkinkan Anda untuk membuat data palsu yang berguna untuk pengujian atau pengembangan.

```
php artisan make:factory NamaFactory
```

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration		✓
	1. Migration Kasus	✓
	2. Migration Siswa	✓
	3. Migration Kelas	✓
	4. Migration Walikelas	✓
Menambahkan Mass Assignment		✓
	1. Model Kasus	✓
	2. Model Siswa	✓
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		
Membuat Controller Auth		
Menambahkan Fungsi di AuthController		
	Menampilkan Halaman Jenis Login	
	Menampilkan Form Login Berdasarkan Jenis	
	Proses Login	
	Proses Logout	
	Membuat Routing untuk Autentikasi	
Membuat View untuk Menampilkan Halaman Login		
	Halaman Pilihan Login	
	Membuat Form Login	

Topik	Sub Topik	Status
Membuat Middleware	Membuat Middleware untuk Autentikasi Membuat Middleware untuk Role Registrasi Middleware	
Membuat Blade Layout		
Membuat Tampilan Homepage		
Membuat Tampilan Dashboard Setelah Login		
Membuat Fitur Kasus	Menambahkan Routing untuk Kasus Membuat Controller Kasus Menampilkan Daftar Kasus (Method index) Membuat View untuk Menampilkan Kasus (View index) Menambahkan Kasus Baru ke Dalam Database (Method create dan store) Membuat Form untuk Menambah Kasus (View create) Edit dan Update Kasus ke Database Membuat View Form Edit Kasus (View edit) Menambahkan Method Destroy di Controller Menambahkan Tombol Hapus di View	
Membuat Fitur Siswa	Menambahkan Routing untuk Siswa Membuat Controller Siswa Menampilkan Daftar Siswa (Method index) Membuat View untuk Menampilkan Siswa (View index) Menambahkan Siswa Baru ke Dalam Database (Method create dan store) Membuat Form untuk Menambah Siswa (View create) Menambahkan Method destroy di Controller Menambahkan Tombol Hapus di View	
Membuat Fitur Walikelas	Menambahkan Routing untuk Walikelas	

Topik	Sub Topik	Status
	Membuat Controller Walikelas	
	Menampilkan Daftar Walikelas (Method index)	
	Membuat View untuk Menampilkan Walikelas (View index)	
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk `Siswa	
	Membuat Controller `Siswa	

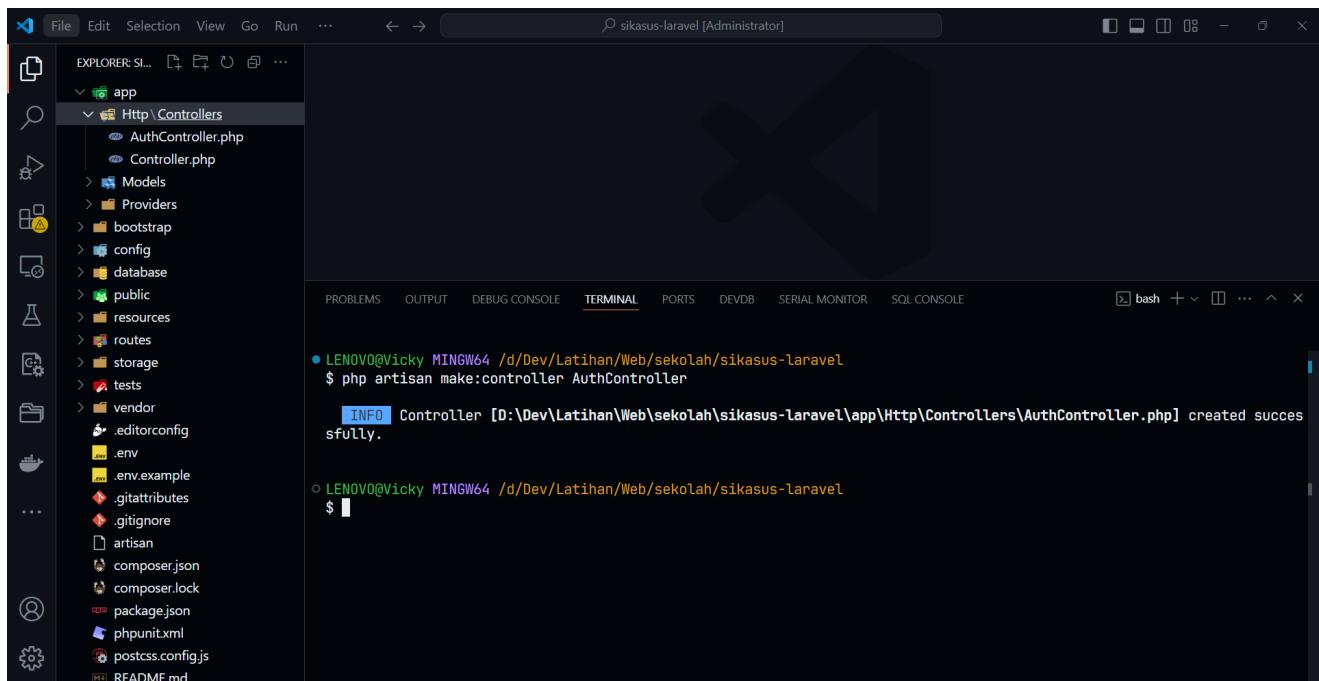
Menerapkan Autentikasi

Membuat Controller Auth

Langkah pertama adalah membuat controller untuk mengelola proses autentikasi. Jalankan perintah berikut di terminal:

```
php artisan make:controller AuthController
```

Controller ini akan digunakan untuk menampilkan halaman login, memproses login, dan menangani logout. Setelah perintah dijalankan, file baru bernama `AuthController.php` akan dibuat di folder `app/Http/Controllers`.



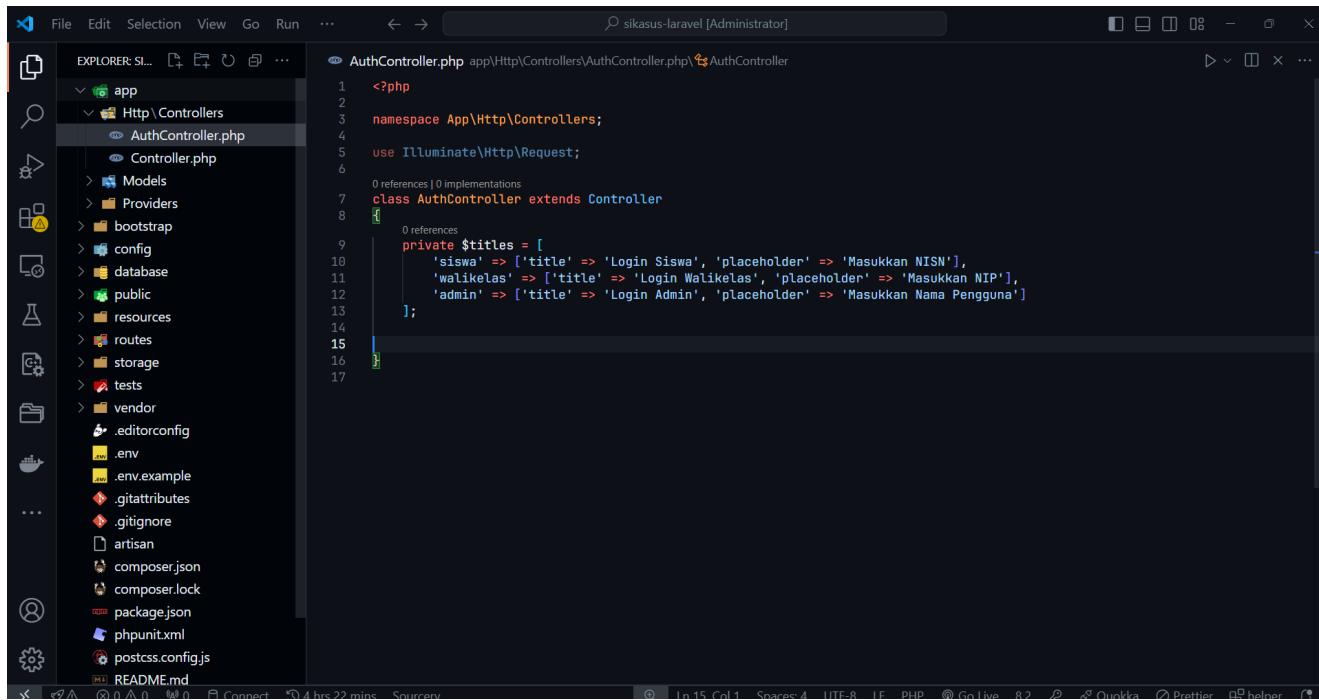
Menambahkan Fungsi di AuthController

Buka file `AuthController.php` yang telah dibuat, lalu tambahkan logika berikut:

```
private $titles = [
    'siswa' => ['title' => 'Login Siswa', 'placeholder' => 'Masukkan NISN'],
    'walikelas' => ['title' => 'Login Walikelas', 'placeholder' => 'Masukkan NIP'],
    'admin' => ['title' => 'Login Admin', 'placeholder' => 'Masukkan Nama Pengguna']
];
```

Penggunaan `private` pada variabel `$titles` berguna untuk membatasi akses ke variabel ini hanya di dalam class **AuthController**. Artinya, variabel ini tidak dapat diakses atau dimodifikasi dari luar class ini, yang memberikan kontrol penuh terhadap data tersebut dan meningkatkan keamanan aplikasi.

Disini kita membuat variabel untuk menentukan jenis jenis login sesuai rolenya.

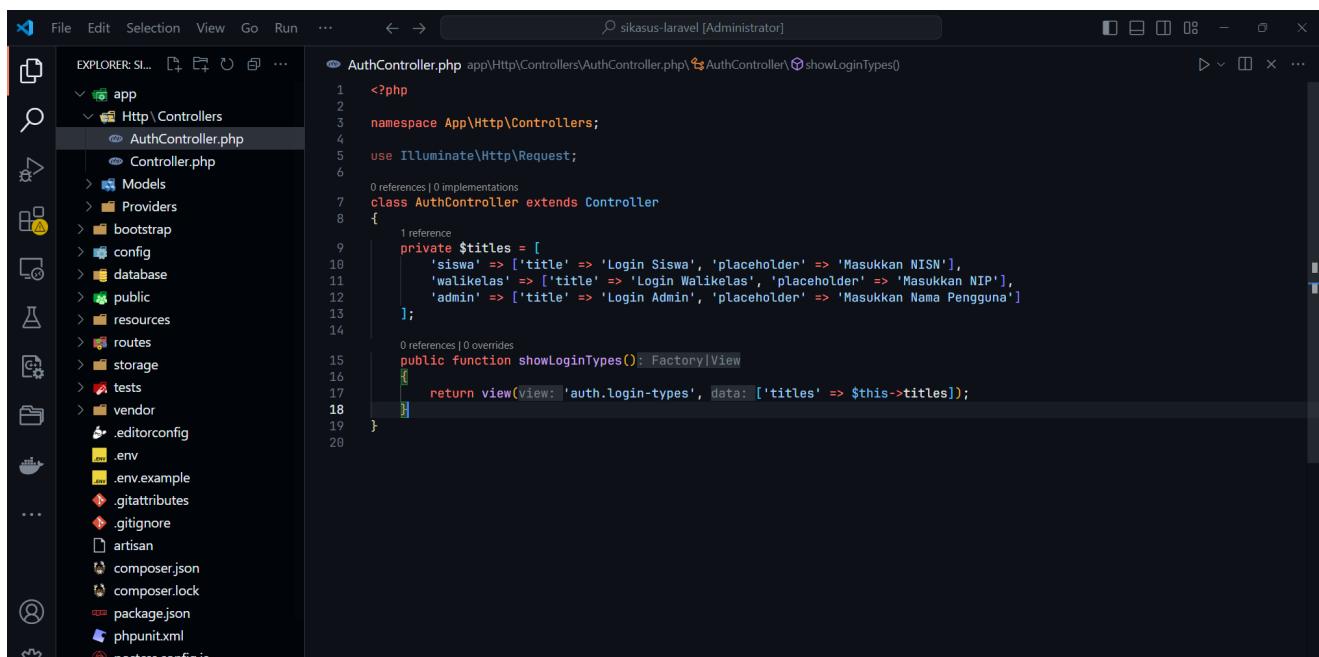


```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class AuthController extends Controller
{
    private $titles = [
        'siswa' => ['title' => 'Login Siswa', 'placeholder' => 'Masukkan NISN'],
        'walikelas' => ['title' => 'Login Walikelas', 'placeholder' => 'Masukkan NIP'],
        'admin' => ['title' => 'Login Admin', 'placeholder' => 'Masukkan Nama Pengguna']
    ];
}
```

Menampilkan Halaman Jenis Login

Langkah pertama adalah menambahkan fungsi `showLoginTypes` di dalam `AuthController`. Fungsi ini bertanggung jawab untuk menampilkan halaman pilihan jenis login (misalnya siswa, wali kelas, atau admin).

```
public function showLoginTypes()
{
    return view('auth.login-types', ['titles' => $this->titles]);
}
```



```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class AuthController extends Controller
{
    private $titles = [
        'siswa' => ['title' => 'Login Siswa', 'placeholder' => 'Masukkan NISN'],
        'walikelas' => ['title' => 'Login Walikelas', 'placeholder' => 'Masukkan NIP'],
        'admin' => ['title' => 'Login Admin', 'placeholder' => 'Masukkan Nama Pengguna']
    ];
    public function showLoginTypes(): Factory|View
    {
        return view('auth.login-types', ['titles' => $this->titles]);
    }
}
```

Penjelasan:

Fungsi `showLoginTypes` ini mengembalikan tampilan (`view`) dengan nama `auth.login-`

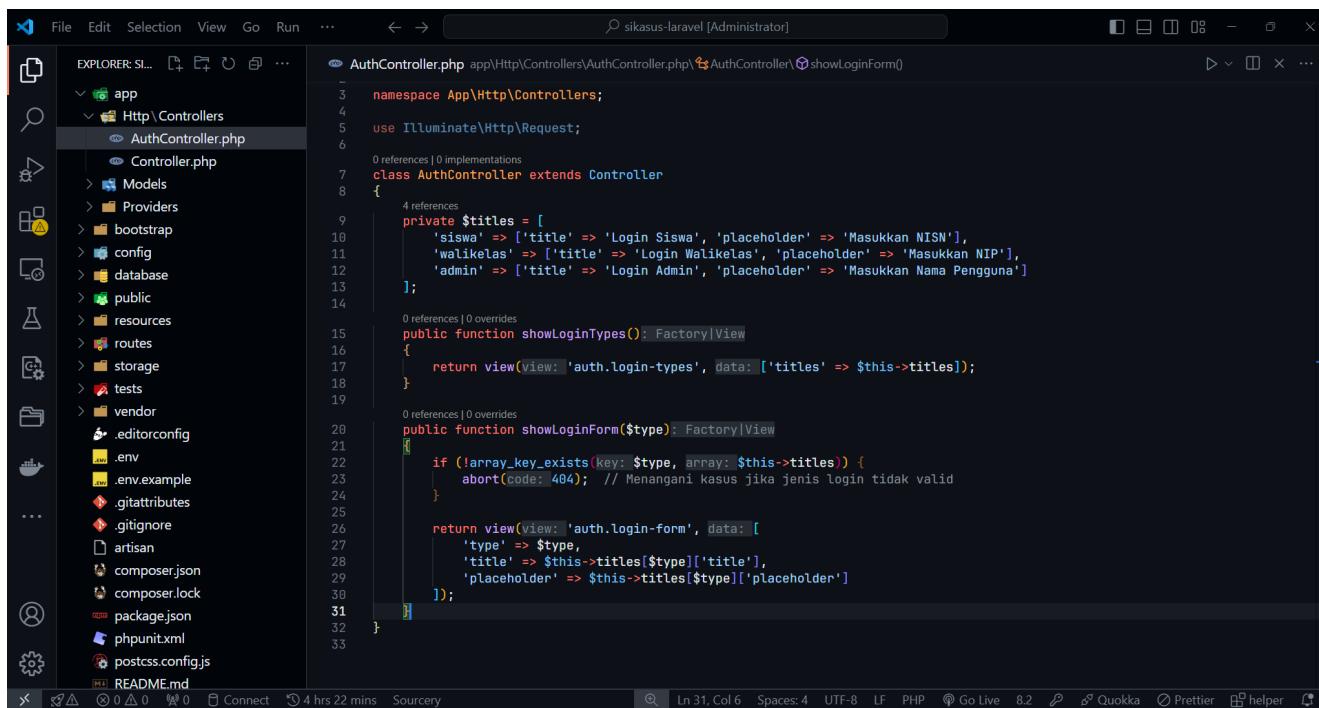
types . Di sini, kita juga mengirimkan data titles yang berisi jenis-jenis login yang akan ditampilkan di halaman tersebut.

Menampilkan Form Login Berdasarkan Jenis

Langkah berikutnya adalah menambahkan fungsi showLoginForm yang akan menampilkan form login sesuai dengan jenis login yang dipilih oleh pengguna.

```
public function showLoginForm($type)
{
    if (!array_key_exists($type, $this->titles)) {
        abort(404); // Menangani kasus jika jenis login tidak valid
    }

    return view('auth.login-form', [
        'type' => $type,
        'title' => $this->titles[$type]['title'],
        'placeholder' => $this->titles[$type]['placeholder']
    ]);
}
```



The screenshot shows the Visual Studio Code interface with the file 'AuthController.php' open in the main editor area. The file is located at 'app/Http/Controllers/AuthController.php'. The code implements a 'showLoginForm' method that checks if the requested login type exists in the '\$titles' array. If it does, it returns a view named 'auth.login-form' with the type, title, and placeholder information. The '\$titles' array contains three entries: 'siswa' (Login Siswa), 'walikelas' (Login Walikelas), and 'admin' (Login Admin). The 'showLoginTypes' method is also shown, which returns a view named 'auth.login-types' with the '\$titles' array.

```
namespace App\Http\Controllers;
use Illuminate\Http\Request;

class AuthController extends Controller
{
    private $titles = [
        'siswa' => ['title' => 'Login Siswa', 'placeholder' => 'Masukkan NISN'],
        'walikelas' => ['title' => 'Login Walikelas', 'placeholder' => 'Masukkan NIP'],
        'admin' => ['title' => 'Login Admin', 'placeholder' => 'Masukkan Nama Pengguna']
    ];

    public function showLoginTypes(): Factory|View
    {
        return view('auth.login-types', ['titles' => $this->titles]);
    }

    public function showLoginForm($type): Factory|View
    {
        if (!array_key_exists($type, $this->titles)) {
            abort(404); // Menangani kasus jika jenis login tidak valid
        }

        return view('auth.login-form', [
            'type' => $type,
            'title' => $this->titles[$type]['title'],
            'placeholder' => $this->titles[$type]['placeholder']
        ]);
    }
}
```

Penjelasan:

Pada fungsi ini, kita memeriksa apakah jenis login yang diminta ada dalam array \$this->titles . Jika jenis login tidak ditemukan, maka akan ditampilkan halaman 404. Jika valid, fungsi ini akan mengirimkan data untuk digunakan di tampilan auth.login-form , termasuk tipe login, judul halaman, dan placeholder untuk input username.

Fungsi Validasi di Laravel

Laravel menyediakan berbagai aturan validasi untuk memverifikasi input pengguna dengan cara yang mudah dan efisien. Aturan validasi ini bisa digunakan baik dalam form request maupun di dalam controller. Berikut adalah beberapa aturan validasi yang sering digunakan:

1. required

Aturan ini memastikan bahwa kolom yang divalidasi tidak boleh kosong. Jika kolom tidak diisi, maka akan muncul pesan error.

Contoh:

```
'username' => 'required'
```

2. string

Aturan ini memastikan bahwa kolom berisi data bertipe string (teks). Ini penting untuk memverifikasi bahwa input yang diterima sesuai dengan format yang diinginkan.

Contoh:

```
'username' => 'string'
```

3. numeric

Aturan ini memastikan bahwa kolom hanya berisi angka. Misalnya, jika Anda menginginkan ID atau nomor telepon, aturan ini akan memastikan hanya angka yang diterima.

Contoh:

```
'phone_number' => 'numeric'
```

4. in

Aturan ini memastikan bahwa nilai kolom adalah salah satu dari sekumpulan nilai yang diperbolehkan. Biasanya digunakan untuk menentukan nilai tertentu pada dropdown atau pilihan tertentu.

Contoh:

```
'role' => 'required|in:siswa,walikelas,admin'
```

5. unique

Aturan ini memastikan bahwa nilai kolom unik di dalam tabel tertentu, berguna untuk memastikan bahwa username atau email tidak terduplikasi di database.

Contoh:

```
'email' => 'required|unique:users,email'
```

6. email

Aturan ini memastikan bahwa kolom berisi format alamat email yang valid. Ini digunakan untuk memverifikasi bahwa input adalah email yang benar.

Contoh:

```
'email' => 'required|email'
```

7. confirmed

Aturan ini digunakan untuk memverifikasi bahwa dua kolom memiliki nilai yang sama. Ini biasanya digunakan pada form registrasi untuk memastikan bahwa kolom `password` dan `password_confirmation` memiliki nilai yang sama.

Contoh:

```
'password' => 'required|confirmed'
```

8. min dan max

Aturan `min` dan `max` digunakan untuk memvalidasi panjang karakter dari kolom. Anda bisa menentukan jumlah minimal dan maksimal karakter yang diizinkan.

Contoh:

```
'password' => 'required|min:8|max:16'
```

9. date

Aturan ini memastikan bahwa input yang diterima memiliki format tanggal yang valid.

Contoh:

```
'birthdate' => 'required|date'
```

10. exists

Aturan ini memastikan bahwa nilai kolom ada dalam tabel tertentu. Biasanya digunakan untuk memverifikasi bahwa input merujuk ke data yang ada di database.

Contoh:

```
'user_id' => 'required|exists:users,id'
```

Proses Login

Setelah form login ditampilkan, kita perlu menangani proses login melalui fungsi `login`.

```
public function login(Request $request)
{
    $request->validate([
        'type' => 'required|in:siswa,walikelas,admin',
        'username' => 'required|string',
        'password' => 'required|string'
    ]);

    $type = $request->type;
    $username = $request->username;
    $password = $request->password;

    $user = $this->authenticateUser($type, $username, $password);

    if ($user) {
        session([
            'user' => $user,
            'role' => $type
        ]);

        $request->session()->regenerate();
        return redirect()->intended('dashboard');
    }

    // Menampilkan pesan error jika login gagal
    $usernameLabel = match ($type) {
        'siswa' => 'NISN',
        'walikelas' => 'NIP',
        default => 'Username'
    };

    throw ValidationException::withMessages([
        'username' => ["{$usernameLabel} atau kata sandi salah. Silakan coba lagi."],
    ]);
}

private function authenticateUser($type, $username, $password)
{
    $model = match ($type) {
        'siswa' => Siswa::class,
        'walikelas' => Walikelas::class,
        'admin' => User::class,
        default => null
    };

    if (!$model) {
        return null;
    }
}
```

```

$field = $this->getUsernameField($type);
$user = $model::where($field, $username)->first();

// Verifikasi login berdasarkan tipe
if ($type === 'siswa' && $user && $user->nisn === $password) {
    return $user;
}

if ($type === 'walikelas' && $user && $user->nip === $password) {
    return $user;
}

if ($type === 'admin' && $user && \Hash::check($password, $user->password)) {
    return $user;
}

return null;
}

private function getUsernameField($type)
{
    return match ($type) {
        'siswa' => 'nisn',
        'walikelas' => 'nip',
        'admin' => 'username'
    };
}

```

The screenshot shows a code editor interface with the following details:

- File Path:** app\Http\Controllers\AuthController.php
- Code Content:**

```

class AuthController extends Controller
{
    public function login(Request $request): mixed|RedirectResponse
    {
        $request->validate([
            'type' => 'required|in:siswa,walikelas,admin',
            'username' => 'required|string',
            'password' => 'required|string'
        ]);

        $type = $request->type;
        $username = $request->username;
        $password = $request->password;

        $user = $this->authenticateUser($type, $username, $password);

        if ($user) {
            session([
                'user' => $user,
                'role' => $type
            ]);

            $request->session()->regenerate();
            return redirect()->intended(default: 'dashboard');
        }

        // Menampilkan pesan error jika login gagal
        $usernameLabel = match ($type) {
            'siswa' => 'NISN',
            'walikelas' => 'NIP',
            default => 'Username'
        };

        throw ValidationException::withMessages(messages: [
            'username' => ["{$usernameLabel} atau kata sandi salah. Silakan coba lagi."]
        ]);
    }
}

```

Penjelasan:

1. Fungsi login(Request \$request) :

- **Validasi Input:** Fungsi ini dimulai dengan melakukan validasi terhadap data yang diterima dari request. Tiga parameter yang divalidasi adalah `type`, `username`, dan `password`. `type` harus berupa salah satu dari `siswa`, `walikelas`, atau `admin`. `username` dan `password` harus berupa string dan tidak boleh kosong.
- **Proses Autentikasi Pengguna:** Setelah validasi, fungsi ini akan mengambil nilai `type`, `username`, dan `password`, kemudian memanggil metode `authenticateUser` untuk memeriksa kredensial pengguna.
- **Menyimpan Data Pengguna di Sesi:** Jika pengguna berhasil diautentikasi (ditemukan dalam database), data pengguna dan jenis peran (`role`) disimpan dalam sesi menggunakan `session()`. Sesi kemudian diperbarui dengan `session() -> regenerate()`, dan pengguna diarahkan ke halaman `dashboard`.
- **Menangani Kesalahan Login:** Jika login gagal, maka akan ditampilkan pesan kesalahan yang menjelaskan bahwa `username` atau `password` salah. Jenis label untuk `username` tergantung pada tipe login yang dipilih (misalnya `NISN` untuk siswa, `NIP` untuk `walikelas`).

2. Fungsi authenticateUser(\$type, \$username, \$password) :

- Fungsi ini bertugas untuk memverifikasi kredensial pengguna. Berdasarkan tipe login (`siswa`, `walikelas`, atau `admin`), fungsi ini mencari pengguna di tabel yang sesuai.
- **Verifikasi untuk Setiap Tipe:** Setiap tipe memiliki cara verifikasi yang berbeda:
 - Untuk `siswa`, password dibandingkan langsung dengan `nisn`.
 - Untuk `walikelas`, password dibandingkan langsung dengan `nip`.
 - Untuk `admin`, password diverifikasi menggunakan hashing dengan `Hash::check()`.
- Jika kredensial cocok, pengguna yang ditemukan akan dikembalikan.

3. Fungsi getUsernameField(\$type) :

- Fungsi ini mengembalikan nama kolom yang digunakan untuk memverifikasi `username` berdasarkan tipe login. Misalnya, untuk `siswa`, digunakan kolom `nisn`; untuk `walikelas`, digunakan kolom `nip`; dan untuk `admin`, digunakan kolom `username`.

Dengan menggunakan struktur ini, sistem login dapat menangani berbagai jenis pengguna dan memverifikasi kredensial mereka dengan cara yang sesuai.

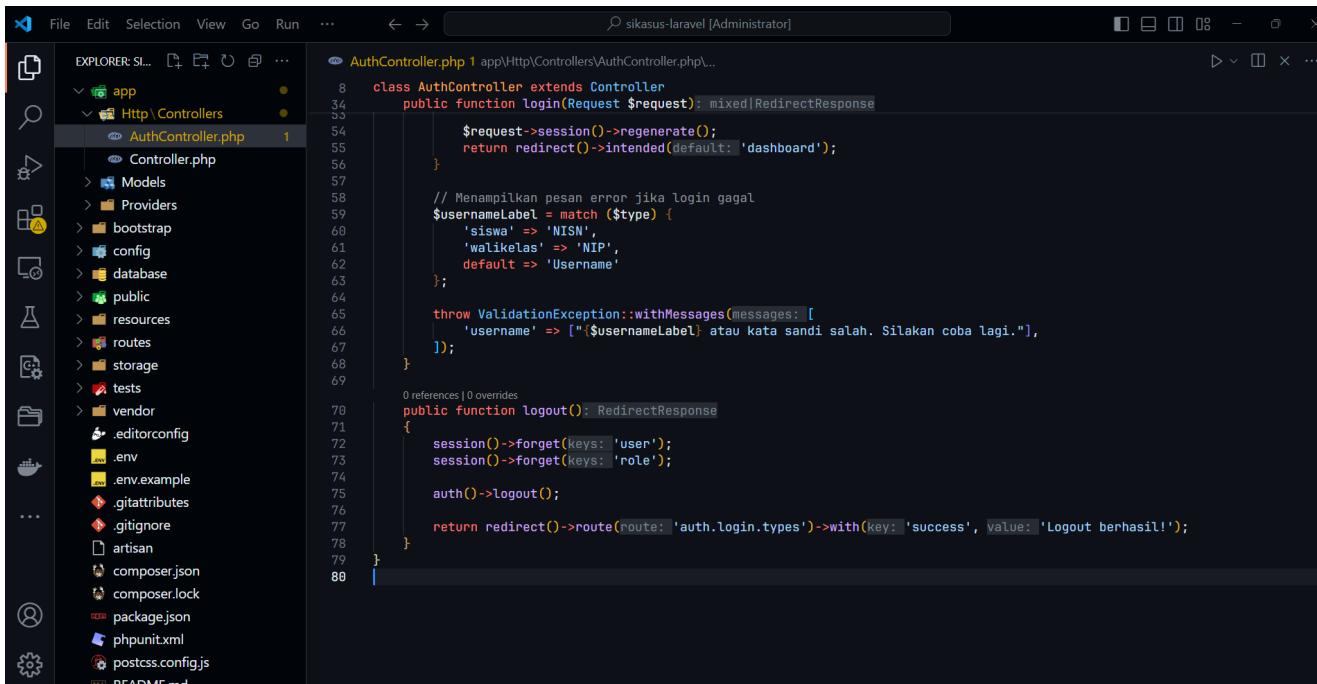
Proses Logout

Untuk logout, kita akan membuat fungsi `logout` untuk menghapus sesi pengguna dan mengarahkan kembali ke halaman login.

```
public function logout()
{
    session()->forget('user');
    session()->forget('role');

    auth()->logout();

    return redirect()->route('auth.login.types')->with('success', 'Logout
berhasil!');
}
```



Penjelasan:

Fungsi ini bertugas untuk menghapus data sesi pengguna, termasuk data `user` dan `role`. Setelah logout, pengguna akan diarahkan kembali ke halaman pilihan jenis login (`auth.login.types`) dengan pesan sukses yang memberitahukan bahwa proses logout berhasil.

Membuat Routing untuk Autentikasi

Terakhir, kita perlu menambahkan routing di file `routes/web.php` untuk menghubungkan fungsi-fungsi ini dengan URL yang sesuai.

```
Route::prefix('auth')
    ->name('auth.')
    ->controller(AuthController::class)
    ->group(function () {
        Route::get('/login', 'showLoginTypes')->name('login.types');
        Route::get('/login/{type}', 'showLoginForm')->name('login.form');
```

```

        Route::post('/login', 'login')->name('login.submit');
        Route::get('/logout', 'logout')->name('logout');
    });

```

```

web.php routes\web.php
1 <?php
2
3 use App\Http\Controllers\AuthController;
4 use Illuminate\Support\Facades\Route;
5
6 Route::prefix('auth')
7     ->name('value: auth.')
8     ->controller(AuthController::class)
9     ->group(function () {
10         Route::get('/Login', action: 'showLoginTypes')->name(name: 'Login.types');
11         Route::get('/Login/{type}', action: 'showLoginForm')->name(name: 'Login.form');
12         Route::post('/Login', action: 'Login')->name(name: 'Login.submit');
13         Route::get('/Logout', action: 'Logout')->name(name: 'Logout');
14     });
15

```

Penjelasan:

Di sini, kita mendefinisikan routing untuk menampilkan halaman jenis login, form login, proses login, dan proses logout. Setiap route menggunakan controller yang sudah dibuat sebelumnya (`AuthController::class`), dengan metode yang sesuai.

Membuat View untuk Menampilkan Halaman Login

Pengenalan Singkat tentang Blade View

Blade adalah template engine bawaan Laravel yang dirancang untuk menyederhanakan pembuatan tampilan dinamis dalam aplikasi. Berikut adalah beberapa fitur utama Blade:

1. Ekspresi Blade:

- Sintaks `{{ }}` digunakan untuk mencetak data pada tampilan dengan aman, misalnya:

```

{{ $variable }}

```

- Blade secara otomatis melakukan escaping terhadap karakter HTML untuk mencegah serangan XSS (Cross-Site Scripting).

2. Kondisional (@if):

- Kondisional digunakan untuk menampilkan bagian tertentu dari tampilan berdasarkan kondisi yang diberikan.

```
@if($role == 'admin')
    <p>Selamat datang, Admin!</p>
@elseif($role == 'guru')
    <p>Selamat datang, Guru!</p>
@else
    <p>Selamat datang, Siswa!</p>
@endif
```

3. Perulangan (@foreach , @for , @while):

- Mempermudah iterasi data untuk menampilkan daftar atau elemen dinamis.

```
@foreach($items as $item)
    <li>{{ $item }}</li>
@endforeach
```

4. Penggunaan Komponen Blade:

- Blade mendukung pembuatan komponen untuk membuat tampilan yang dapat digunakan ulang. Misalnya:

```
<x-alert type="success" message="Berhasil menyimpan data!" />
```

5. Direktif Blade Lainnya:

- @extends : Untuk mewarisi layout utama.

```
@extends('layouts.app')
```

- @section dan @yield : Untuk mendefinisikan dan menampilkan bagian tertentu dalam layout.

```
@section('content')
    <p>Isi konten halaman.</p>
@endsection
```

- @include : Menyisipkan file Blade lain dalam tampilan.

```
@include('partials.header')
```

6. Routing dalam Blade:

- Menggunakan helper `route()` untuk membuat tautan dinamis berdasarkan nama rute.

```
<a href="{{ route('dashboard') }}">Dashboard</a>
```

7. Manajemen Aset:

- Blade mendukung fungsi seperti `asset()` untuk memuat file CSS atau JavaScript.

```
<link rel="stylesheet" href="{{ asset('css/style.css') }}">
```

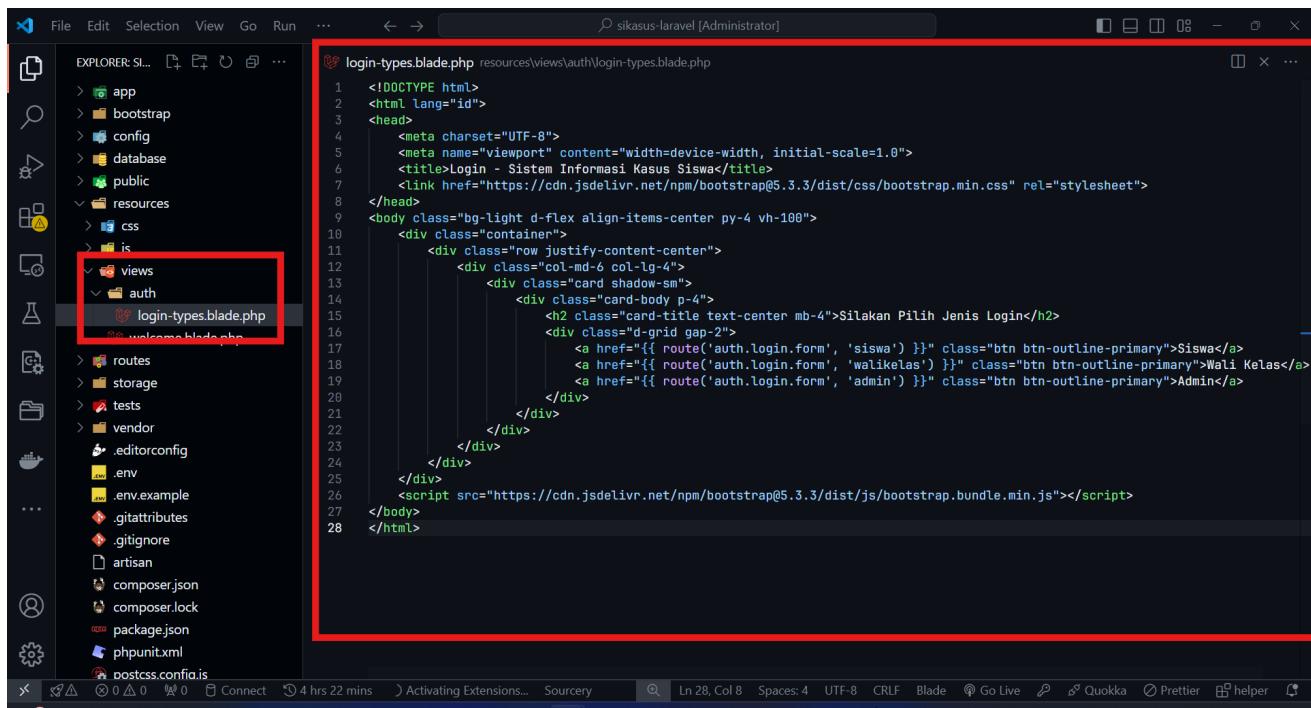
Untuk informasi lebih lengkap tentang Blade dan penggunaannya, silakan kunjungi dokumentasi resmi Laravel di [Blade Documentation](#).

Halaman Pilihan Login

Buat file baru dengan nama `resources/views/auth/login-types.blade.php`. Isi file ini dengan kode berikut:

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Login – Sistem Informasi Kasus Siswa</title>
    <link
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
        rel="stylesheet">
</head>
<body class="bg-light d-flex align-items-center py-4 vh-100">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-6 col-lg-4">
                <div class="card shadow-sm">
                    <div class="card-body p-4">
                        <h2 class="card-title text-center mb-4">Silakan
                        Pilih Jenis Login</h2>
                        <div class="d-grid gap-2">
                            <a href="{{ route('auth.login.form', 'siswa') }}"
                                class="btn btn-outline-primary">Siswa</a>
                            <a href="{{ route('auth.login.form', 'wali_kelas') }}"
                                class="btn btn-outline-primary">Wali Kelas</a>
                            <a href="{{ route('auth.login.form', 'admin') }}"
                                class="btn btn-outline-primary">Admin</a>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</body>
```

```
        </div>
    </div>
</div>
</div>
</div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```



Penjelasan:

- Halaman ini memungkinkan pengguna untuk memilih jenis login: Siswa, Wali Kelas, atau Admin.
 - Setiap jenis login mengarah ke form login yang sesuai dengan jenis yang dipilih.
 - Menggunakan framework Bootstrap untuk styling dan layout halaman.

Membuat Form Login

Buat file baru dengan nama `resources/views/auth/login-form.blade.php` dan isi dengan kode berikut:

```
<!DOCTYPE html>
<html lang="id">
<head>
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Login - Sistem Informasi Kasus Siswa</title>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light d-flex align-items-center py-4 vh-100">
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-md-6 col-lg-4">
                <div class="card shadow-sm">
                    <div class="card-body p-4">
                        <h2 class="card-title text-center mb-4">{{ $title }}</h2>
                        <form method="POST" action="{{ route('auth.login.submit') }}" autocomplete="on">
                            @csrf
                            <input type="hidden" name="type" value="{{ $type }}>
                            <div class="mb-3">
                                <input type="text" name="username" class="form-control" placeholder="{{ $placeholder }}" value="{{ old('username') }}" autofocus required>
                            </div>
                            <div class="mb-3">
                                <input type="password" name="password" class="form-control" placeholder="Password" required>
                            </div>
                            @if ($errors->any())
                                <div class="alert alert-danger mb-3">
                                    @foreach ($errors->all() as $error)
                                        {{ $error }}
                                    @endforeach
                                </div>
                            @endif
                            <div class="d-grid gap-2">
                                <button type="submit" class="btn btn-primary">Login</button>
                                <a href="{{ route('auth.login.types') }}" class="btn btn-outline-secondary">Kembali</a>
                            </div>
                        </form>
                </div>
            </div>
        </div>
    </div>
</body>
```

```

        </div>
    </div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.m
in.js"></script>
</body>
</html>

```

The screenshot shows a code editor interface with the following details:

- Project Structure:** The left sidebar shows the project structure under "EXPLORER". It includes the "app", "bootstrap", "config", "database", "public", "resources" (containing "css", "js", "views" with "auth", "login-form.blade.php", "login-types.blade.php", and "welcome.blade.php"), "routes", "storage", "tests", "vendor", ".editorconfig", ".env", ".env.example", ".gitattributes", ".gitignore", "artisan", "composer.json", "composer.lock", and "package.json".
- File Content:** The main editor area displays the "login-form.blade.php" file content. The code uses Bootstrap 5 styling and Laravel Blade templating. It features a centered card for user input, handling POST requests to the "auth.login.submit" route, and displaying validation errors if any occur.
- Editor Tools:** The bottom of the editor shows various toolbars and status indicators, including "Convert", "4 hrs 20 min", "Source", "In 52 Col 1", "Spaces 4", "IME 0", "Blade", "Go To Line", "QuickFix", "Prettier", and "helpful".

Penjelasan:

- Halaman ini digunakan untuk form login berdasarkan jenis yang dipilih sebelumnya (Siswa, Wali Kelas, atau Admin).
- Form ini mengirimkan data ke route `auth.login.submit` untuk memproses login.
- Menggunakan Bootstrap untuk styling dan layout form login. Jika terjadi kesalahan saat login, pesan error akan ditampilkan di bawah form input.

localhost:8000/auth/login

Silakan Pilih Jenis Login

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration		✓
	1. Migration Kasus	✓
	2. Migration Siswa	✓
	3. Migration Kelas	✓
	4. Migration Walikelas	✓
Menambahkan Mass Assignment		✓
	1. Model Kasus	✓
	2. Model Siswa	✓
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓
	Menampilkan Halaman Jenis Login	✓
	Menampilkan Form Login Berdasarkan Jenis	✓

Topik	Sub Topik	Status
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		
	Membuat Middleware untuk Autentikasi	
	Membuat Middleware untuk Role	
	Registrasi Middleware	
Membuat Blade Layout		
Membuat Tampilan Homepage		
Membuat Tampilan Dashboard Setelah Login		
Membuat Fitur Kasus		
	Menambahkan Routing untuk Kasus	
	Membuat Controller Kasus	
	Menampilkan Daftar Kasus (Method index)	
	Membuat View untuk Menampilkan Kasus (View index)	
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kasus (View create)	
	Edit dan Update Kasus ke Database	
	Membuat View Form Edit Kasus (View edit)	
	Menambahkan Method Destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Siswa		
	Menambahkan Routing untuk Siswa	
	Membuat Controller Siswa	
	Menampilkan Daftar Siswa (Method index)	
	Membuat View untuk Menampilkan Siswa (View index)	

Topik	Sub Topik	Status
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Siswa (View create)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Walikelas		
	Menambahkan Routing untuk Walikelas	
	Membuat Controller Walikelas	
	Menampilkan Daftar Walikelas (Method index)	
	Membuat View untuk Menampilkan Walikelas (View index)	
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	

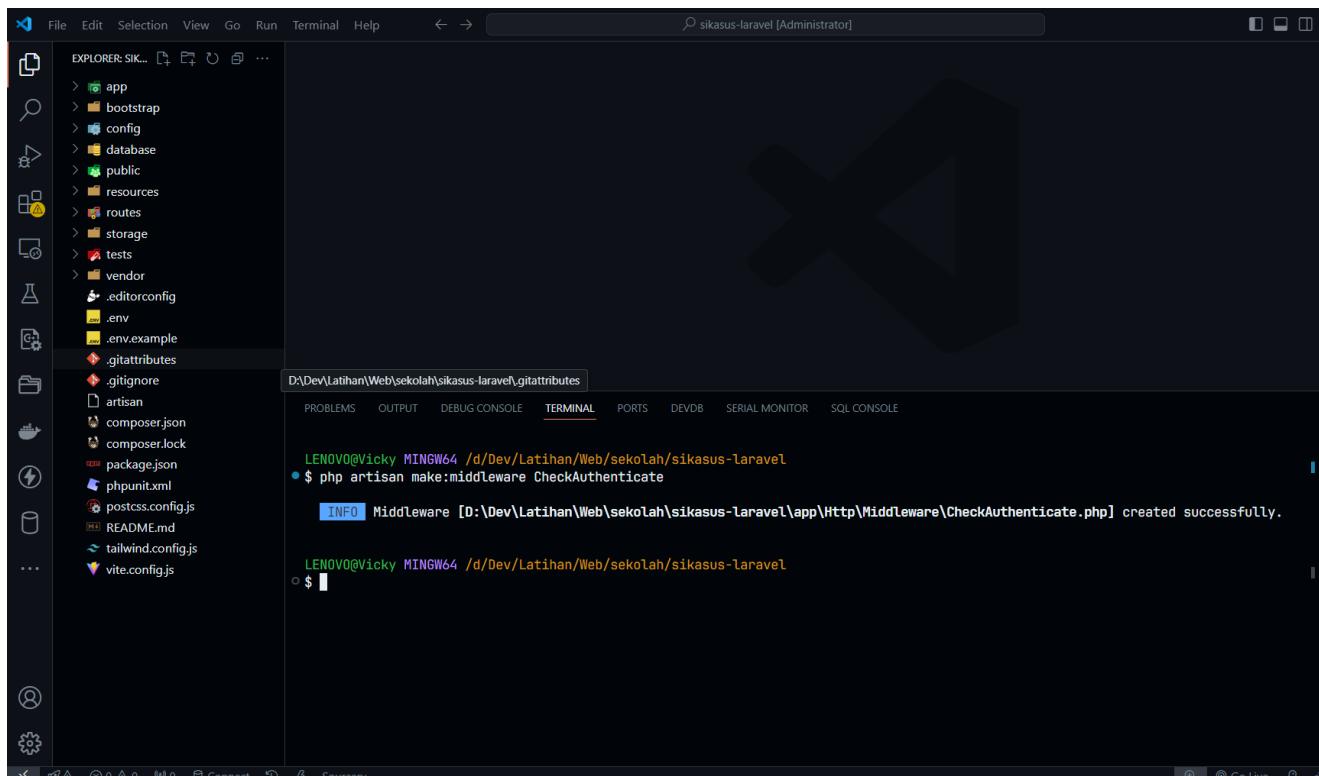
Topik	Sub Topik	Status
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk 'Siswa'	
	Membuat Controller 'Siswa'	

Membuat Middleware

Membuat Middleware untuk Autentikasi

Langkah pertama adalah membuat middleware yang akan mengecek apakah pengguna sudah login atau belum. Untuk itu, kita akan membuat middleware `CheckAuthenticate`. Jalankan perintah berikut untuk membuat middleware:

```
php artisan make:middleware CheckAuthenticate
```



The screenshot shows a Microsoft Visual Studio Code interface. On the left is the Explorer sidebar displaying the project structure of a Laravel application. The terminal tab is active at the bottom, showing the command `$ php artisan make:middleware CheckAuthenticate` being run and the output indicating success: `Middleware [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\app\Http\Middleware\CheckAuthenticate.php] created successfully.`

Setelah middleware berhasil dibuat, buka file `app/Http/Middleware/CheckAuthenticate.php`.

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with a red box highlighting the 'Http' folder under 'app'. Inside 'Http' is a 'Middleware' folder containing a file named 'CheckAuthenticate.php'. The main editor area displays the code for this file. Below the editor is the Terminal panel, which shows the command 'php artisan make:middleware CheckAuthenticate' being run and its successful execution.

```
CheckAuthenticate.php app\Http\Middleware\CheckAuthenticate.php...
namespace App\Http\Middleware;
use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;
...
class CheckAuthenticate
{
    /**
     * Handle an incoming request.
     *
     * @param \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response) $next
     */
    public function handle(Request $request, Closure $next): Response
    {
        return $next($request);
    }
}
```

```
LENVOO@Vicky MINGW64 /d/Dev/Latihan/Web/sekolah/sikasus-laravel
$ php artisan make:middleware CheckAuthenticate
INFO Middleware [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\app\Http\Middleware\CheckAuthenticate.php] created successfully.

LENVOO@Vicky MINGW64 /d/Dev/Latihan\Web/sekolah/sikasus-laravel
$
```

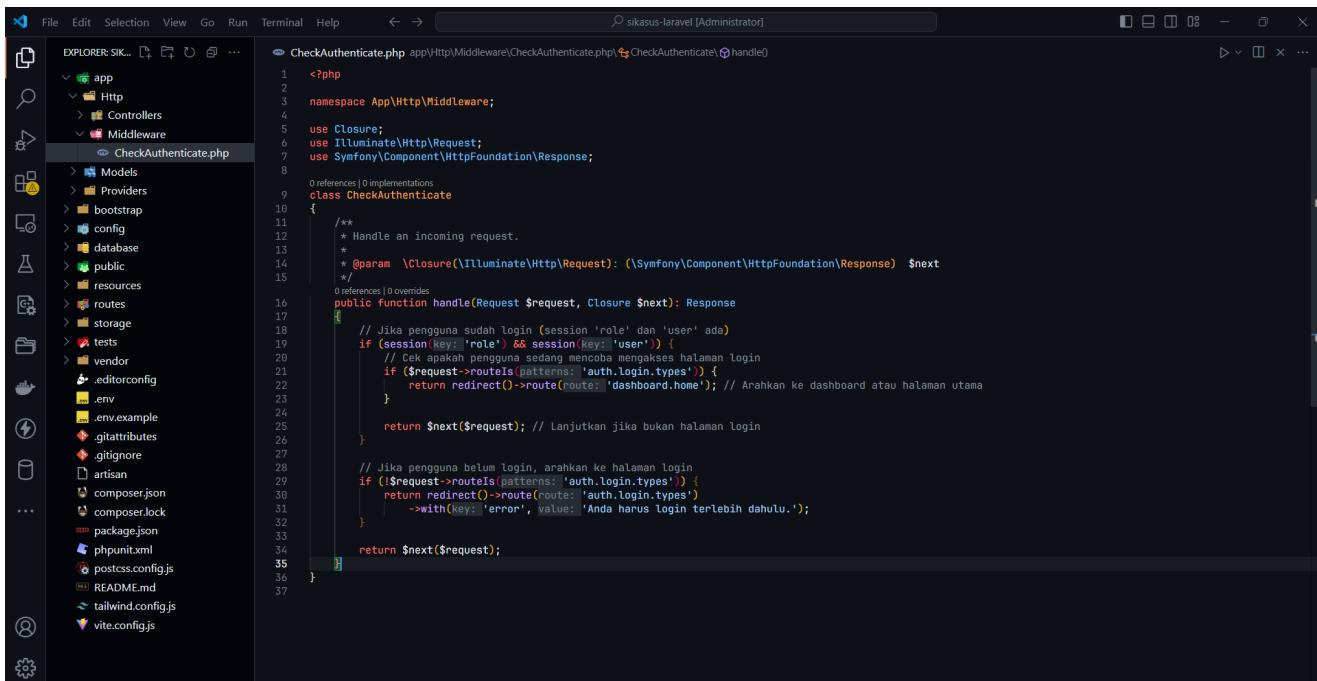
kemudian tambahkan kode berikut:

```
public function handle(Request $request, Closure $next): Response
{
    // Jika pengguna sudah login (session 'role' dan 'user' ada)
    if (session('role') && session('user')) {
        // Cek apakah pengguna sedang mencoba mengakses halaman login
        if ($request->routeIs('auth.login.types')) {
            return redirect()->route('dashboard.home'); // Arahkan ke
            dashboard atau halaman utama
        }

        return $next($request); // Lanjutkan jika bukan halaman login
    }

    // Jika pengguna belum login, arahkan ke halaman login
    if (!$request->routeIs('auth.login.types')) {
        return redirect()->route('auth.login.types')
            ->with('error', 'Anda harus login terlebih dahulu.');
    }

    return $next($request);
}
```



```
CheckAuthenticate.php app\Http\Middleware\CheckAuthenticate.php CheckAuthenticate handle
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use Illuminate\Http\Request;
7 use Symfony\Component\HttpFoundation\Response;
8
9 class CheckAuthenticate
10 {
11     /**
12      * Handle an incoming request.
13      *
14      * @param \Closure(\Illuminate\Http\Request) $next
15      */
16     public function handle(Request $request, Closure $next): Response
17     {
18         // Jika pengguna sudah login (session 'role' dan 'user' ada)
19         if ($request->session('role') && $request->session('user')) {
20             // Cek apakah pengguna sedang mencoba mengakses halaman login
21             if ($request->routeIs('auth.login.types')) {
22                 return redirect()>route('dashboard.home'); // Arahkan ke dashboard atau halaman utama
23             }
24
25             // Jika pengguna belum login, arahkan ke halaman login
26             if (!$request->routeIs('auth.login.types')) {
27                 return redirect()>route('auth.login.types')
28                     >with(['error' => 'Anda harus login terlebih dahulu.']);
29             }
30
31             return $next($request); // Lanjutkan jika bukan halaman login
32         }
33
34         return $next($request);
35     }
36
37 }
```

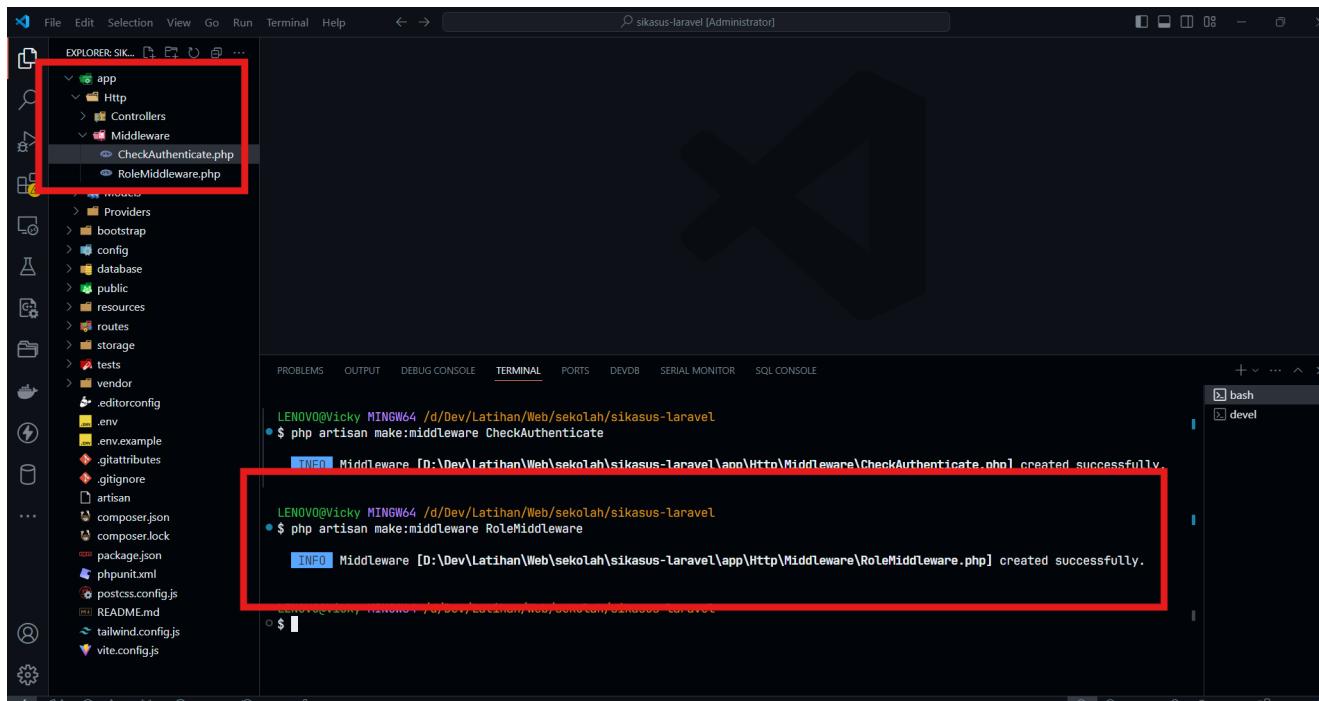
Penjelasan:

- Middleware ini memeriksa apakah sesi `role` dan `user` ada, yang menunjukkan bahwa pengguna sudah login.
- Jika pengguna sudah login, mereka dapat melanjutkan ke halaman yang diminta, kecuali halaman login.
- Jika pengguna belum login dan mencoba mengakses halaman selain login, mereka akan diarahkan ke halaman login.

Membuat Middleware untuk Role

Langkah berikutnya adalah membuat middleware yang akan memeriksa peran pengguna (role) sebelum mengakses halaman tertentu. Jalankan perintah berikut untuk membuat middleware `RoleMiddleware` :

```
php artisan make:middleware RoleMiddleware
```



Setelah middleware berhasil dibuat, buka file `app/Http/Middleware/RoleMiddleware.php`, kemudian tambahkan kode berikut:

```
public function handle(Request $request, Closure $next, ...$roles)
{
    $userRole = session('role');

    if (!in_array($userRole, $roles)) {
        $redirectRoute = $userRole === 'siswa' ? 'siswa.dashboard' :
'dashboard.home';

        return redirect()->route($redirectRoute)
            ->with('error', 'Anda tidak memiliki akses.');
    }

    return $next($request);
}
```

```

1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use Illuminate\Http\Request;
7 use Symfony\Component\HttpFoundation\Response;
8
9
10 class RoleMiddleware
11 {
12     /**
13      * Handle an incoming request.
14      *
15      * @param \Closure(\Illuminate\Http\Request: (\Symfony\Component\HttpFoundation\Response) $next
16      */
17     public function handle(Request $request, Closure $next, ...$roles): RedirectResponse|Response
18     {
19         $userRole = session('role');
20
21         if (!in_array($userRole, $roles)) {
22             $redirectTo = $userRole === 'siswa' ? 'siswa.dashboard' : 'dashboard.home';
23
24             return redirect()->route($redirectTo)
25                         ->with('error', 'Anda tidak memiliki akses.');
26         }
27
28         return $next($request);
29     }
30 }

```

Penjelasan:

- Middleware ini memeriksa peran pengguna (`role`) yang disimpan dalam sesi.
- Jika peran pengguna tidak sesuai dengan peran yang diizinkan untuk mengakses halaman tersebut, maka pengguna akan diarahkan ke halaman yang sesuai berdasarkan peran mereka.
- Jika peran cocok, maka pengguna dapat melanjutkan akses ke halaman yang diminta.

Registrasi Middleware

Setelah membuat middleware, langkah selanjutnya adalah mendaftarkannya dalam aplikasi Laravel. Untuk melakukannya, buka file `bootstrap/app.php` kemudian kita perlu mengkonfigurasi alias middleware yang telah dibuat.

```

1 <?php
2
3 use Illuminate\Foundation\Application;
4 use Illuminate\Foundation\Configuration\Exceptions;
5 use Illuminate\Foundation\Configuration\Middleware;
6
7 return Application::configure(basePath: dirname(path: __DIR__))
8     ->withRouting(
9         web: __DIR__.'/../routes/web.php',
10        commands: __DIR__.'/../routes/console.php',
11        health: '/up',
12    )
13     ->withMiddleware(callback: function (Middleware $middleware): void {
14         //
15     })
16     ->withExceptions(using: function (Exceptions $exceptions): void {
17         //
18     })
19     ->create();

```

Pada kode middleware ini didaftarkan dalam aplikasi melalui konfigurasi berikut:

```

use App\Http\Middleware\CheckAuthenticate;
use App\Http\Middleware\RoleMiddleware;

// Mendaftarkan alias middleware di aplikasi
$middleware->alias([
    'auth' => CheckAuthenticate::class, // Alias untuk middleware autentikasi
    'role' => RoleMiddleware::class, // Alias untuk middleware role
]);

```

```

EXPLORER: SIK...
File Edit Selection View Go Run Terminal Help
app.php bootstrap/app.php
1 <?php
2
3 use Illuminate\Foundation\Application;
4 use Illuminate\Foundation\Configuration\Exceptions;
5 use Illuminate\Foundation\Configuration\Middleware;
6
7 use App\Http\Middleware\CheckAuthenticate;
8 use App\Http\Middleware\RoleMiddleware;
9
10 return Application::configure(basePath: dirname(path: __DIR__))
11     ->withRouting(
12         web: __DIR__ . '/../routes/web.php',
13         commands: __DIR__ . '/../routes/console.php',
14         health: '/up',
15     )
16     ->withMiddleware(callback: function (Middleware $middleware): void {
17
18         // Mendaftarkan alias middleware di aplikasi
19         $middleware->alias(aliases: [
20             'auth' => CheckAuthenticate::class, // Alias untuk middleware autentikasi
21             'role' => RoleMiddleware::class, // Alias untuk middleware role
22         ]);
23     })
24     ->withExceptions(using: function (Exceptions $exceptions): void {
25         //
26     })->create();
27

```

Penjelasan:

- `withMiddleware()` adalah metode untuk mendaftarkan middleware kustom ke aplikasi Laravel.
- Di dalam `withMiddleware()`, Anda mendefinisikan alias untuk masing-masing middleware yang telah Anda buat. Dengan mendefinisikan alias 'auth' untuk `CheckAuthenticate` dan 'role' untuk `RoleMiddleware`, Anda akan dapat menggunakan alias ini saat mendaftarkan middleware pada route di file `web.php`.
- Bagian `withRouting()` mengonfigurasi file routing untuk aplikasi Laravel, sementara `withExceptions()` digunakan untuk menambahkan penanganan error jika diperlukan.

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration		✓
	1. Migration Kasus	✓

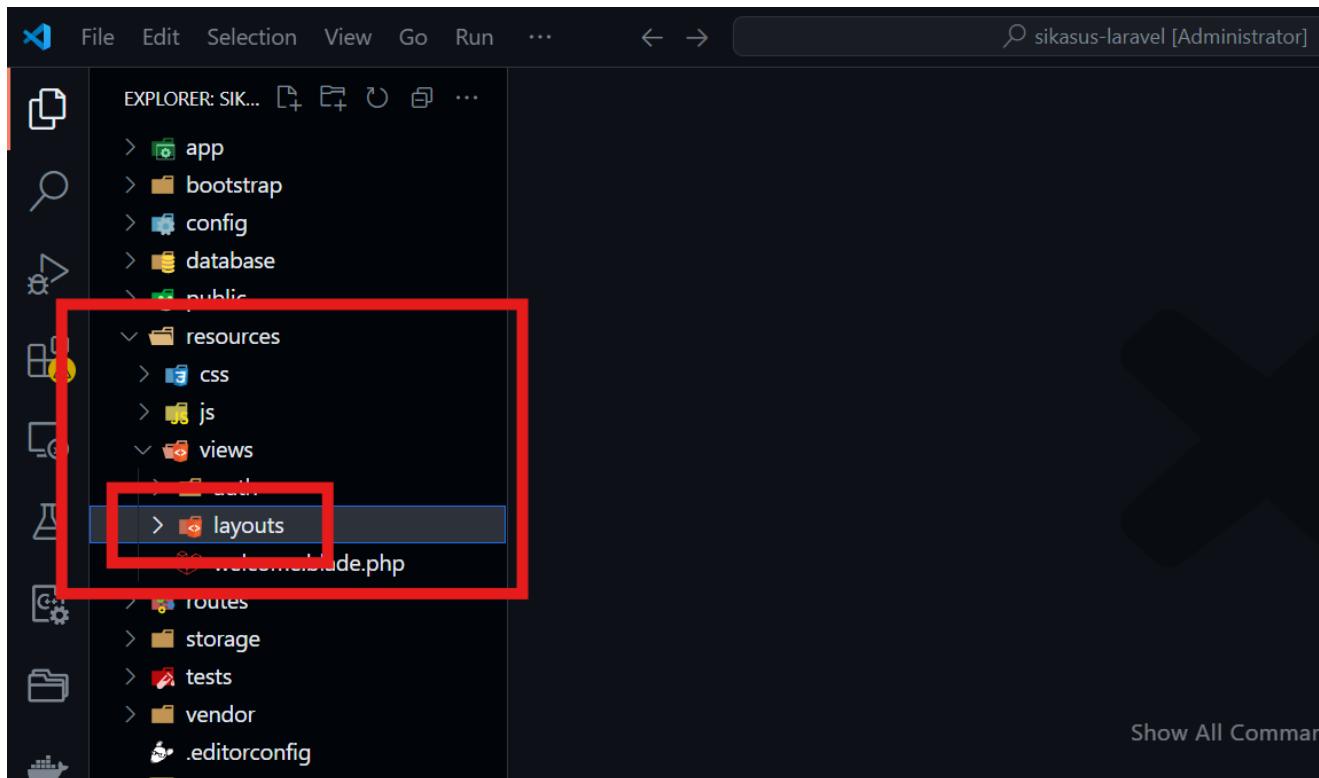
Topik	Sub Topik	Status
	2. Migration Siswa	✓
	3. Migration Kelas	✓
	4. Migration Walikelas	✓
Menambahkan Mass Assignment		✓
	1. Model Kasus	✓
	2. Model Siswa	✓
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓
	Menampilkan Halaman Jenis Login	✓
	Menampilkan Form Login Berdasarkan Jenis	✓
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		✓
	Membuat Middleware untuk Autentikasi	✓
	Membuat Middleware untuk Role	✓
	Registrasi Middleware	✓
Membuat Blade Layout		
Membuat Tampilan Homepage		
Membuat Tampilan Dashboard Setelah Login		
Membuat Fitur Kasus		
	Menambahkan Routing untuk Kasus	
	Membuat Controller Kasus	
	Menampilkan Daftar Kasus (Method index)	

Topik	Sub Topik	Status
	Membuat View untuk Menampilkan Kasus (View index)	
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kasus (View create)	
	Edit dan Update Kasus ke Database	
	Membuat View Form Edit Kasus (View edit)	
	Menambahkan Method Destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Siswa		
	Menambahkan Routing untuk Siswa	
	Membuat Controller Siswa	
	Menampilkan Daftar Siswa (Method index)	
	Membuat View untuk Menampilkan Siswa (View index)	
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Siswa (View create)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Walikelas		
	Menambahkan Routing untuk Walikelas	
	Membuat Controller Walikelas	
	Menampilkan Daftar Walikelas (Method index)	
	Membuat View untuk Menampilkan Walikelas (View index)	
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	

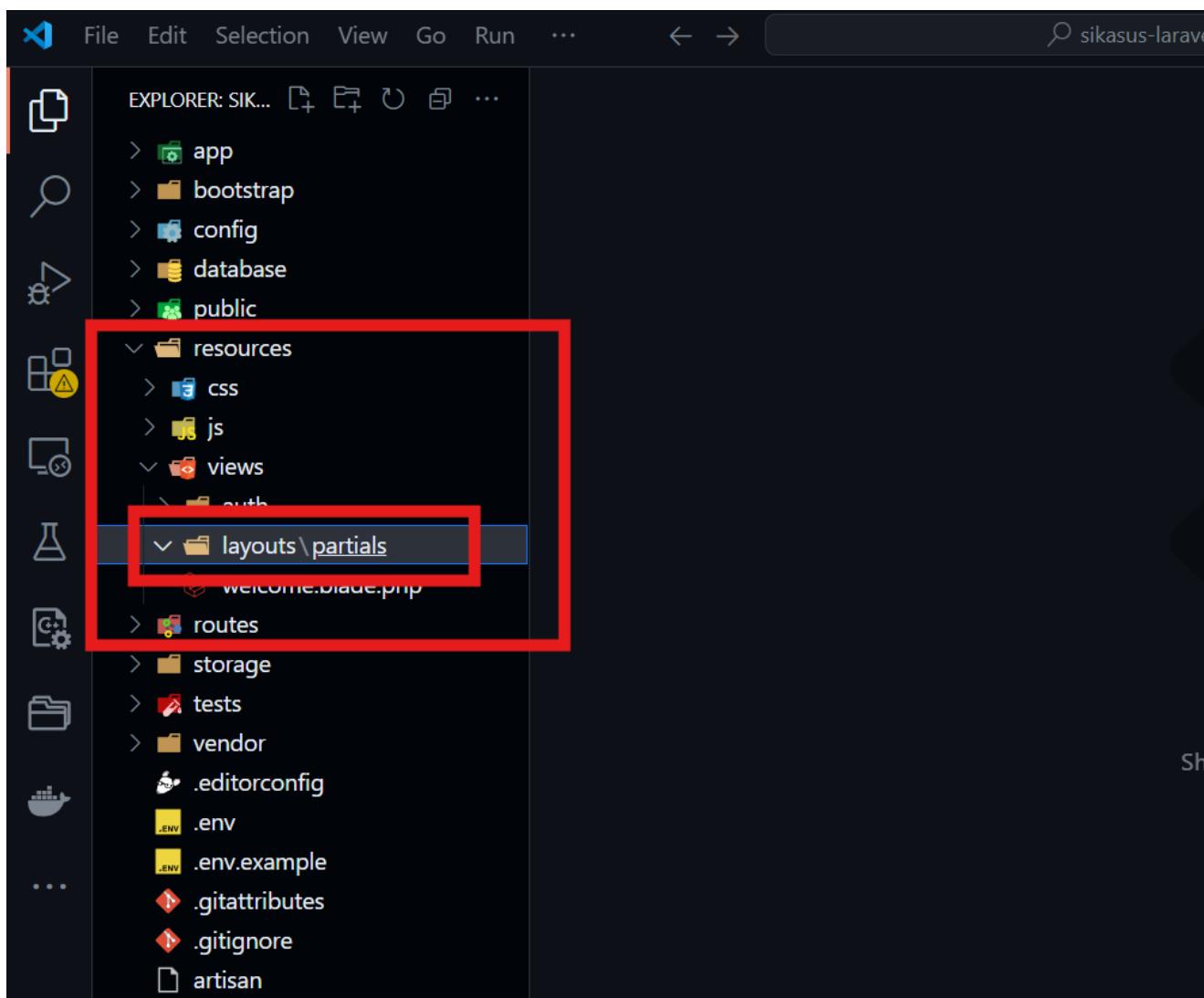
Topik	Sub Topik	Status
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk `Siswa	
	Membuat Controller `Siswa	

Membuat Blade Layout

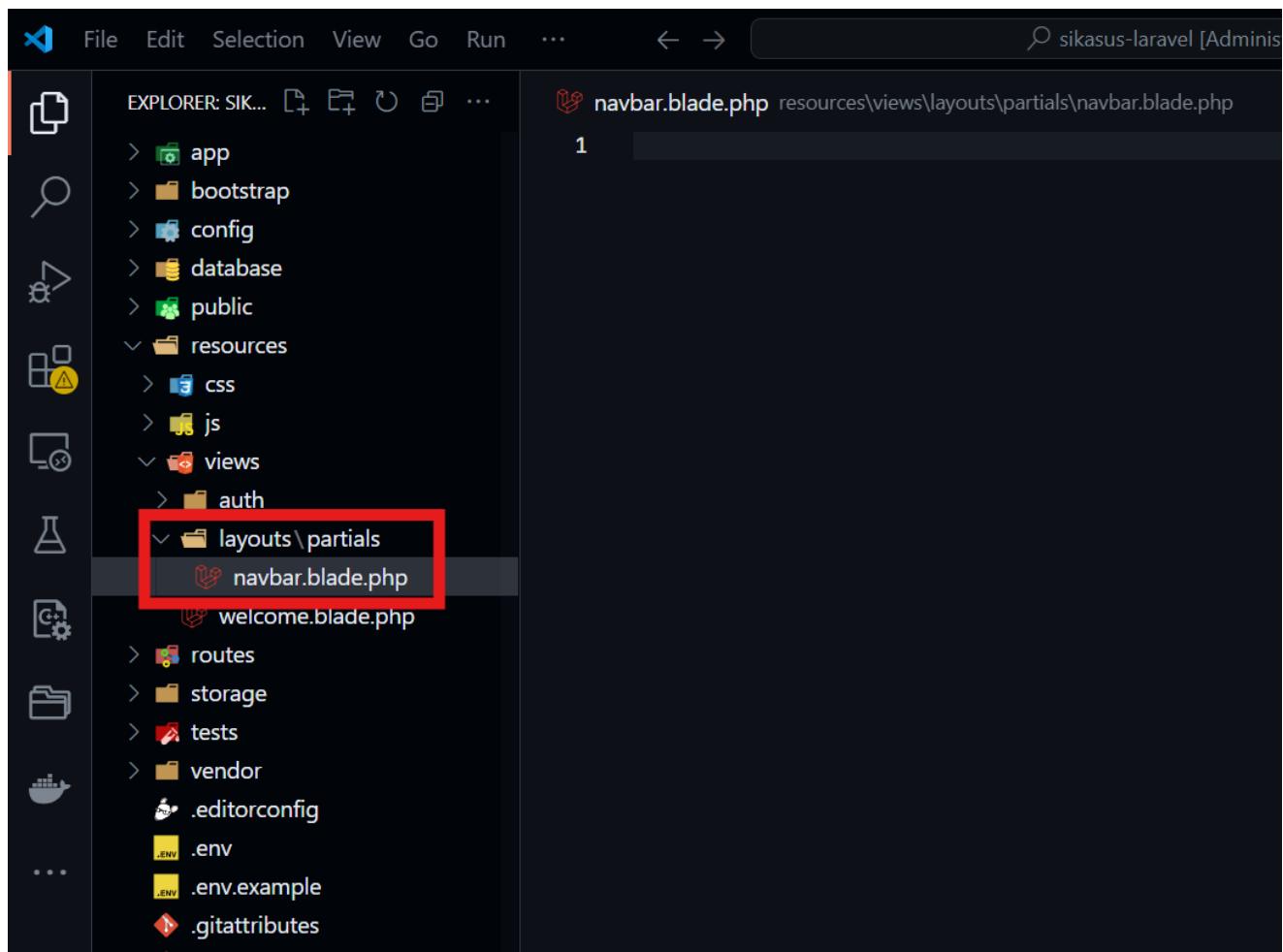
Langkah pertama dalam membuat **Blade Layouts** adalah memastikan struktur folder dan file untuk layout sesuai. navigasikan ke folder `resources/views` . Di dalam folder tersebut, buat folder baru bernama `layouts` untuk menyimpan file layout utama.



Selanjutnya, buat subfolder di dalam folder layouts bernama partials yang akan digunakan untuk menyimpan komponen seperti navbar, sidebar, dan footer.



Kemudian, lanjutkan dengan membuat komponen seperti navbar, sidebar, dan footer. Untuk navbar, buat file baru di folder `partials` dengan nama `navbar.blade.php`.



Tambahkan kode untuk navbar menggunakan struktur HTML dan Bootstrap.

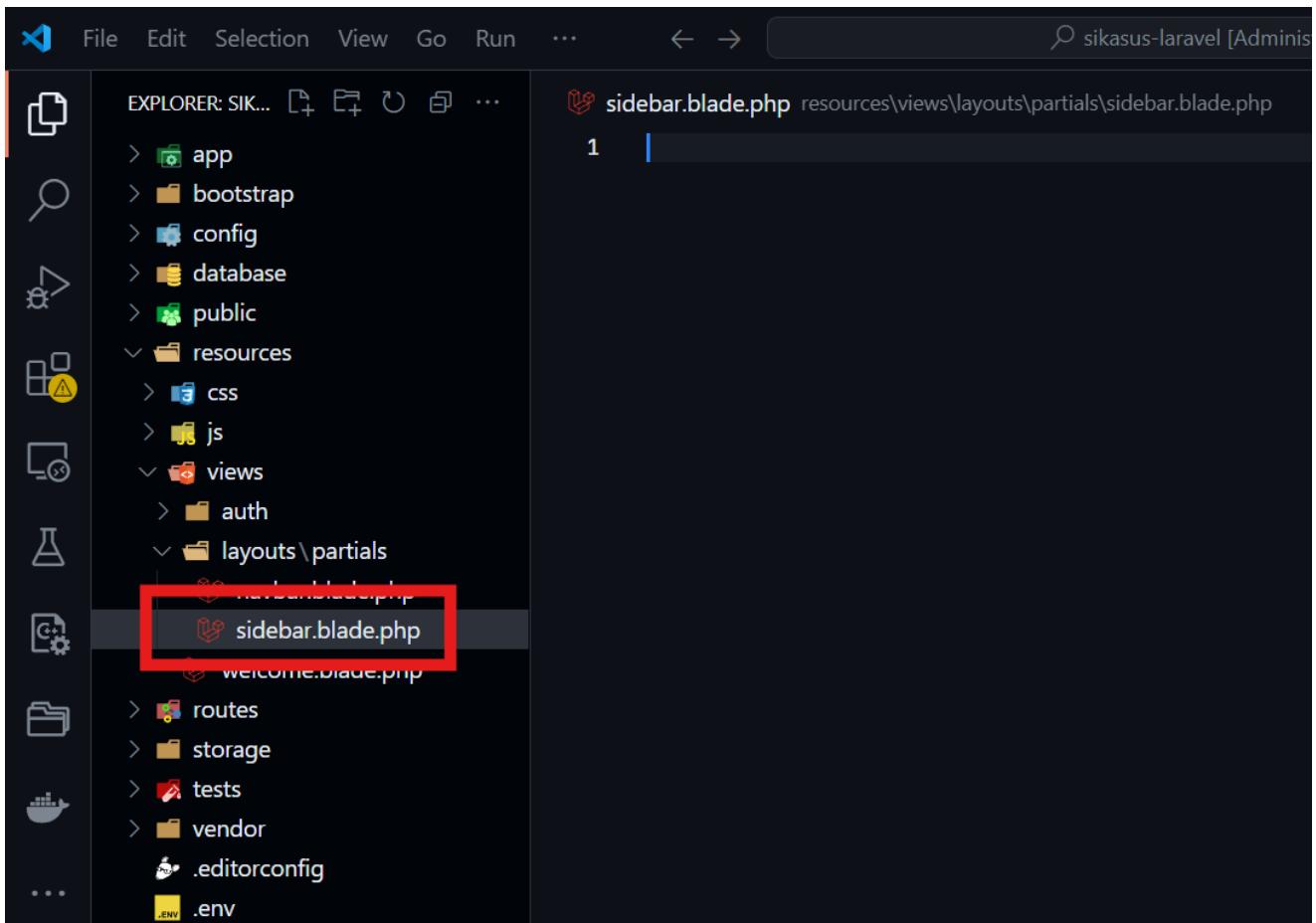
```
<nav class="sb-topnav navbar navbar-expand navbar-dark bg-dark">
    <!-- Navbar Brand-->
    <a class="navbar-brand ps-3" href="index.html">SIKASUS</a>
    <!-- Sidebar Toggle-->
    <button class="btn btn-link btn-sm order-1 order-lg-0 me-4 me-lg-0"
id="sidebarToggle" href="#"></button>
    <!-- Navbar Search-->
    <form class="d-none d-md-inline-block form-inline ms-auto me-0 me-md-3
my-2 my-md-0">
        <div class="input-group">
            <input class="form-control" type="text" placeholder="Search
for..." aria-label="Search for..." aria-describedby="btnNavbarSearch" />
            <button class="btn btn-primary" id="btnNavbarSearch"
type="button"><i class="fas fa-search"></i></button>
        </div>
    </form>
    <!-- Navbar-->
    <ul class="navbar-nav ms-auto ms-md-0 me-3 me-lg-4">
```

```

<li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle" id="navbarDropdown" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false"><i class="fas fa-user fa-fw"></i></a>
        <ul class="dropdown-menu dropdown-menu-end" aria-labelledby="navbarDropdown">
            <li><a class="dropdown-item" href="{{ route('auth.logout') }}>Logout</a></li>
        </ul>
    </li>
</ul>
</nav>

```

Langkah berikutnya adalah membuat sidebar. Masih di folder `partials`, buat file baru bernama `sidebar.blade.php`.



Tambahkan kode HTML berikut untuk sidebar navigasi.

```
<div id="layoutSidenav_nav">
    <nav class="sb-sidenav accordion sb-sidenav-dark" id="sidenavAccordion">
        <div class="sb-sidenav-menu">
            <div class="nav">
                <div class="sb-sidenav-menu-heading">Menu Utama</div>
                @php
                    $role = session('role');
                @endphp
                {{-- Role Admin --}}
                @if (isset($role) && $role === 'admin')
                    <a class="nav-link" href="{{ url('/dashboard') }}>
                        <div class="sb-nav-link-icon"><i class="fas fa-tachometer-alt"></i></div>
                        Dashboard
                    </a>

                    <div class="sb-sidenav-menu-heading">Manajemen</div>
                    <a class="nav-link collapsed" href="#" data-bs-
toggle="collapse"
                        data-bs-target="#collapseManagement" aria-
expanded="false" aria-controls="collapseManagement">
                        <div class="sb-nav-link-icon"><i class="fas fa-
columns"></i></div>
                        Data Master
                    </a>
                @endif
            </div>
        </div>
    </nav>
</div>
```

```

        <div class="sb-sidenav-collapse-arrow"><i class="fas fa-angle-down"></i></div>
        </a>
        <div class="collapse" id="collapseManagement" aria-labelledby="headingOne"
            data-bs-parent="#sidenavAccordion">
            <nav class="sb-sidenav-menu-nested nav">
                <a class="nav-link" href="{{ url('/walikelas') }}>Wali Kelas</a>
                <a class="nav-link" href="{{ url('/kelas') }}>Kelas</a>
                <a class="nav-link" href="{{ url('/siswa') }}>Siswa</a>
            </nav>
        </div>

        <a class="nav-link" href="{{ url('/kasus') }}>
            <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i></div>
            Kasus Siswa
        </a>

        {{-- Role Wali Kelas --}}
        @elseif (isset($role) && $role === 'walikelas')
            <a class="nav-link" href="{{ url('/dashboard') }}>
                <div class="sb-nav-link-icon"><i class="fas fa-tachometer-alt"></i></div>
                Dashboard
            </a>

            <div class="sb-sidenav-menu-heading">Menu
Walikelas</div>
            <a class="nav-link" href="{{ url('/siswa') }}>
                <div class="sb-nav-link-icon"><i class="fas fa-user-graduate"></i></div>
                Siswa
            </a>
            <a class="nav-link" href="{{ url('/kasus') }}>
                <div class="sb-nav-link-icon"><i class="fas fa-book-open"></i></div>
                Kasus Siswa
            </a>
        @endif

        <div class="sb-sidenav-menu-heading">Akun</div>
        <a class="nav-link" href="{{ route('auth.logout') }}>
            <div class="sb-nav-link-icon"><i class="fas fa-sign-out-alt"></i></div>
            Logout
        </a>
    
```

```

        </div>
    </div>
    <div class="sb-sidenav-footer">
        <div class="small">Logged in as:</div>
        {{ $role ?? 'Pengguna' }}
    </div>
</nav>
</div>

```

```

<div id="LayoutSidenav_nav">
    <nav class="sb-sidenav accordion sb-sidenav-dark" id="sidenavAccordion">
        <div class="sb-sidenav-menu">
            <div class="nav">
                <div class="sb-sidenav-menu-heading">Menu Utama</div>
                @php
                    $role = session('role');
                @endphp
                @if ($role === 'admin')
                    <a class="nav-link" href="{{ url('/dashboard') }}>
                        <div class="sb-nav-link-icon"><i class="fas fa-tachometer-alt"></i></div>
                        Dashboard
                    </a>
                @elseif ($role === 'walikelas')
                    <a class="nav-link" href="{{ url('/dashboard') }}>
                        <div class="sb-nav-link-icon"><i class="fas fa-tachometer-alt"></i></div>
                        Dashboard
                    </a>
                @endif
                <a class="nav-link" href="{{ route('auth.logout') }}>
                    <div class="sb-nav-link-icon"><i class="fas fa-sign-out-alt"></i></div>
                    Logout
                </a>
            </div>
        </div>
    </nav>
</div>

```

Terakhir, tambahkan footer. Buat file baru bernama `footer.blade.php` di folder `partials`

```

1

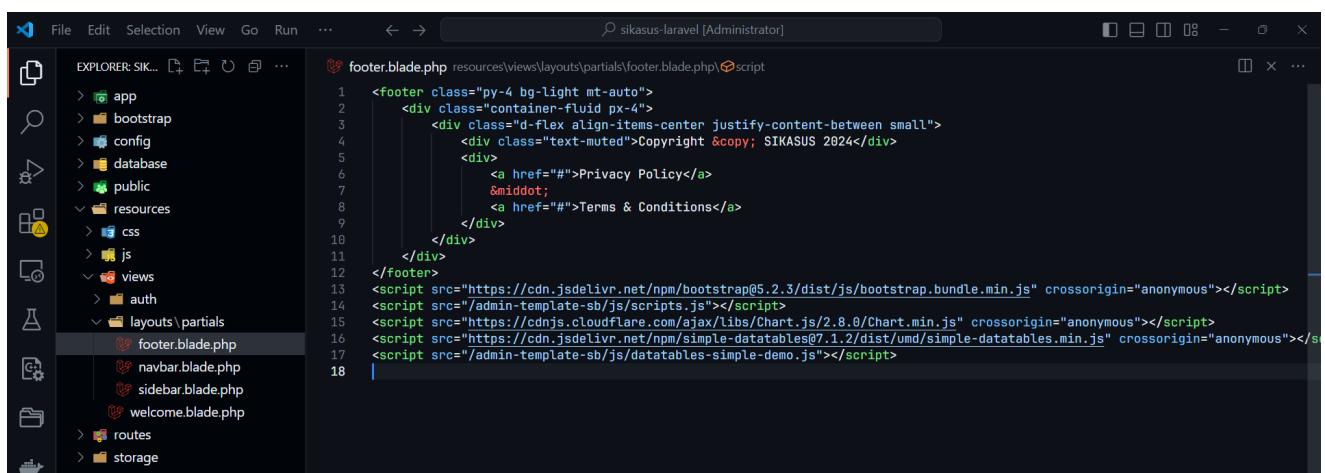
```

Kemudian, tambahkan kode berikut:

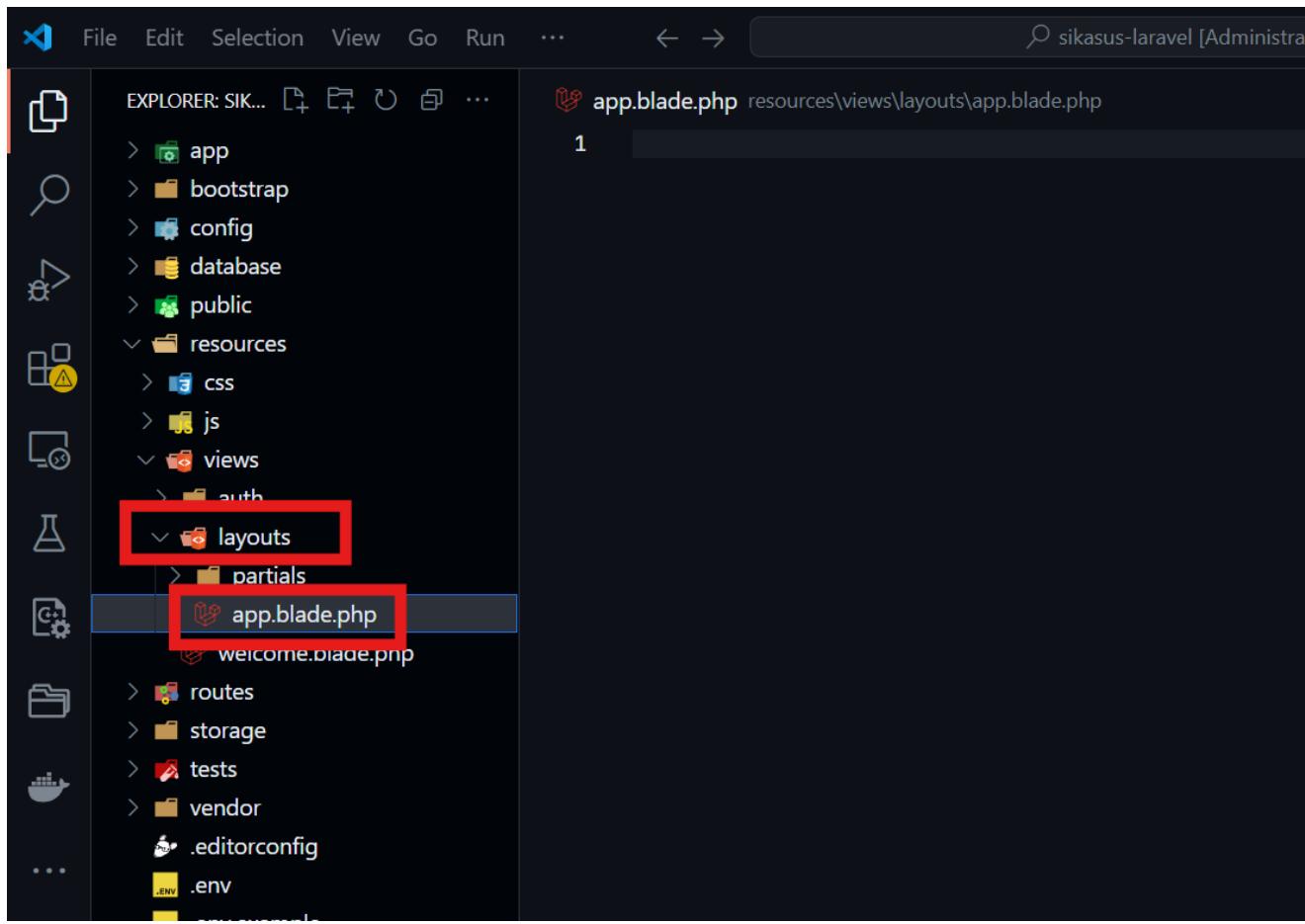
```

<footer class="py-4 bg-light mt-auto">
    <div class="container-fluid px-4">
        <div class="d-flex align-items-center justify-content-between small">
            <div class="text-muted">Copyright © SIKASUS 2024</div>
            <div>
                <a href="#">Privacy Policy</a>
                &middot;
                <a href="#">Terms & Conditions</a>
            </div>
        </div>
    </div>
</div>
</footer>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" crossorigin="anonymous"></script>
<script src="/admin-template-sb/js/scripts.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.8.0/Chart.min.js"
crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/simple-datatables@7.1.2/dist/umd/simple-datatables.min.js" crossorigin="anonymous">
</script>
<script src="/admin-template-sb/js/datatables-simple-demo.js"></script>

```



Setelah struktur folder selesai dibuat, langkah berikutnya adalah membuat file layout utama. Klik kanan pada folder `layouts`, pilih **New File**, lalu beri nama file tersebut `app.blade.php`.



Lalu, Isi file dengan struktur HTML dasar yang mencakup penggunaan `@yield` untuk bagian yang akan diisi oleh konten halaman lain.

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="utf-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no" />
    <title>
        @yield('title', ucwords(str_replace('-', ' ', Route::currentRouteName())))
    </title>
    <link href="https://cdn.jsdelivr.net/npm/simple-
datatables@7.1.2/dist/style.min.css" rel="stylesheet" />
    <link href="/admin-template-sb/css/styles.css" rel="stylesheet" />
    <script src="https://use.fontawesome.com/releases/v6.3.0/js/all.js"
crossorigin="anonymous"></script>
</head>

<body class="sb-nav-fixed">
    @include('layouts.partials.navbar')

    <div id="layoutSidenav">
```

```

@include('layouts.partials.sidebar')
<div id="layoutSidenav_content">
    <main>
        <div class="container-fluid px-4">
            @yield('content')
        </div>
    </main>
    @include('layouts.partials.footer')
</div>
</body>
</html>

```

```

<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />
        <title>
            @yield('title', ucwords(str_replace('-', ' ', Route::currentRouteName())))
        </title>
        <link href="https://cdn.jsdelivr.net/npm/simple-datatables@7.1.2/dist/style.min.css" rel="stylesheet" />
        <link href="/admin-template-sb/css/styles.css" rel="stylesheet" />
        <script src="https://use.fontawesome.com/releases/v6.3.0/js/all.js" crossorigin="anonymous"></script>
    </head>
    <body class="sb-nav-fixed">
        @include('layouts.partials.navbar')

        <div id="layoutSidenav">
            @include('layouts.partials.sidebar')
            <div id="layoutSidenav_content">
                <main>
                    <div class="container-fluid px-4">
                        @yield('content')
                    </div>
                </main>
                @include('layouts.partials.footer')
            </div>
        </div>
    </body>

```

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration	1. Migration Kasus 2. Migration Siswa 3. Migration Kelas 4. Migration Walikelas	✓
Menambahkan Mass Assignment		✓
	1. Model Kasus 2. Model Siswa	✓

Topik	Sub Topik	Status
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓
	Menampilkan Halaman Jenis Login	✓
	Menampilkan Form Login Berdasarkan Jenis	✓
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		✓
	Membuat Middleware untuk Autentikasi	✓
	Membuat Middleware untuk Role	✓
	Registrasi Middleware	✓
Membuat Blade Layout		✓
Membuat Tampilan Homepage		
Membuat Tampilan Dashboard Setelah Login		
Membuat Fitur Kasus		
	Menambahkan Routing untuk Kasus	
	Membuat Controller Kasus	
	Menampilkan Daftar Kasus (Method index)	
	Membuat View untuk Menampilkan Kasus (View index)	
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kasus (View create)	
	Edit dan Update Kasus ke Database	
	Membuat View Form Edit Kasus (View edit)	

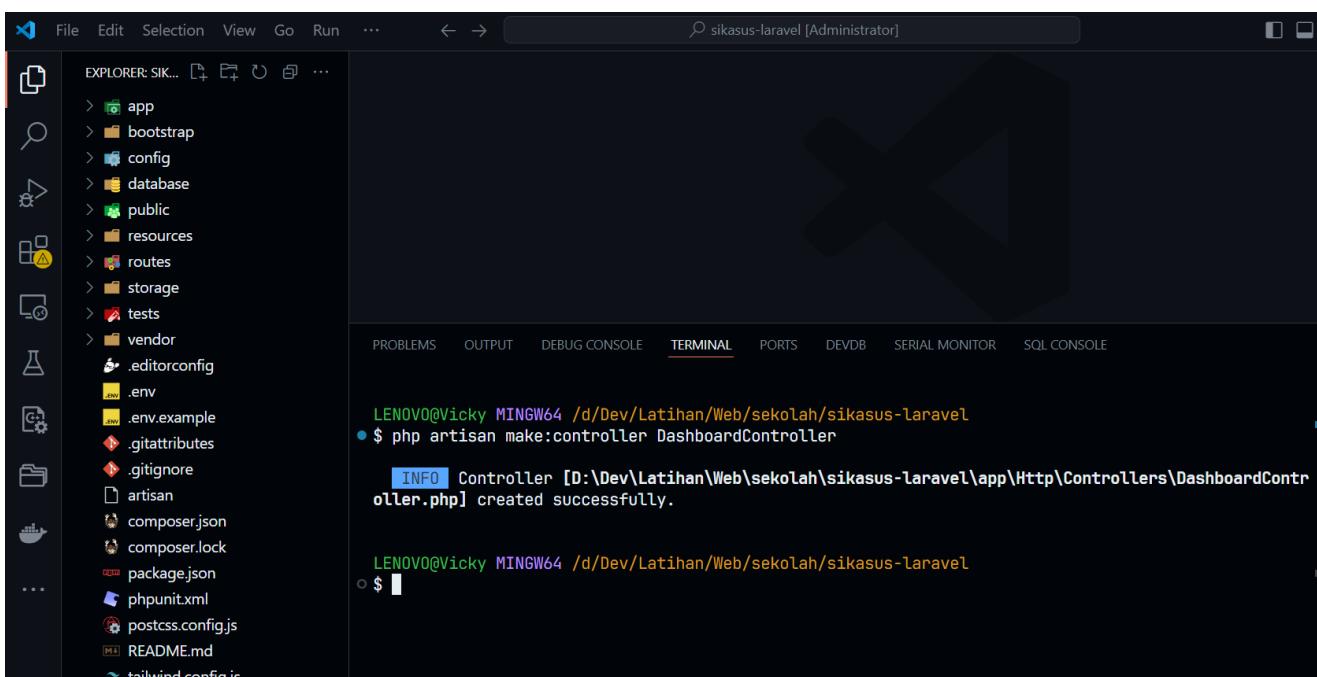
Topik	Sub Topik	Status
	Menambahkan Method Destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Siswa		
	Menambahkan Routing untuk Siswa	
	Membuat Controller Siswa	
	Menampilkan Daftar Siswa (Method index)	
	Membuat View untuk Menampilkan Siswa (View index)	
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Siswa (View create)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Walikelas		
	Menambahkan Routing untuk Walikelas	
	Membuat Controller Walikelas	
	Menampilkan Daftar Walikelas (Method index)	
	Membuat View untuk Menampilkan Walikelas (View index)	
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	

Topik	Sub Topik	Status
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk 'Siswa'	
	Membuat Controller 'Siswa'	

Membuat Tampilan Homepage

Langkah pertama dalam pembuatan tampilan homepage adalah dengan membuat controller yang menangani tampilan tersebut. Untuk itu, jalankan perintah Artisan di terminal untuk membuat controller baru bernama `DashboardController` dengan perintah:

```
php artisan make:controller DashboardController
```



The screenshot shows a Microsoft Visual Studio Code interface. On the left is the Explorer sidebar displaying project files like app, bootstrap, config, database, public, routes, storage, tests, vendor, .env, .env.example, .gitattributes, .gitignore, artisan, composer.json, composer.lock, package.json, phpunit.xml, postcss.config.js, README.md, and tailwind.config.js. The main area is the terminal window with the following content:

```
LENovo@Vicky MINGW64 /d/Dev/Latihan/Web/sekolah/sikasus-laravel
$ php artisan make:controller DashboardController
INFO [Controller [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\app\Http\Controllers\DashboardController.php] created successfully.

LENovo@Vicky MINGW64 /d/Dev/Latihan/Web/sekolah/sikasus-laravel
$
```

Setelah controller berhasil dibuat, buka file `DashboardController.php` yang terletak di folder `app/Http/Controllers`.

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
0 references | 0 implementations  
class DashboardController extends Controller  
{  
    //
```

Di dalam controller ini, buat fungsi `index` yang akan mengembalikan tampilan homepage dengan kode berikut:

```
public function index()  
{  
    return view('home');  
}
```

kita menggunakan `view('home')`, nama `home` merujuk pada file tampilan yang terletak di direktori `resources/views`. Laravel akan mencari file `home.blade.php` di dalam folder tersebut. Jadi, saat fungsi `index()` dipanggil, Laravel akan menampilkan file `home.blade.php` sebagai halaman utama aplikasi. Nama `home` di sini hanya merupakan nama alias untuk file tampilan, tanpa perlu menyebutkan ekstensi `.blade.php`.

The screenshot shows a code editor interface with the following details:

- Explorer Panel:** Shows the project structure under the "app" directory:
 - Http
 - Controllers
 - AuthController.php
 - Controller.php
 - DashboardController.php
- Main Panel:** Displays the code for `DashboardController.php`:

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class DashboardController extends Controller
{
    public function index(): View
    {
        return view('home');
    }
}
```

Selanjutnya, tambahkan fungsi `search` yang akan menangani pencarian data berdasarkan NISN dan Tanggal Lahir. Fungsi ini akan menerima input dari form yang ada di homepage, memvalidasi input tersebut, dan kemudian mencari data siswa yang sesuai dalam database. Berikut adalah kode untuk fungsi `search`:

```
public function search(Request $request)
{
    $request->validate([
        'nisn' => 'required|string',
        'tanggal_lahir' => 'required|date',
    ]);

    $nisn = $request->input('nisn');
    $tanggal_lahir = $request->input('tanggal_lahir');

    $siswa = \App\Models\Siswa::with('kasus')
        ->where('nisn', $nisn)
        ->where('tanggal_lahir', $tanggal_lahir)
        ->first();

    if (!$siswa) {
        return back()->withErrors(['message' => 'Data siswa tidak ditemukan.'])->withInput();
    }

    return view('home', [
        'siswa' => $siswa,
        'success_message' => 'Data ditemukan!',
    ]);
}
```

```
];
}

EXPLORER: SIK...
File Edit Selection View Go Run ...
DashboardController.php app\Http\Controllers\DashboardController.php DashboardController\search()
DashboardController.php
class DashboardController extends Controller
{
    public function index(): View
    {
        return view('home');
    }

    public function search(Request $request): RedirectResponse|View
    {
        $request->validate([
            'nisn' => 'required|string',
            'tanggal_lahir' => 'required|date',
        ]);

        $nisn = $request->input('nisn');
        $tanggal_lahir = $request->input('tanggal_lahir');

        $siswa = \App\Models\Siswa::with('kelas')
            ->where('nisn', $nisn)
            ->where('tanggal_lahir', $tanggal_lahir)
            ->first();

        if (!$siswa) {
            return back()->withErrors(['message' => 'Data siswa tidak ditemukan.']);
        }

        return view('home', [
            'siswa' => $siswa,
            'success_message' => 'Data ditemukan!',
        ]);
    }
}
```

Setelah membuat fungsi-fungsi tersebut, langkah berikutnya adalah menambahkan routing untuk menampilkan halaman homepage dan proses pencarian data. Buka file `routes/web.php` dan tambahkan routing untuk menghubungkan URL dengan fungsi yang telah dibuat di controller, seperti ini:

```
Route::controller(App\Http\Controllers\DashboardController::class)
    ->group(function () {
        Route::get('/', 'index')->name('dashboard');
        Route::post('/', 'search')->name('dashboard.search');
    });
}
```

```
File Edit Selection View Go Run ...
web.php routes\web.php
<?php
use App\Http\Controllers\AuthController;
use Illuminate\Support\Facades\Route;

Route::prefix('auth')
    ->name('auth.')
    ->controller(AuthController::class)
    ->group(function (): void {
        Route::get('Login', action: 'showLoginTypes')->name(name: 'login.types');
        Route::get('Login/{type}', action: 'showLoginForm')->name(name: 'login.form');
        Route::post('Login', action: 'login')->name(name: 'login.submit');
        Route::get('Logout', action: 'logout')->name(name: 'logout');
    });

Route::controller(DashboardController::class)
    ->group(function (): void {
        Route::get('/', action: 'index')->name(name: 'dashboard');
        Route::post('/', action: 'search')->name(name: 'dashboard.search');
    });
}

EXPLORER: SIK...
File Edit Selection View Go Run ...
routes\web.php
File Edit Selection View Go Run ...
DashboardController.php app\Http\Controllers\DashboardController.php DashboardController\search()
DashboardController.php
class DashboardController extends Controller
{
    public function index(): View
    {
        return view('home');
    }

    public function search(Request $request): RedirectResponse|View
    {
        $request->validate([
            'nisn' => 'required|string',
            'tanggal_lahir' => 'required|date',
        ]);

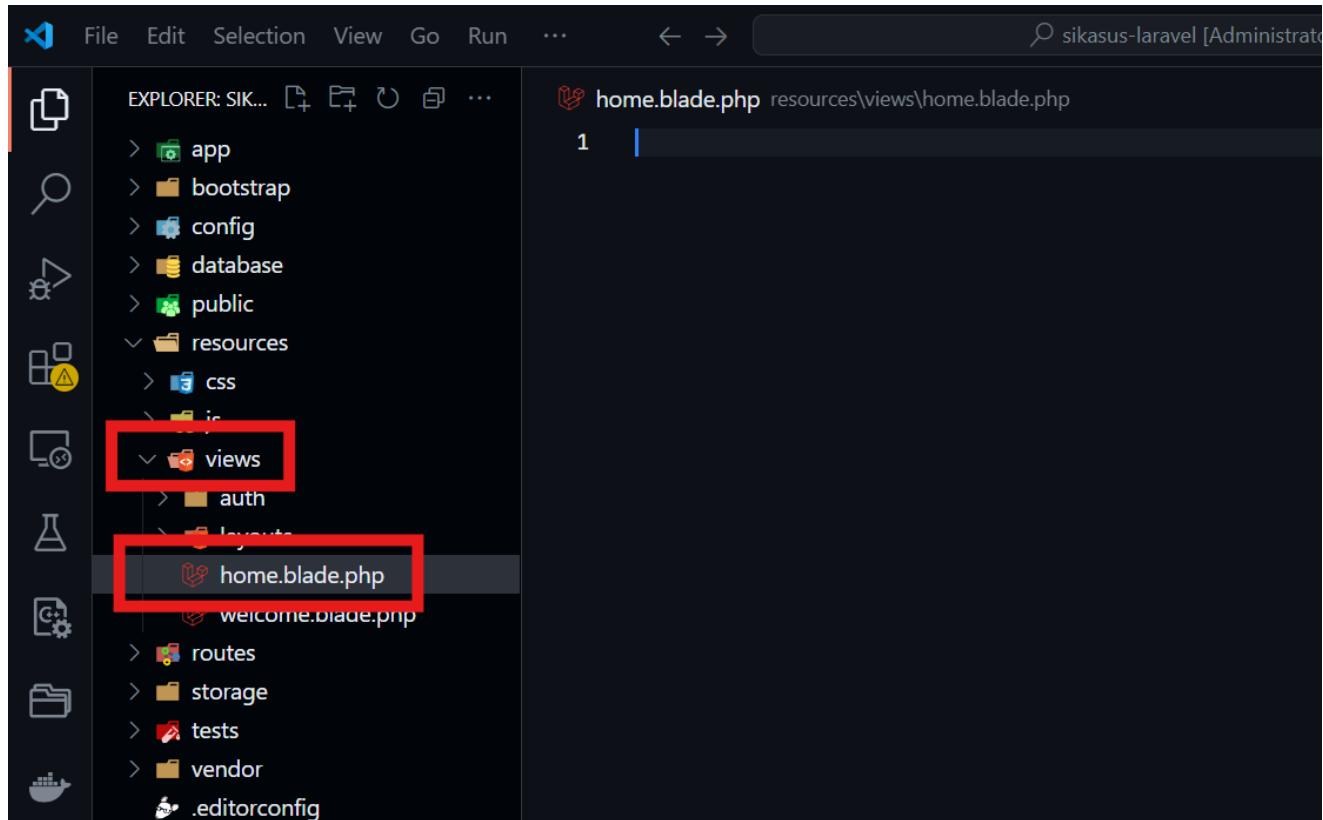
        $nisn = $request->input('nisn');
        $tanggal_lahir = $request->input('tanggal_lahir');

        $siswa = \App\Models\Siswa::with('kelas')
            ->where('nisn', $nisn)
            ->where('tanggal_lahir', $tanggal_lahir)
            ->first();

        if (!$siswa) {
            return back()->withErrors(['message' => 'Data siswa tidak ditemukan.']);
        }

        return view('home', [
            'siswa' => $siswa,
            'success_message' => 'Data ditemukan!',
        ]);
    }
}
```

Kemudian, buat tampilan homepage menggunakan Blade template di dalam folder `resources/views`. Buat file `home.blade.php` yang akan menampilkan form pencarian dan informasi terkait siswa beserta kasus yang dimilikinya jika ditemukan.



Kemudian, tambahkan kode berikut untuk file `home.blade.php`:

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sistem Informasi Kasus Siswa</title>
    <link
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
        rel="stylesheet">
</head>
<body class="bg-light">
    <div class="container py-5">
        <div class="text-center mb-4">
            <h1 class="display-6 text-primary">Sistem Informasi Kasus
            Siswa</h1>
            <p class="text-secondary">Masukkan NISN dan Tanggal Lahir untuk
            memeriksa data kasus</p>
        </div>

        <form action="{{ route('dashboard.search') }}" method="POST"
        class="p-4 border rounded w-50 mx-auto shadow-sm">
            @csrf
```

```

<div class="mb-3">
    <label for="nisn" class="form-label">NISN</label>
    <input type="text" id="nisn" name="nisn" class="form-control" placeholder="Masukkan NISN"
           value="{{ old('nisn') }}" required>
</div>
<div class="mb-3">
    <label for="tanggal_lahir" class="form-label">Tanggal Lahir</label>
    <input type="date" id="tanggal_lahir" name="tanggal_lahir" class="form-control"
           value="{{ old('tanggal_lahir') }}" required>
</div>
<button type="submit" class="btn btn-primary w-100">Cari Data</button>
</form>

@if ($errors->any())
    <div class="alert alert-danger mt-4">
        {{ $errors->first('message') }}
    </div>
@endif

@if (session('success_message'))
    <div class="alert alert-success mt-4">
        {{ session('success_message') }}
    </div>
@endif

@if (isset($siswa))
    <div class="card mt-4">
        <div class="card-header bg-primary text-white">
            Informasi Kasus Siswa
        </div>
        <div class="card-body">
            <p><strong>NISN:</strong> {{ $siswa->nisn }}</p>
            <p><strong>Nama Lengkap:</strong> {{ $siswa->nama_lengkap }}</p>
            <p><strong>Tanggal Lahir:</strong> {{ \Carbon\Carbon::parse($siswa->tanggal_lahir)->format('d F Y') }}</p>
        @if ($siswa->kasus->isNotEmpty())
            <hr>
            <h5>Daftar Kasus:</h5>
            <ul class="list-group">
                @foreach ($siswa->kasus as $kasus)
                    <li class="list-group-item">
                        <strong>Kasus:</strong> {{ $kasus->deskripsi_kasus }}<br>

```

```

        <strong>Tanggal Kejadian:</strong>
        {{ \Carbon\Carbon::parse($kasus-
>tanggal_kasus)->format('d F Y') }}
```


@endforeach

@else

<p class="text-muted">Tidak ada kasus tercatat</p>

@endif

</div>

</div>

@endif

<div class="text-center mt-4">

@if (session()->has('user'))

Logout

@else

Login

@endif

</div>

</div>

<script

src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.m
in.js"></script>

</body>

</html>

```

<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sistem Informasi Kasus Siswa</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body class="bg-light">
    <div class="container py-5">
        <div class="text-center mb-4">
            <h1 class="display-6 text-primary">Sistem Informasi Kasus Siswa</h1>
            <p class="text-secondary">Masukkan NISN dan Tanggal Lahir untuk memeriksa data kasus</p>
        </div>

        <form action="{{ route('dashboard.search') }}" method="POST" class="p-4 border rounded w-50 mx-auto shadow-sm">
            @csrf
            <div class="mb-3">
                <label for="nisn" class="form-label">NISN</label>
                <input type="text" id="nisn" name="nisn" class="form-control" placeholder="Masukkan NISN"
                    value="{{ old('nisn') }}" required>
            </div>
            <div class="mb-3">
                <label for="tanggal_lahir" class="form-label">Tanggal Lahir</label>
                <input type="date" id="tanggal_lahir" name="tanggal_lahir" class="form-control"
                    value="{{ old('tanggal_lahir') }}" required>
            </div>
            <button type="submit" class="btn btn-primary w-100">Cari Data</button>
        </form>

        @if ($errors->any())
            <div class="alert alert-danger mt-4">
                {{ $errors->first('message') }}
            </div>
        @endif
    </div>
</body>
</html>
```

Setelah langkah-langkah di atas selesai, buka browser dan akses halaman

<http://localhost:8000>. Di halaman tersebut, coba masukkan NISN dan Tanggal Lahir untuk mencari data siswa dan melihat daftar kasus yang terkait.

Sistem Informasi Kasus Siswa

Masukkan NISN dan Tanggal Lahir untuk memeriksa data kasus

NISN	<input type="text"/>
Masukkan NISN	
Tanggal Lahir	<input type="text"/>
mm/dd/yyyy	
<input type="button" value="Cari Data"/>	

Informasi Kasus Siswa

NISN: 0077865020

Nama Lengkap: Andi Muh. Raihan Alkawsar

Tanggal Lahir: 01 August 2007

Daftar Kasus:

Kasus: Ab eos incident quia suscipit libero aut.

Tanggal Kejadian: 11 January 2024

Kasus: Et et quasi optio tenetur ea.

Tanggal Kejadian: 20 January 2024

Kasus: Nulla aut optio sunt enim sint error.

Tanggal Kejadian: 23 May 2024

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration		✓
	1. Migration Kasus	✓
	2. Migration Siswa	✓
	3. Migration Kelas	✓
	4. Migration Walikelas	✓
Menambahkan Mass Assignment		✓
	1. Model Kasus	✓
	2. Model Siswa	✓
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓
	Menampilkan Halaman Jenis Login	✓

Topik	Sub Topik	Status
	Menampilkan Form Login Berdasarkan Jenis	✓
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		✓
	Membuat Middleware untuk Autentikasi	✓
	Membuat Middleware untuk Role	✓
	Registrasi Middleware	✓
Membuat Blade Layout		✓
Membuat Tampilan Homepage		✓
Membuat Tampilan Dashboard Setelah Login		
Membuat Fitur Kasus		
	Menambahkan Routing untuk Kasus	
	Membuat Controller Kasus	
	Menampilkan Daftar Kasus (Method index)	
	Membuat View untuk Menampilkan Kasus (View index)	
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kasus (View create)	
	Edit dan Update Kasus ke Database	
	Membuat View Form Edit Kasus (View edit)	
	Menambahkan Method Destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Siswa		
	Menambahkan Routing untuk Siswa	
	Membuat Controller Siswa	
	Menampilkan Daftar Siswa (Method index)	
	Membuat View untuk Menampilkan Siswa (View index)	

Topik	Sub Topik	Status
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Siswa (View create)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Walikelas		
	Menambahkan Routing untuk Walikelas	
	Membuat Controller Walikelas	
	Menampilkan Daftar Walikelas (Method index)	
	Membuat View untuk Menampilkan Walikelas (View index)	
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	

Topik	Sub Topik	Status
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk 'Siswa'	
	Membuat Controller 'Siswa'	

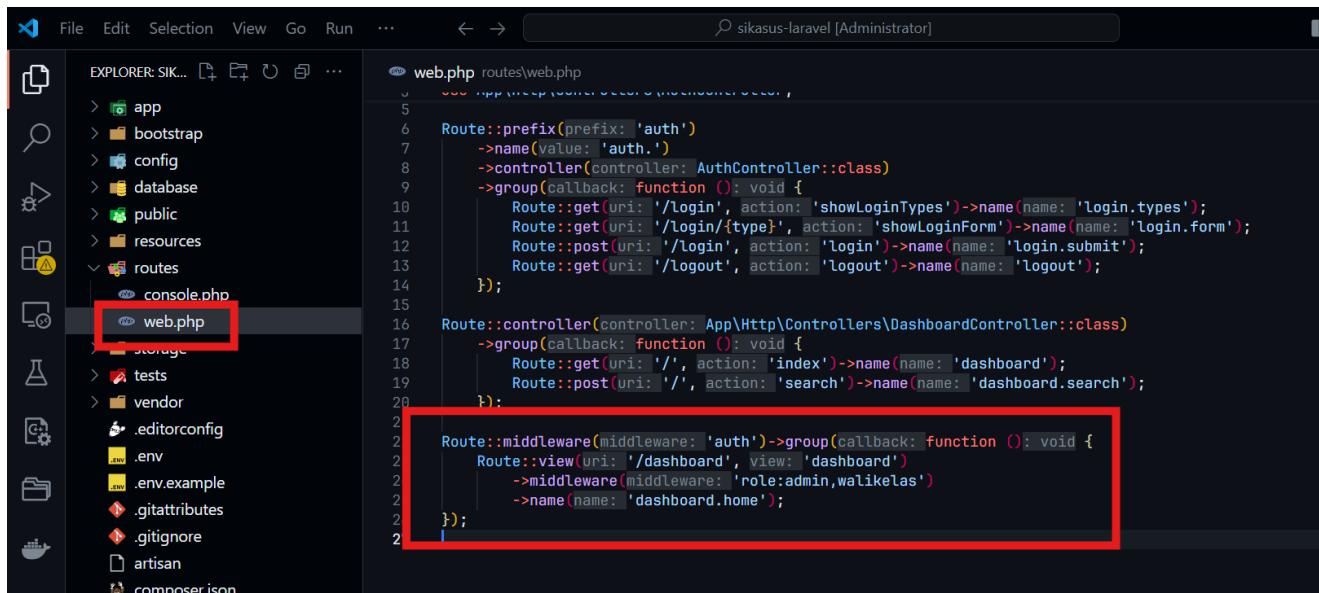
Membuat Tampilan Dashboard Setelah Login

Untuk memulai pembuatan tampilan dashboard setelah login, langkah pertama adalah menambahkan routing untuk memastikan bahwa hanya pengguna yang sudah login yang dapat mengakses halaman dashboard. Gunakan middleware auth untuk memeriksa apakah pengguna sudah terautentikasi, dan role untuk memeriksa peran pengguna. Di file `routes/web.php`, tambahkan kode berikut:

```
Route::middleware('auth')->group(function () {
    Route::view('/dashboard', 'dashboard')
        ->middleware('role:admin,walikelas')
        ->name('dashboard.home');
});
```

Penjelasan:

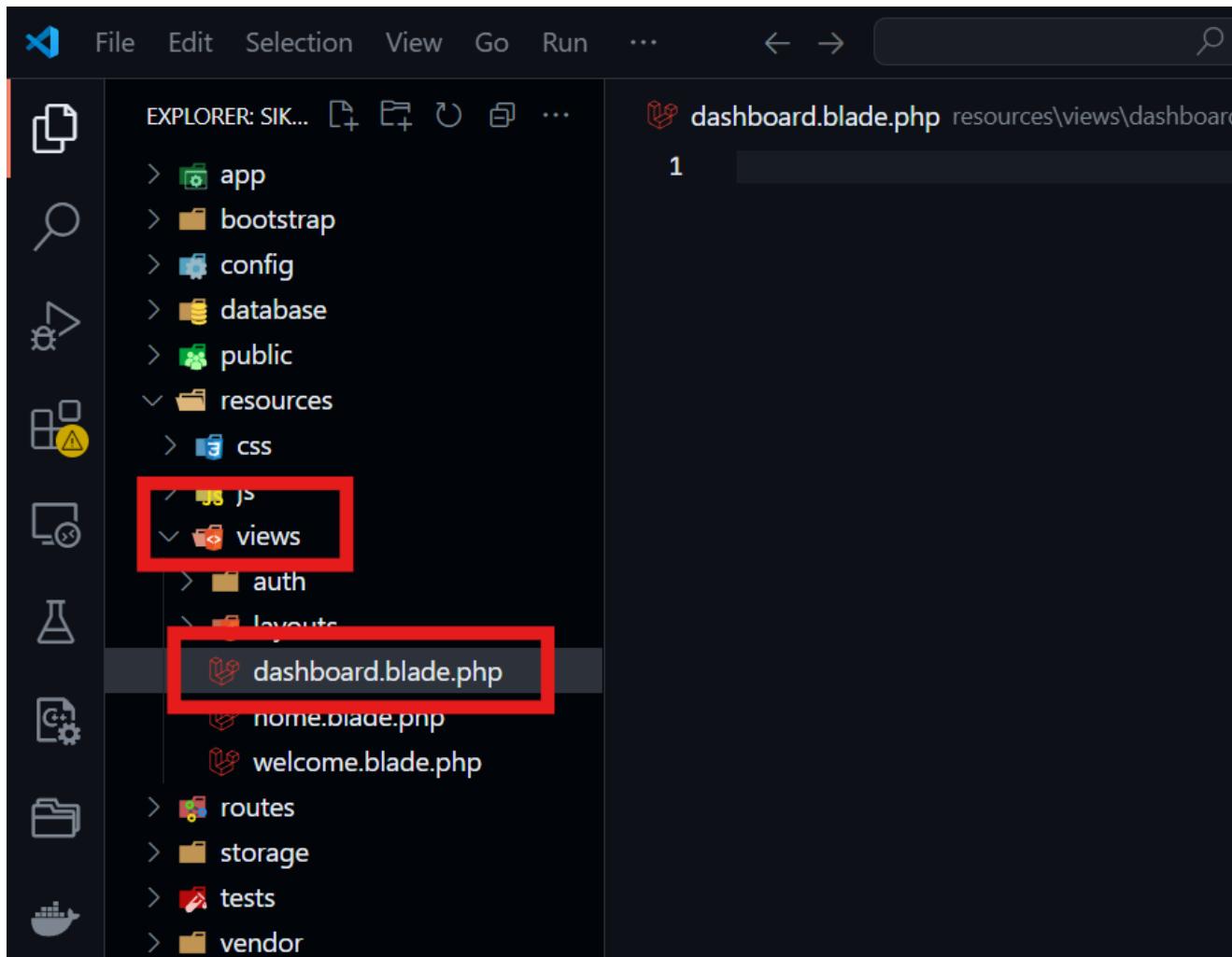
rute `/dashboard` hanya dapat diakses oleh pengguna yang sudah terautentikasi, yang dijamin dengan middleware `auth`. Kemudian, rute ini dibatasi lebih lanjut dengan middleware `role:admin,walikelas`, yang memastikan bahwa hanya pengguna dengan peran `admin` atau `walikelas` yang dapat mengaksesnya. Fungsi `Route::view()` digunakan untuk menampilkan tampilan `dashboard` tanpa memerlukan controller. Terakhir, rute ini diberi nama `dashboard.home` untuk memudahkan pengarahan dan pengelolaan rute di aplikasi.



```
web.php routes\web.php
5
6 Route::prefix('prefix: 'auth')
7     ->name('value: 'auth.')
8     ->controller(controller: AuthController::class)
9     ->group(callback: function () void {
10         Route::get(uri: '/Login', action: 'showLoginTypes')->name(name: 'login.types');
11         Route::get(uri: '/Login/{type}', action: 'showLoginForm')->name(name: 'login.form');
12         Route::post(uri: '/login', action: 'Login')->name(name: 'login.submit');
13         Route::get(uri: '/logout', action: 'Logout')->name(name: 'Logout');
14     });
15
16 Route::controller(controller: App\Http\Controllers\DashboardController::class)
17     ->group(callback: function () void {
18         Route::get(uri: '/', action: 'index')->name(name: 'dashboard');
19         Route::post(uri: '/', action: 'search')->name(name: 'dashboard.search');
20     });
21
22 Route::middleware(middleware: 'auth')->group(callback: function () void {
23     Route::view(uri: '/dashboard', view: 'dashboard')
24         ->middleware(middleware: 'role:admin, walikelas')
25     ->name(name: 'dashboard.home');
26 });
27
```

Disini hanya pengguna yang memiliki peran `admin` atau `walikelas` yang dapat mengakses halaman dashboard setelah login.

Setelah itu, buat file tampilan dashboard dengan menggunakan Blade template di dalam `resources/views`. Buat file baru bernama `dashboard.blade.php`



```
EXPLORER: SIK...
> app
> bootstrap
> config
> database
> public
< resources
> css
< views
> auth
> layouts
> dashboard.blade.php resources\views\dashboard
> nome.blade.php
> welcome.blade.php
> routes
> storage
> tests
> vendor
```

Kemudian, isi dengan kode berikut:

```

@extends('layouts.app')

@section('content')
    <h1 class="mt-4">Dashboard</h1>
    <div class="row">
        <h1>Selamat Datang, {{ ucfirst(session('role')) }}!</h1>
        <br>
        <div>Di dashboard ini, Anda dapat mengelola data kasus yang terkait melalui menu
            yang tersedia di sidebar.</div>
        <p>Pastikan Anda menggunakan fitur-fitur dengan bijak. Jika membutuhkan bantuan, silakan hubungi
            administrator melalui menu <em>Bantuan</em>.</p>
    </div>
@endsection

```

halaman dashboard akan menampilkan ucapan selamat datang berdasarkan peran pengguna, seperti admin atau walikelas .

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration		✓
	1. Migration Kasus	✓
	2. Migration Siswa	✓
	3. Migration Kelas	✓
	4. Migration Walikelas	✓
Menambahkan Mass Assignment		✓
	1. Model Kasus	✓
	2. Model Siswa	✓
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓
	Menampilkan Halaman Jenis Login	✓
	Menampilkan Form Login Berdasarkan Jenis	✓
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		✓
	Membuat Middleware untuk Autentikasi	✓
	Membuat Middleware untuk Role	✓
	Registrasi Middleware	✓
Membuat Blade Layout		✓

Topik	Sub Topik	Status
Membuat Tampilan Homepage		✓
Membuat Tampilan Dashboard Setelah Login		✓
Membuat Fitur Kasus	Menambahkan Routing untuk Kasus Membuat Controller Kasus Menampilkan Daftar Kasus (Method index) Membuat View untuk Menampilkan Kasus (View index) Menambahkan Kasus Baru ke Dalam Database (Method create dan store) Membuat Form untuk Menambah Kasus (View create) Edit dan Update Kasus ke Database Membuat View Form Edit Kasus (View edit) Menambahkan Method Destroy di Controller Menambahkan Tombol Hapus di View	
Membuat Fitur Siswa	Menambahkan Routing untuk Siswa Membuat Controller Siswa Menampilkan Daftar Siswa (Method index) Membuat View untuk Menampilkan Siswa (View index) Menambahkan Siswa Baru ke Dalam Database (Method create dan store) Membuat Form untuk Menambah Siswa (View create) Menambahkan Method destroy di Controller Menambahkan Tombol Hapus di View	
Membuat Fitur Walikelas	Menambahkan Routing untuk Walikelas Membuat Controller Walikelas Menampilkan Daftar Walikelas (Method index) Membuat View untuk Menampilkan Walikelas (View index)	

Topik	Sub Topik	Status
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk `Siswa	
	Membuat Controller `Siswa	

Membuat Fitur Kasus

Menambahkan Routing untuk Kasus

Langkah pertama adalah menambahkan routing untuk mengelola data kasus. Hal ini dilakukan dengan menggunakan resource route di `routes/web.php`, yang memungkinkan kita untuk membuat semua operasi CRUD secara otomatis.

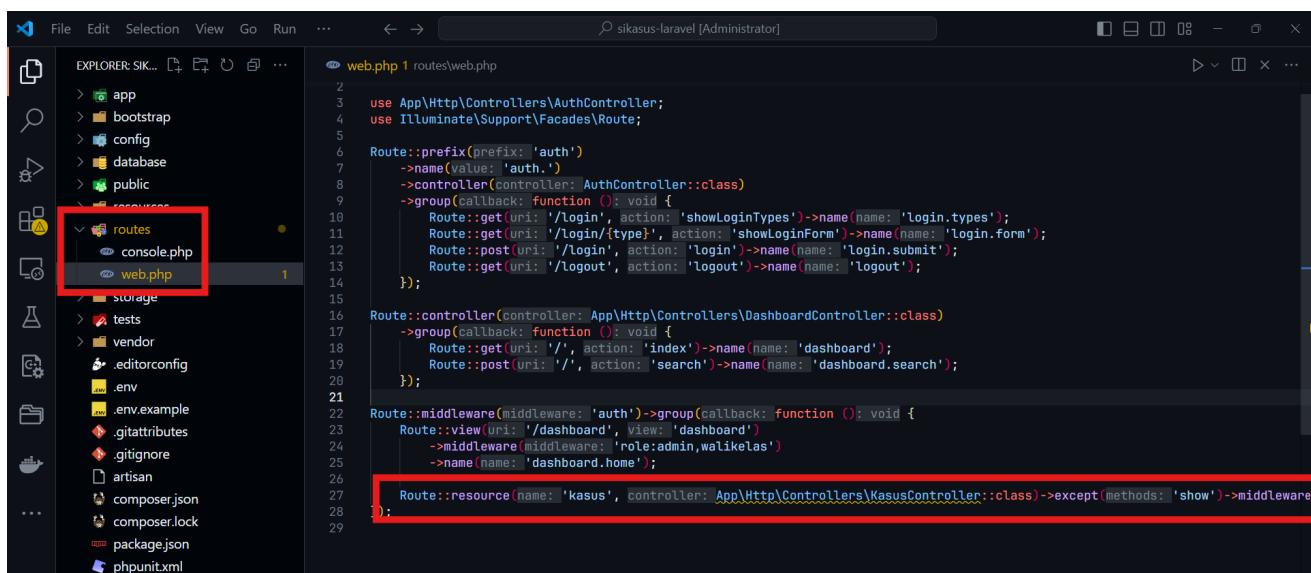
Buka file `routes/web.php` dan tambahkan routing berikut:

```
Route::middleware('auth')->group(function () {
    Route::view('/dashboard', 'dashboard')
        ->middleware('role:admin,walikelas')
        ->name('dashboard.home');

    Route::resource('kasus', \App\Http\Controllers\KasusController::class)
        ->except('show')->middleware('role:admin,walikelas');
});
```

Penjelasan:

Pada bagian `Route::resource('kasus')`, Laravel mendefinisikan rute untuk resource controller `KasusController` yang mencakup semua metode CRUD, kecuali metode `show`, dengan menggunakan `except('show')`. Middleware `role:admin,walikelas` diterapkan untuk membatasi akses hanya kepada pengguna yang memiliki peran `admin` atau `walikelas`. Dengan demikian, hanya pengguna dengan peran tersebut yang dapat mengakses rute terkait `kasus`, memastikan pengamanan yang sesuai untuk data dan operasi yang sensitif.



```
Route::prefix('auth')
    ->name(value: 'auth.')
    ->controller(controller: AuthController::class)
    ->group(callback: function (): void {
        Route::get(uri: '/Login', action: 'showLoginTypes')->name(name: 'login.types');
        Route::get(uri: '/Login/{type}', action: 'showLoginForm')->name(name: 'login.form');
        Route::post(uri: '/login', action: 'login')->name(name: 'login.submit');
        Route::get(uri: '/Logout', action: 'logout')->name(name: 'logout');
    });

Route::controller(controller: App\Http\Controllers\DashboardController::class)
    ->group(callback: function (): void {
        Route::get(uri: '/', action: 'index')->name(name: 'dashboard');
        Route::post(uri: '/', action: 'search')->name(name: 'dashboard.search');
    });

Route::middleware(middleware: 'auth')->group(callback: function (): void {
    Route::view(uri: '/dashboard', view: 'dashboard')
        ->middleware(middleware: 'role:admin,walikelas')
        ->name(name: 'dashboard.home');

    Route::resource(name: 'kasus', controller: App\Http\Controllers\KasusController::class)->except(methods: 'show')->middleware
});
```

Membuat Controller Kasus

Langkah selanjutnya adalah membuat controller untuk menangani operasi CRUD pada data kasus. Gunakan Artisan untuk membuat controller baru.

Jalankan perintah berikut di terminal:

```
php artisan make:controller KasusController -r
```

The screenshot shows the VS Code interface with the terminal tab active. The command `$ php artisan make:controller KasusController -r` has been run, and the terminal output shows:

```
INFO Controller [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\app\Http\Controllers\KasusController.php] created successfully.
```

Kemudian, buka file `KasusController.php` yang terletak di `app\Http\Controllers`, dan tambahkan method-method untuk menangani operasi CRUD seperti `index`, `create`, `store`, `edit`, `update`, dan `destroy`.

The screenshot shows the VS Code interface with the code editor tab active. The file `KasusController.php` is open in the editor. The code includes basic CRUD methods:

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
  
class KasusController extends Controller  
{  
    /**  
     * Display a listing of the resource.  
     */  
    public function index(): void  
    {  
        //  
    }  
  
    /**  
     * Show the form for creating a new resource.  
     */  
    public function create(): void  
    {  
        //  
    }  
  
    /**  
     * Store a newly created resource in storage.  
     */  
    public function store(Request $request): void  
    {  
        //  
    }  
  
    /**  
     * Display the specified resource.  
     */  
}
```

Penjelasan:

Pada Laravel, `Route::resource()` digunakan untuk mendefinisikan rute untuk resource controller yang secara otomatis akan menghasilkan beberapa rute sesuai dengan metode

CRUD (Create, Read, Update, Delete). Berikut adalah penjelasan singkat untuk tiap fungsi yang dihasilkan oleh `Route::resource()` :

1. `index()`

Fungsi ini digunakan untuk menampilkan daftar sumber daya (misalnya, semua kasus). Rute yang dihasilkan adalah `GET /kasus`.

2. `create()`

Fungsi ini digunakan untuk menampilkan form untuk membuat sumber daya baru (misalnya, form untuk membuat kasus baru). Rute yang dihasilkan adalah `GET /kasus/create`.

3. `store()`

Fungsi ini menangani penyimpanan data yang baru dibuat. Setelah formulir diisi dan dikirim, data diproses dan disimpan. Rute yang dihasilkan adalah `POST /kasus`.

4. `show()`

Fungsi ini digunakan untuk menampilkan detail dari sumber daya tertentu. Rute yang dihasilkan adalah `GET /kasus/{kasus}`. (Namun, dalam contoh Anda, `show()` dikecualikan dengan `except('show')`.)

5. `edit()`

Fungsi ini digunakan untuk menampilkan form untuk mengedit sumber daya yang ada. Rute yang dihasilkan adalah `GET /kasus/{kasus}/edit`.

6. `update()`

Fungsi ini menangani pembaruan data untuk sumber daya yang ada. Setelah form edit dikirim, data diperbarui. Rute yang dihasilkan adalah `PUT/PATCH /kasus/{kasus}`.

7. `destroy()`

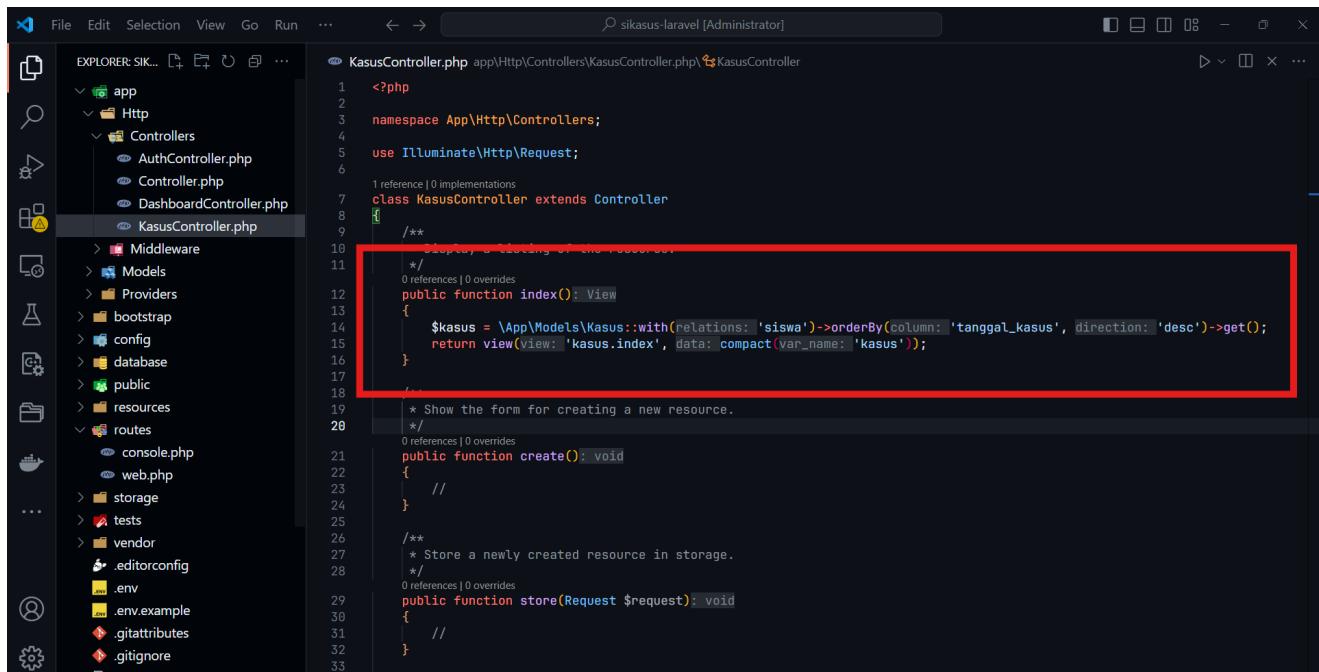
Fungsi ini digunakan untuk menghapus sumber daya yang ada. Rute yang dihasilkan adalah `DELETE /kasus/{kasus}`.

Dengan `Route::resource()`, Laravel memudahkan pengelolaan rute-rute yang berkaitan dengan operasi CRUD pada sumber daya, serta menghubungkannya langsung ke controller yang relevan.

Menampilkan Daftar Kasus (Method `index`)

Dalam method `index`, kita akan mengambil data semua kasus yang ada di database beserta data siswa yang terkait dengan menggunakan `Kasus::with('siswa')`. Data ini kemudian dikirim ke view `kasus.index` untuk ditampilkan.

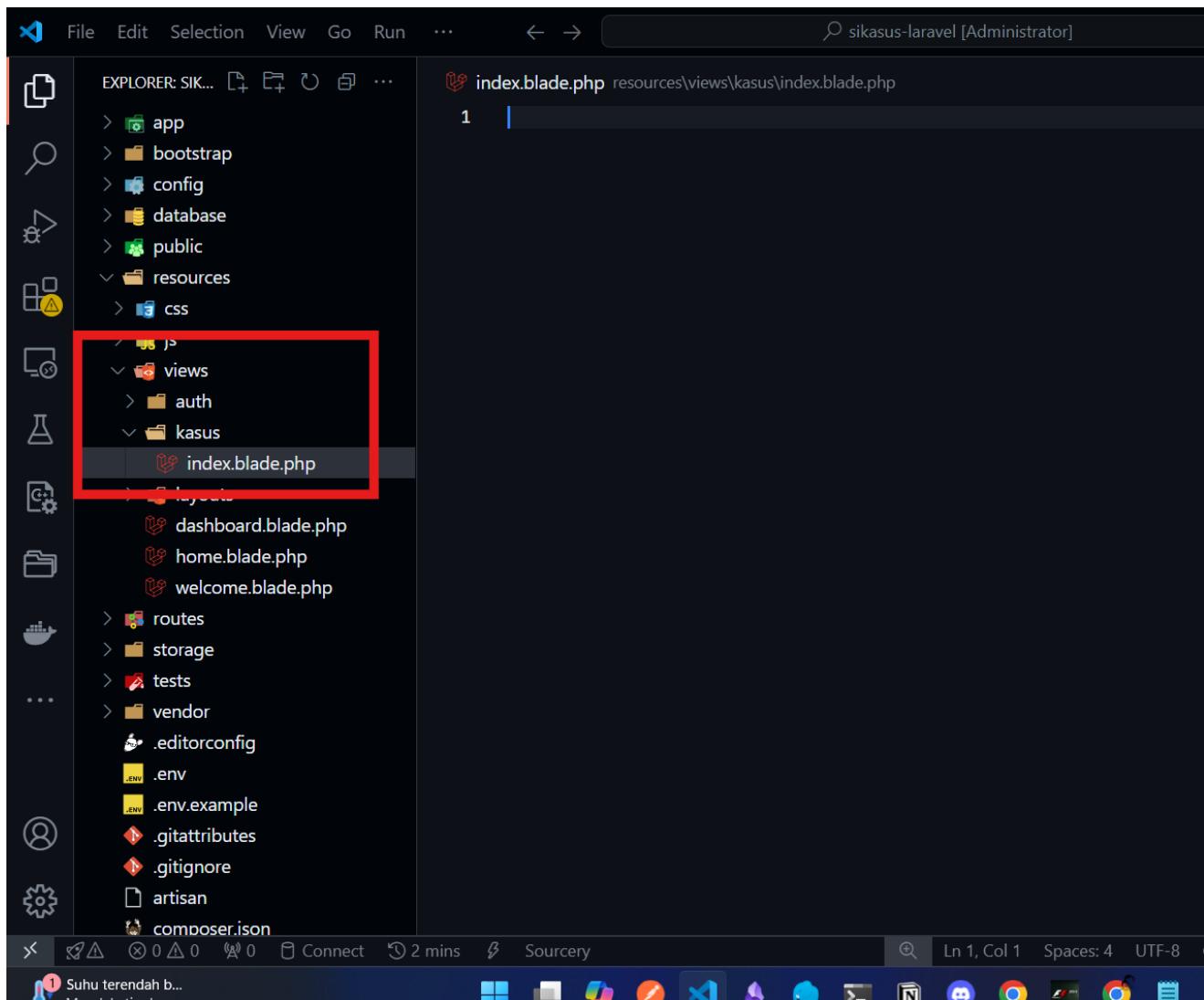
```
public function index()
{
    $kasus = \App\Models\Kasus::with('siswa')->orderBy('tanggal_kasus',
'desc')->get();
    return view('kasus.index', compact('kasus'));
}
```



```
<?php  
namespace App\\Http\\Controllers;  
use Illuminate\\Http\\Request;  
  
class KasusController extends Controller  
  
{  
    /**  
     * Show a listing of the resource.  
     *  
     * @return Response  
     */  
    public function index(): View  
    {  
        $kasus = \\App\\Models\\Kasus::with('siswa')->orderBy('tanggal_kasus', 'desc')->get();  
        return view('kasus.index', compact('kasus'));  
    }  
  
    /**  
     * Show the form for creating a new resource.  
     *  
     * @return Response  
     */  
    public function create(): void  
    {  
        //  
    }  
  
    /**  
     * Store a newly created resource in storage.  
     *  
     * @param Request $request  
     * @return Response  
     */  
    public function store(Request $request): void  
    {  
        //  
    }  
}
```

Membuat View untuk Menampilkan Kasus (View index)

Langkah selanjutnya adalah membuat tampilan untuk menampilkan daftar **Kasus**. Buat file `resources/views/kasus/index.blade.php`



```
EXPLORER: SIK... File Edit Selection View Go Run ... ← → siker [Administrator]  
index.blade.php resources\\views\\kasus\\index.blade.php  
1 |  
|> app  
|> bootstrap  
|> config  
|> database  
|> public  
> resources  
|> css  
|> js  
|> views  
|> auth  
> kasus  
|> index.blade.php  
|> dashboard.blade.php  
|> home.blade.php  
|> welcome.blade.php  
> routes  
> storage  
> tests  
> vendor  
.editorconfig  
.env  
.env.example  
.gitattributes  
.gitignore  
artisan  
composer.json
```

Kemudian, masukkan kode untuk menampilkan data kasus:

```
@extends('layouts.app')

@section('title', 'Daftar Kasus')

@section('content')
    <h1>Daftar Kasus</h1>
    <a href="{{ route('kasus.create') }}" class="my-2 btn btn-primary">Tambah Kasus</a>

    <div class="card mb-4">
        <div class="card-body">
            <table id="datatablesSimple">
                <thead>
                    <tr>
                        <th>#</th>
                        <th>Siswa</th>
                        <th>Kasus</th>
                        <th>Tanggal Kasus</th>
                        <th>Aksi</th>
                    </tr>
                </thead>
                <tfoot>
                    <tr>
                        <th>#</th>
                        <th>Siswa</th>
                        <th>Kasus</th>
                        <th>Tanggal Kasus</th>
                        <th>Aksi</th>
                    </tr>
                </tfoot>
                <tbody>
                    @foreach ($kasus as $item)
                        <tr>
                            <td>{{ $loop->iteration }}</td>
                            <td>{{ $item->siswa->nama_lengkap ?? '' }}</td>
                            <td>{{ $item->deskripsi_kasus }}</td>
                            <td>{{ $item->tanggal_kasus->format('d M Y') }}</td>
                        <td>
                            <td>
                                <a href="{{ route('kasus.edit', $item->id_kasus) }}" class="btn btn-warning btn-sm">Edit</a>
                            </td>
                        </td>
                    </tr>
                @endforeach
            </tbody>
        </table>
    </div>
```

```

        </div>
    @endsection

```

```

EXPLORER: SIK...
File Edit Selection View Go Run ... ⌂ ⌄ siskasus-laravel [Administrator]
index.blade.php resources\views\kasus\index.blade.php...
1 @extends('layouts.app')
2
3 @section('title', 'Daftar Kasus')
4
5 @section('content')
6     <h1>Daftar Kasus</h1>
7     <a href="{{ route('kasus.create') }}" class="my-2 btn btn-primary">Tambah Kasus</a>
8
9     <div class="card mb-4">
10        <div class="card-body">
11            <table id="datatablesSimple">
12                <thead>
13                    <tr>
14                        <th>#</th>
15                        <th>Siswa</th>
16                        <th>Kasus</th>
17                        <th>Tanggal Kasus</th>
18                        <th>Aksi</th>
19                    </tr>
20                </thead>
21                <tfoot>
22                    <tr>
23                        <th>#</th>
24                        <th>Siswa</th>
25                        <th>Kasus</th>
26                        <th>Tanggal Kasus</th>
27                        <th>Aksi</th>
28                    </tr>
29                </tfoot>
30                <tbody>
31                    @foreach ($kasus as $item)
32                        <tr>
33                            <td>{{ $loop->iteration }}</td>
34                            <td>{{ $item->siswa->nama_lengkap ?? '' }}</td>
35                            <td>{{ $item->deskripsi_kasus }}</td>
36                            <td>{{ $item->tanggal_kasus->format('d M Y') }}</td>
37                        </tr>

```

Menambahkan Kasus Baru ke Dalam Database (Method create dan store)

Untuk menambahkan kasus baru, kita buat method `create` dan `store` di `KasusController`. Method `create` akan menampilkan form untuk menambah kasus baru, sedangkan method `store` akan menyimpan data kasus ke dalam database.

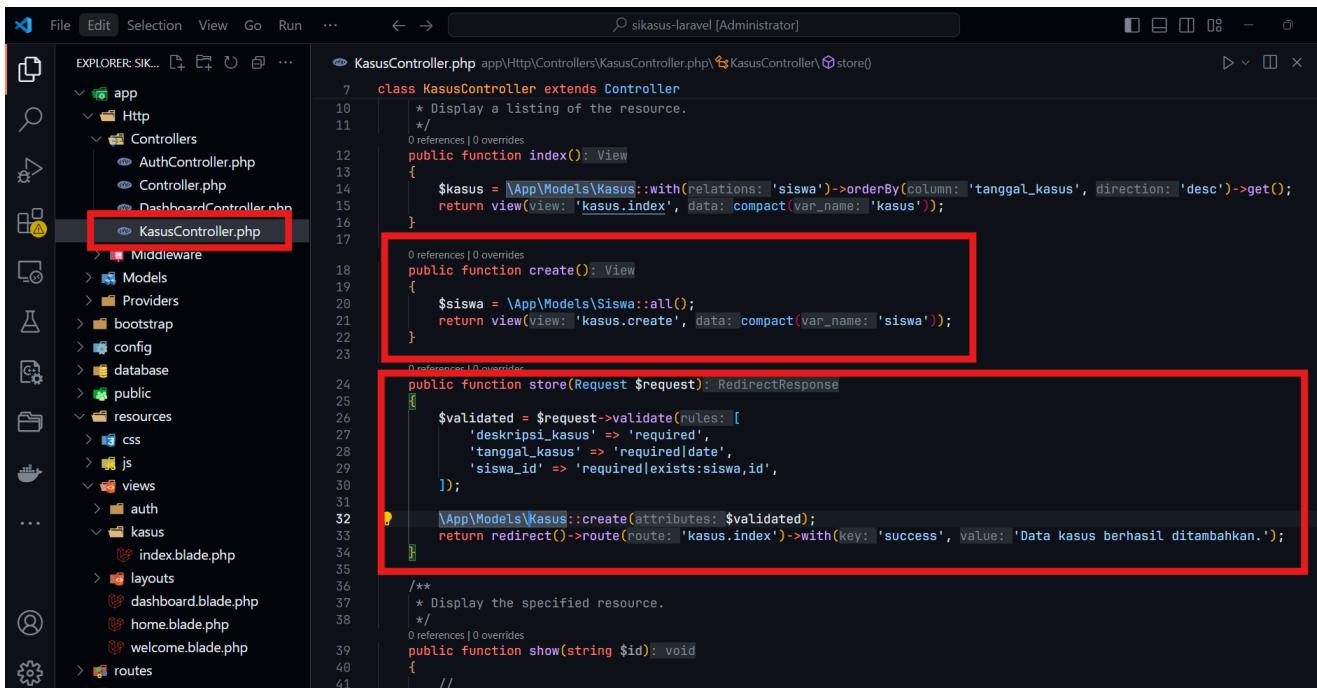
```

public function create()
{
    $siswa = \App\Models\Siswa::all();
    return view('kasus.create', compact('siswa'));
}

public function store(Request $request)
{
    $validated = $request->validate([
        'deskripsi_kasus' => 'required',
        'tanggal_kasus' => 'required|date',
        'siswa_id' => 'required|exists:siswa,id',
    ]);

    \App\Models\Kasus::create($validated);
    return redirect()->route('kasus.index')->with('success', 'Data kasus berhasil ditambahkan.');
}

```



```
class KasusController extends Controller
{
    /**
     * Display a listing of the resource.
     */
    public function index(): View
    {
        $kasus = \App\Models\Kasus::with('siswa')->orderBy(column: 'tanggal_kasus', direction: 'desc')->get();
        return view(view: 'kasus.index', data: compact(var_name: 'kasus'));
    }

    /**
     * Show the form for creating a new resource.
     */
    public function create(): View
    {
        $siswa = \App\Models\Siswa::all();
        return view(view: 'kasus.create', data: compact(var_name: 'siswa'));
    }

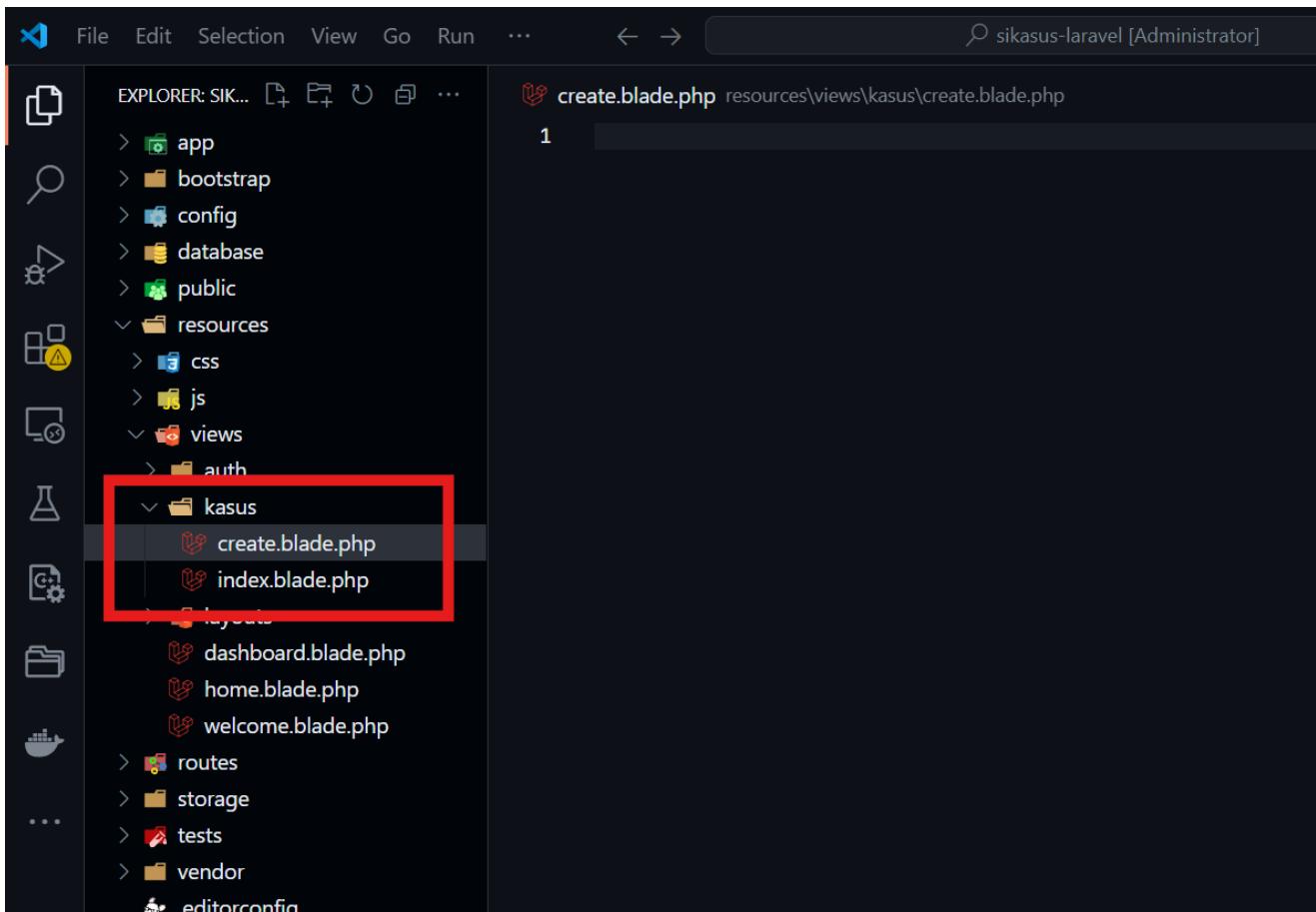
    /**
     * Store a newly created resource in storage.
     */
    public function store(Request $request): RedirectResponse
    {
        $validated = $request->validate(rules: [
            'deskripsi_kasus' => 'required',
            'tanggal_kasus' => 'required|date',
            'siswa_id' => 'required|exists:siswa,id',
        ]);

        \App\Models\Kasus::create(attributes: $validated);
        return redirect()->route(route: 'kasus.index')->with(key: 'success', value: 'Data kasus berhasil ditambahkan.');
    }

    /**
     * Display the specified resource.
     */
    public function show(string $id): void
    {
        //
    }
}
```

Membuat Form untuk Menambah Kasus (View create)

Buat file `resources/views/kasus/create.blade.php` untuk menampilkan form tambah kasus.



```
EXPLORER: SIK...
> app
> bootstrap
> config
> database
> public
< resources
> css
> js
< views
> auth
< kasus
> create.blade.php
> index.blade.php
> layouts
> dashboard.blade.php
> home.blade.php
> welcome.blade.php
> routes
> storage
> tests
> vendor
.editorconfig
```

Kemudian, masukkan kode untuk form tambah kasus :

```
@extends('layouts.app')
```

```
@section('content')


# Tambah Kasus Baru



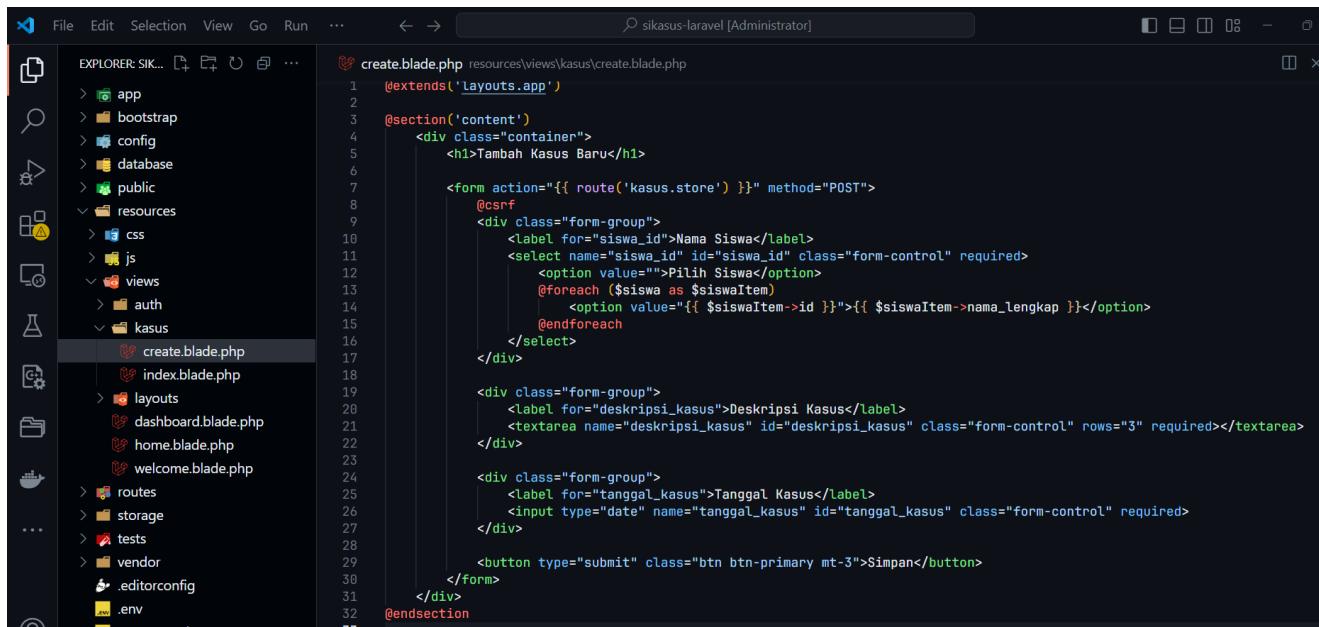
<form action="{{ route('kasus.store') }}" method="POST">
    @csrf
    <div class="form-group">
        <label for="siswa_id">Nama Siswa</label>
        <select name="siswa_id" id="siswa_id" class="form-control" required>
            <option value="">Pilih Siswa</option>
            @foreach ($siswa as $siswaItem)
                <option value="{{ $siswaItem->id }}>{{ $siswaItem->nama_lengkap }}</option>
            @endforeach
            </select>
        </div>

        <div class="form-group">
            <label for="deskripsi_kasus">Deskripsi Kasus</label>
            <textarea name="deskripsi_kasus" id="deskripsi_kasus" class="form-control" rows="3" required></textarea>
        </div>

        <div class="form-group">
            <label for="tanggal_kasus">Tanggal Kasus</label>
            <input type="date" name="tanggal_kasus" id="tanggal_kasus" class="form-control" required>
        </div>

        <button type="submit" class="btn btn-primary mt-3">Simpan</button>
    </form>
</div>
@endsection


```



```
CREATE: SIK...  resources\views\kasus\create.blade.php
1  @extends('layouts.app')
2
3  @section('content')
4      <div class="container">
5          <h1>Tambah Kasus Baru</h1>
6
7          <form action="{{ route('kasus.store') }}" method="POST">
8              @csrf
9              <div class="form-group">
10                 <label for="siswa_id">Nama Siswa</label>
11                 <select name="siswa_id" id="siswa_id" class="form-control" required>
12                     <option value="">Pilih Siswa</option>
13                     @foreach ($siswa as $siswaItem)
14                         <option value="{{ $siswaItem->id }}>{{ $siswaItem->nama_lengkap }}</option>
15                     @endforeach
16                 </select>
17             </div>
18
19             <div class="form-group">
20                 <label for="deskripsi_kasus">Deskripsi Kasus</label>
21                 <textarea name="deskripsi_kasus" id="deskripsi_kasus" class="form-control" rows="3" required></textarea>
22             </div>
23
24             <div class="form-group">
25                 <label for="tanggal_kasus">Tanggal Kasus</label>
26                 <input type="date" name="tanggal_kasus" id="tanggal_kasus" class="form-control" required>
27             </div>
28
29             <button type="submit" class="btn btn-primary mt-3">Simpan</button>
30         </form>
31     </div>
32 @endsection
```

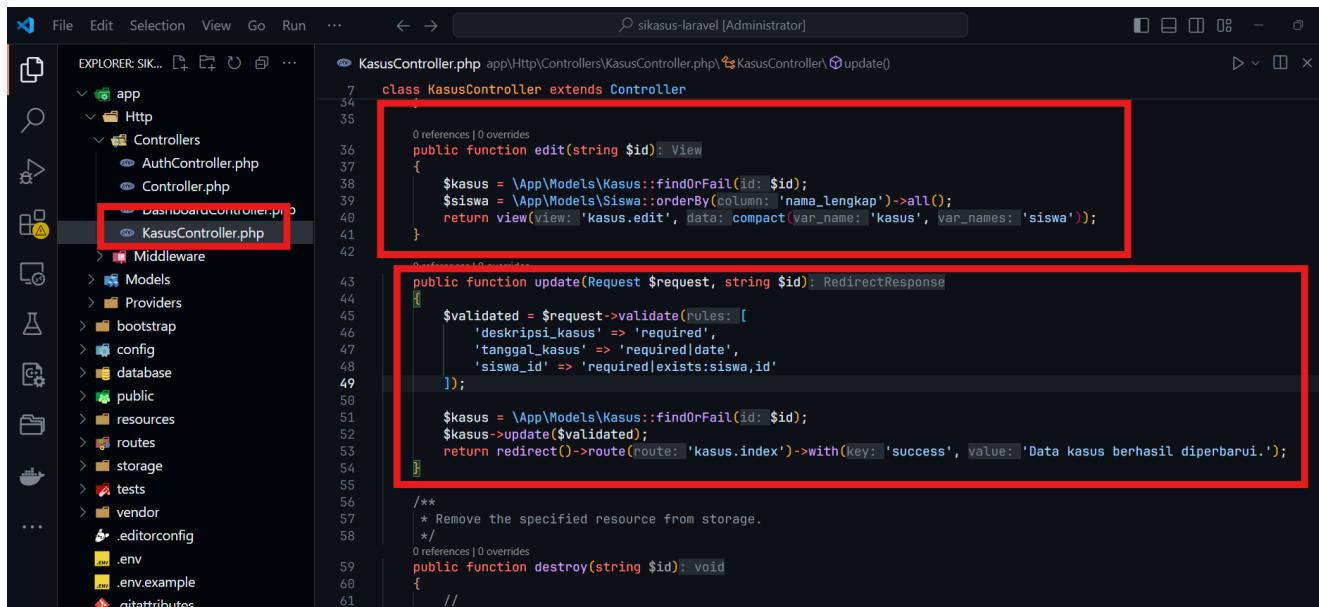
Edit dan Update Kasus ke Database

Tambahkan method `edit` dan `update` untuk mengedit dan memperbarui data kasus di controller.

```
public function edit(string $id)
{
    $kasus = \App\Models\Kasus::findOrFail($id);
    $siswa = \App\Models\Siswa::orderBy('nama_lengkap')->all();
    return view('kasus.edit', compact('kasus', 'siswa'));
}

public function update(Request $request, string $id)
{
    $validated = $request->validate([
        'deskripsi_kasus' => 'required',
        'tanggal_kasus' => 'required|date',
        'siswa_id' => 'required|exists:siswa,id'
    ]);

    $kasus = \App\Models\Kasus::findOrFail($id);
    $kasus->update($validated);
    return redirect()->route('kasus.index')->with('success', 'Data kasus berhasil diperbarui.');
}
```



```
class KasusController extends Controller
{
    public function edit(string $id): View
    {
        $kasus = \App\Models\Kasus::findOrFail($id);
        $siswa = \App\Models\Siswa::orderBy('nama_lengkap')->all();
        return view('kasus.edit', compact('kasus', 'siswa'));
    }

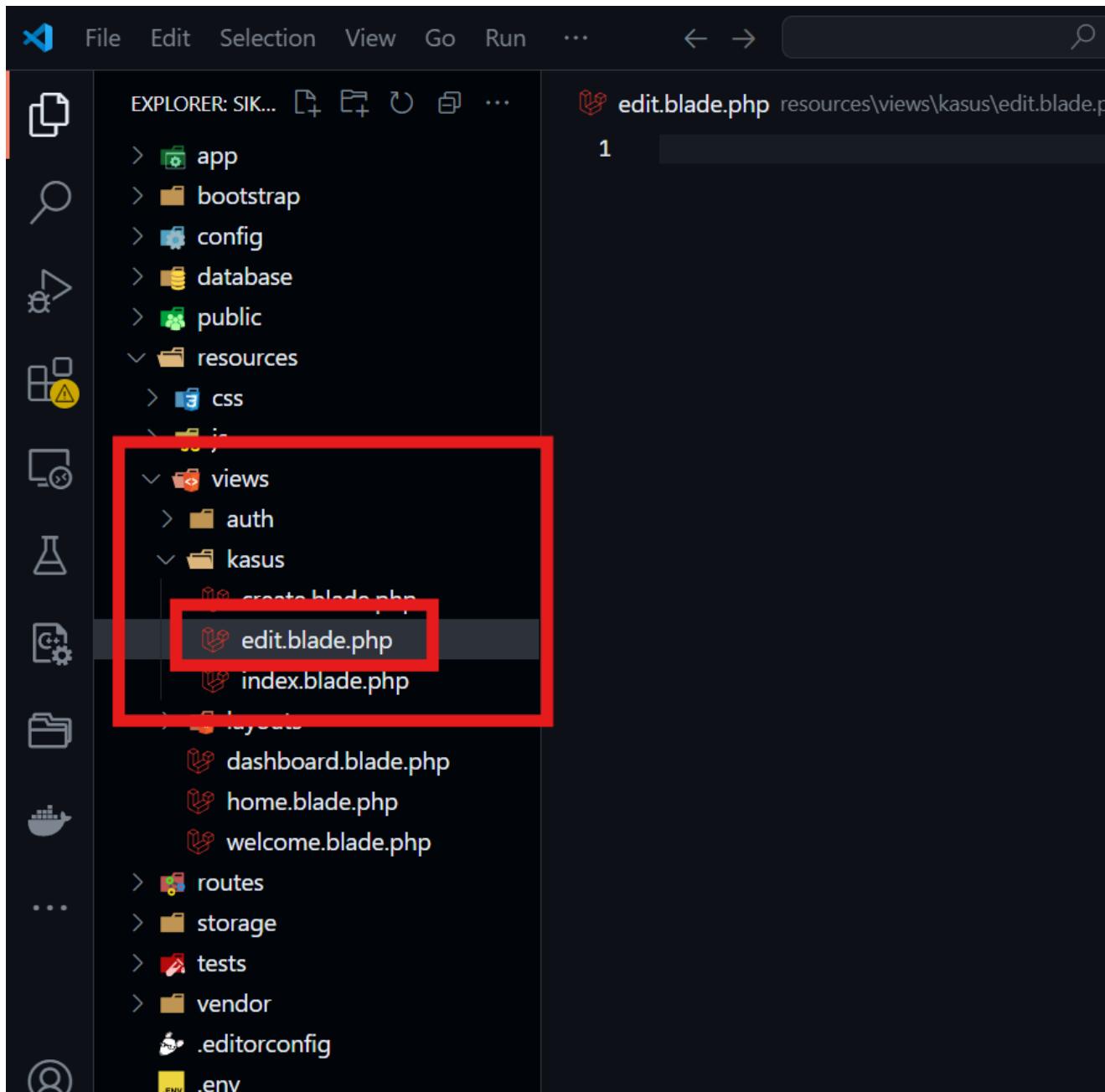
    public function update(Request $request, string $id): RedirectResponse
    {
        $validated = $request->validate([
            'deskripsi_kasus' => 'required',
            'tanggal_kasus' => 'required|date',
            'siswa_id' => 'required|exists:siswa,id'
        ]);

        $kasus = \App\Models\Kasus::findOrFail($id);
        $kasus->update($validated);
        return redirect()->route('kasus.index')->with(['success' => 'Data kasus berhasil diperbarui.']);
    }

    /**
     * Remove the specified resource from storage.
     */
    public function destroy(string $id): void
    {
        //
    }
}
```

Membuat View Form Edit Kasus (View edit)

Buat form untuk mengedit kasus di `resources/views/kasus/edit.blade.php`.



Kemudian, masukkan kode untuk form edit kasus :

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <h1>Edit Kasus</h1>

        <form action="{{ route('kasus.update', $kasus->id_kasus) }}" method="POST">
            @csrf
            @method('PUT')
            <div class="form-group">
                <label for="siswa_id">Nama Siswa</label>
                <select name="siswa_id" id="siswa_id" class="form-control" required>
                    @foreach ($siswa as $siswaItem)
                        <option value="{{ $siswaItem->id_siswa }}>{{ $siswaItem->nama_siswa }}</option>
                    @endforeach
            </div>
            <button type="submit" class="btn btn-primary">Simpan</button>
        </form>
    </div>
</div>
```

```

        <option value="{{ $siswaItem->id }}" {{ $siswaItem-
>id == $kasus->siswa_id ? 'selected' : '' }}>
            {{ $siswaItem->nama_lengkap }}
        </option>
    @endforeach
</select>
</div>

<div class="form-group">
    <label for="deskripsi_kasus">Deskripsi Kasus</label>
    <textarea name="deskripsi_kasus" id="deskripsi_kasus"
class="form-control" rows="3" required>{{ $kasus->deskripsi_kasus }}</textarea>
</div>

<div class="form-group">
    <label for="tanggal_kasus">Tanggal Kasus</label>
    <input type="date" name="tanggal_kasus" id="tanggal_kasus"
class="form-control" value="{{ $kasus->tanggal_kasus->format('Y-m-d') }}"
required>
</div>

<button type="submit" class="btn btn-warning mt-3">Perbarui</button>
</form>
</div>
@endsection

```

```

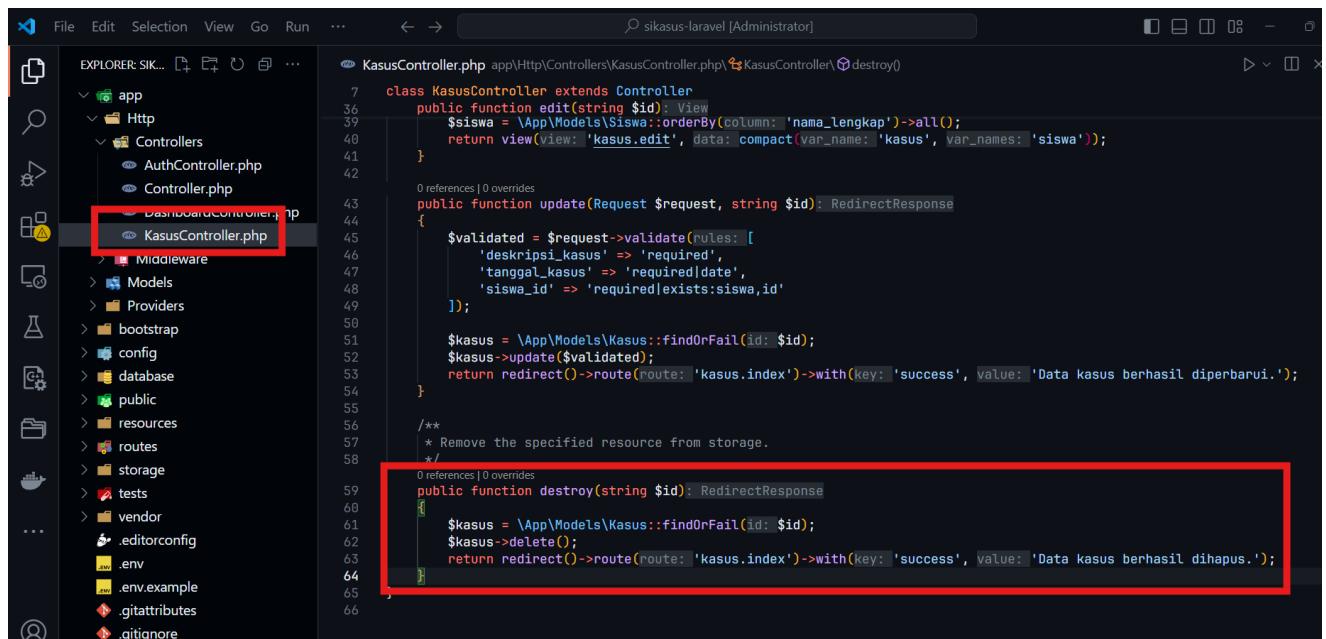
EXPLORER: SIK...  edit.blade.php resources\views\kasus\edit.blade.php
1  @extends('layouts.app')
2
3  @section('content')
4      <div class="container">
5          <h1>Edit Kasus</h1>
6
7          <form action="{{ route('kasus.update', $kasus->id_kasus) }}" method="POST">
8              @csrf
9              @method('PUT')
10             <div class="form-group">
11                 <label for="siswa_id">Nama Siswa</label>
12                 <select name="siswa_id" id="siswa_id" class="form-control" required>
13                     @foreach ($siswa as $siswaItem)
14                         <option value="{{ $siswaItem->id }}" {{ $siswaItem->id == $kasus->siswa_id ? 'selected' : '' }}>
15                             {{ $siswaItem->nama_lengkap }}
16                         </option>
17                     @endforeach
18                 </select>
19             </div>
20
21             <div class="form-group">
22                 <label for="deskripsi_kasus">Deskripsi Kasus</label>
23                 <textarea name="deskripsi_kasus" id="deskripsi_kasus" class="form-control" rows="3" required>{{ $kasus->deskripsi_kasus }}</textarea>
24             </div>
25
26             <div class="form-group">
27                 <label for="tanggal_kasus">Tanggal Kasus</label>
28                 <input type="date" name="tanggal_kasus" id="tanggal_kasus" class="form-control" value="{{ $kasus->tanggal_kasus->format('Y-m-d') }}"
29             </div>
30
31             <button type="submit" class="btn btn-warning mt-3">Perbarui</button>
32         </form>
33     </div>
34
35 @endsection

```

Menambahkan Method Destroy di Controller

Method `destroy` akan menghapus data kasus berdasarkan ID yang dipilih. Kode untuk method ini ada di dalam controller `KasusController`.

```
public function destroy(string $id)
{
    $kasus = \App\Models\Kasus::findOrFail($id);
    $kasus->delete();
    return redirect()->route('kasus.index')->with('success', 'Data kasus berhasil dihapus.');
}
```



Menambahkan Tombol Hapus di View

Tombol hapus ditambahkan di view `index.blade.php` agar admin dapat menghapus kasus yang ada.

```
<form action="{{ route('kasus.destroy', $item->id_kasus) }}" method="POST"
style="display:inline-block;">
    @csrf
    @method('DELETE')
    <button type="submit" class="btn btn-danger btn-sm" onclick="return
confirm('Yakin ingin menghapus?')">Hapus</button>
</form>
```

```

    <div class="card mb-4">
        <div class="card-body">
            <table id="datatablesSimple">
                <thead>
                    <tr>
                        <th>Siswa</th>
                        <th>Kasus</th>
                        <th>Tanggal Kasus</th>
                        <th>Aksi</th>
                    </tr>
                </thead>
                <tbody>
                    @foreach ($kasus as $item)
                    <tr>
                        <td>{{ $loop->iteration }}</td>
                        <td>{{ $item->siswa->nama_lengkap ?? '' }}</td>
                        <td>{{ $item->deskripsi_kasus }}</td>
                        <td>{{ $item->tanggal_kasus->format('d M Y') }}</td>
                    <td>
                        <a href="{{ route('kasus.edit', $item->id_kasus) }}" class="btn btn-warning btn-sm">Edit
                        <form action="{{ route('kasus.destroy', $item->id_kasus) }}" method="POST" style="display:inline-block;">
                            @csrf
                            @method('DELETE')
                            <button type="submit" class="btn btn-danger btn-sm" onclick="return confirm('Yakin ingin menghapus?')">Hapus
                        </form>
                    </td>
                </tr>
            @endforeach
        </tbody>
    </table>
</div>
</div>

```

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration		✓
	1. Migration Kasus	✓
	2. Migration Siswa	✓
	3. Migration Kelas	✓
	4. Migration Walikelas	✓
Menambahkan Mass Assignment		✓
	1. Model Kasus	✓
	2. Model Siswa	✓
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓
	Menampilkan Halaman Jenis Login	✓
	Menampilkan Form Login Berdasarkan Jenis	✓

Topik	Sub Topik	Status
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		✓
	Membuat Middleware untuk Autentikasi	✓
	Membuat Middleware untuk Role	✓
	Registrasi Middleware	✓
Membuat Blade Layout		✓
Membuat Tampilan Homepage		✓
Membuat Tampilan Dashboard Setelah Login		✓
Membuat Fitur Kasus		✓
	Menambahkan Routing untuk Kasus	✓
	Membuat Controller Kasus	✓
	Menampilkan Daftar Kasus (Method index)	✓
	Membuat View untuk Menampilkan Kasus (View index)	✓
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Kasus (View create)	✓
	Edit dan Update Kasus ke Database	✓
	Membuat View Form Edit Kasus (View edit)	✓
	Menambahkan Method Destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Siswa		
	Menambahkan Routing untuk Siswa	
	Membuat Controller Siswa	
	Menampilkan Daftar Siswa (Method index)	
	Membuat View untuk Menampilkan Siswa (View index)	

Topik	Sub Topik	Status
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Siswa (View create)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Walikelas		
	Menambahkan Routing untuk Walikelas	
	Membuat Controller Walikelas	
	Menampilkan Daftar Walikelas (Method index)	
	Membuat View untuk Menampilkan Walikelas (View index)	
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	

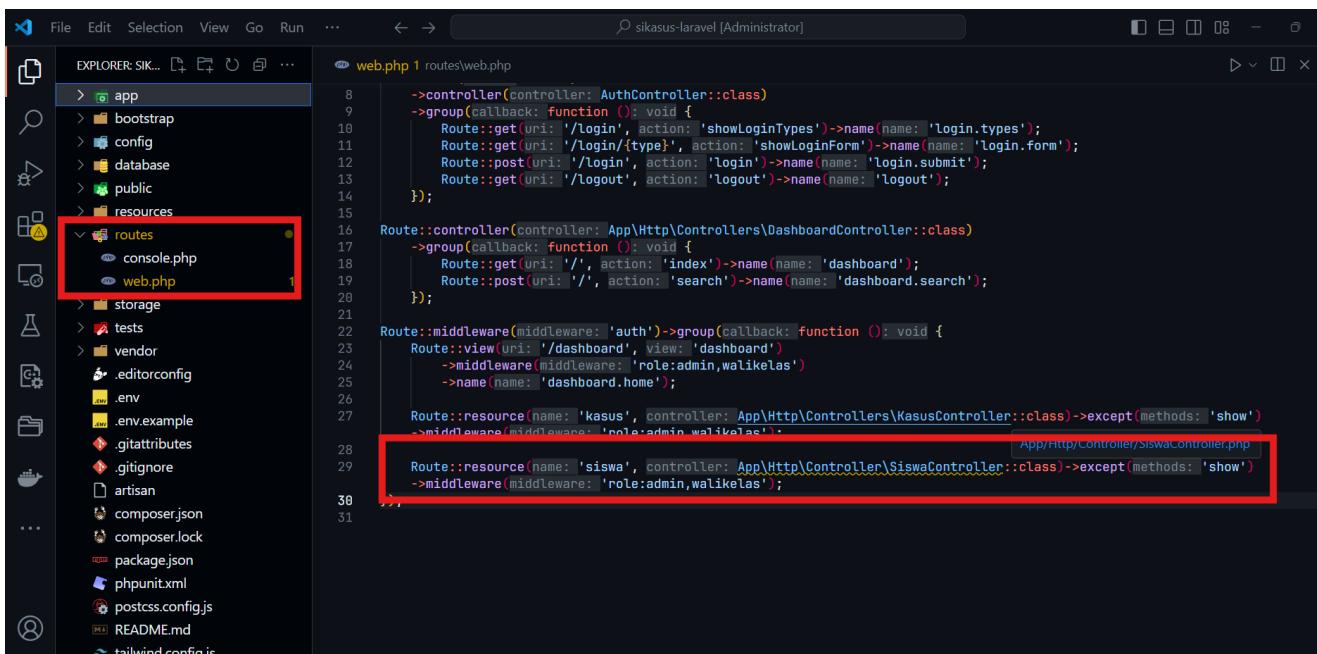
Topik	Sub Topik	Status
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk 'Siswa'	
	Membuat Controller 'Siswa'	

Membuat Fitur Siswa

Menambahkan Routing untuk Siswa

Langkah pertama adalah menambahkan routing yang akan menghubungkan URL dengan method di dalam controller **Siswa**. Untuk itu, buka file `routes/web.php` dan tambahkan routing berikut:

```
Route::resource('siswa', App\Http\Controllers\SiswaController::class)
    >except('show')->middleware('role:admin,walikelas');
```



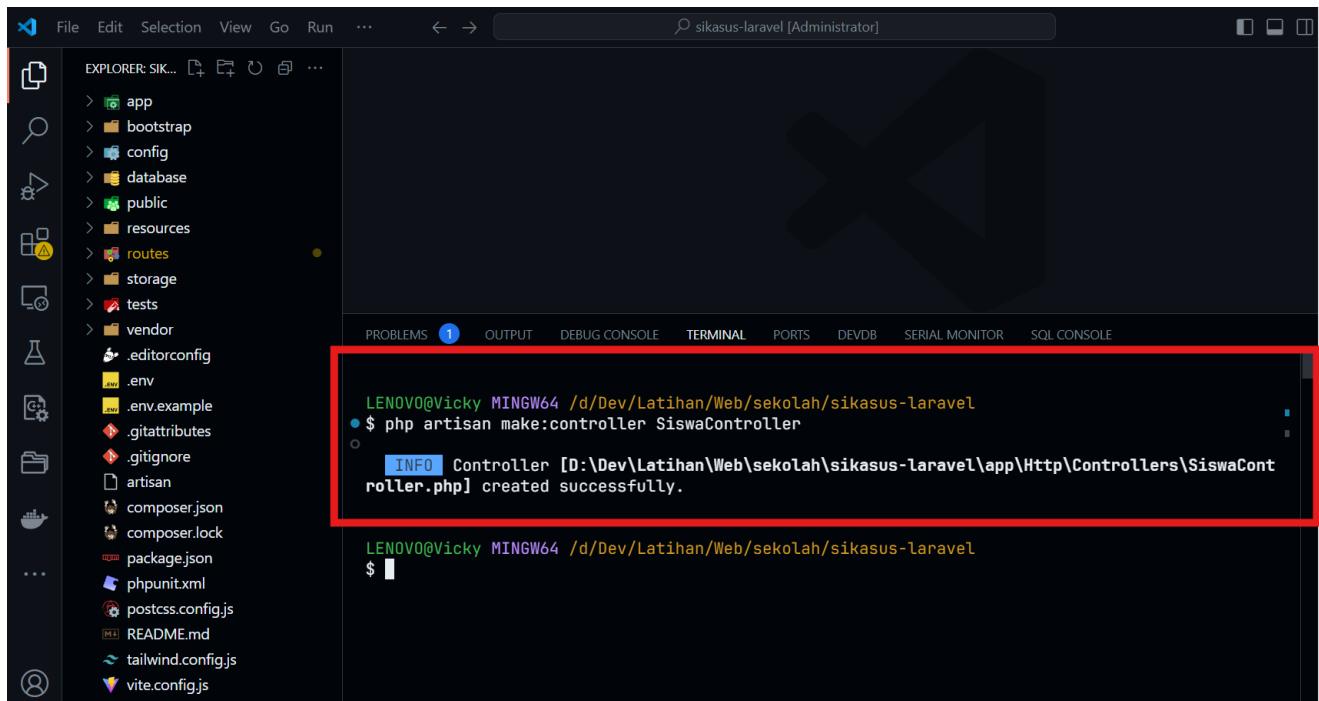
```
EXPLORER: SIK... File Edit Selection View Go Run ... ← → ⌂ sikerus-laravel [Administrator] D v x
routes
  routes
    console.php
    web.php
  app
    bootstrap
    config
    database
    public
    resources
      storage
      tests
      vendor
      .editorconfig
      .env
      .env.example
      .gitattributes
      .gitignore
      artisan
      composer.json
      composer.lock
      package.json
      phpunit.xml
      postcss.config.js
      README.md
      tailwind.config.js

web.php 1 routes\web.php
8   >>>controller(Controller: AuthController::class)
9   >>>group(callback: function () void {
10     Route::get(uri: '/login', action: 'showLoginTypes')->name(name: 'login.types');
11     Route::get(uri: '/login/{type}', action: 'showLoginForm')->name(name: 'login.form');
12     Route::post(uri: '/login', action: 'Login')->name(name: 'login.submit');
13     Route::get(uri: '/logout', action: 'Logout')->name(name: 'logout');
14   });
15
16   Route::controller(controller: App\Http\Controllers\DashboardController::class)
17   >>>group(callback: Function () void {
18     Route::get(uri: '/', action: 'index')->name(name: 'dashboard');
19     Route::post(uri: '/', action: 'search')->name(name: 'dashboard.search');
20   });
21
22   Route::middleware(middleware: 'auth')->group(callback: function () void {
23     Route::view(uri: '/dashboard', view: 'dashboard')
24       >>>middleware(middleware: 'role:admin,walikelas')
25       >>>name(name: 'dashboard.home');
26
27   Route::resource(name: 'kasus', controller: App\Http\Controllers\KasusController::class)->except(methods: 'show')
28     >>>middleware(middleware: 'role:admin,walikelas');
29
30   Route::resource(name: 'siswa', controller: App\Http\Controllers\SiswaController::class)->except(methods: 'show')
31     >>>middleware(middleware: 'role:admin,walikelas');
```

Membuat Controller Siswa

Setelah menambahkan routing, langkah berikutnya adalah membuat controller untuk menangani semua proses terkait data **Siswa**. Jalankan perintah berikut untuk membuat controller `SiswaController`:

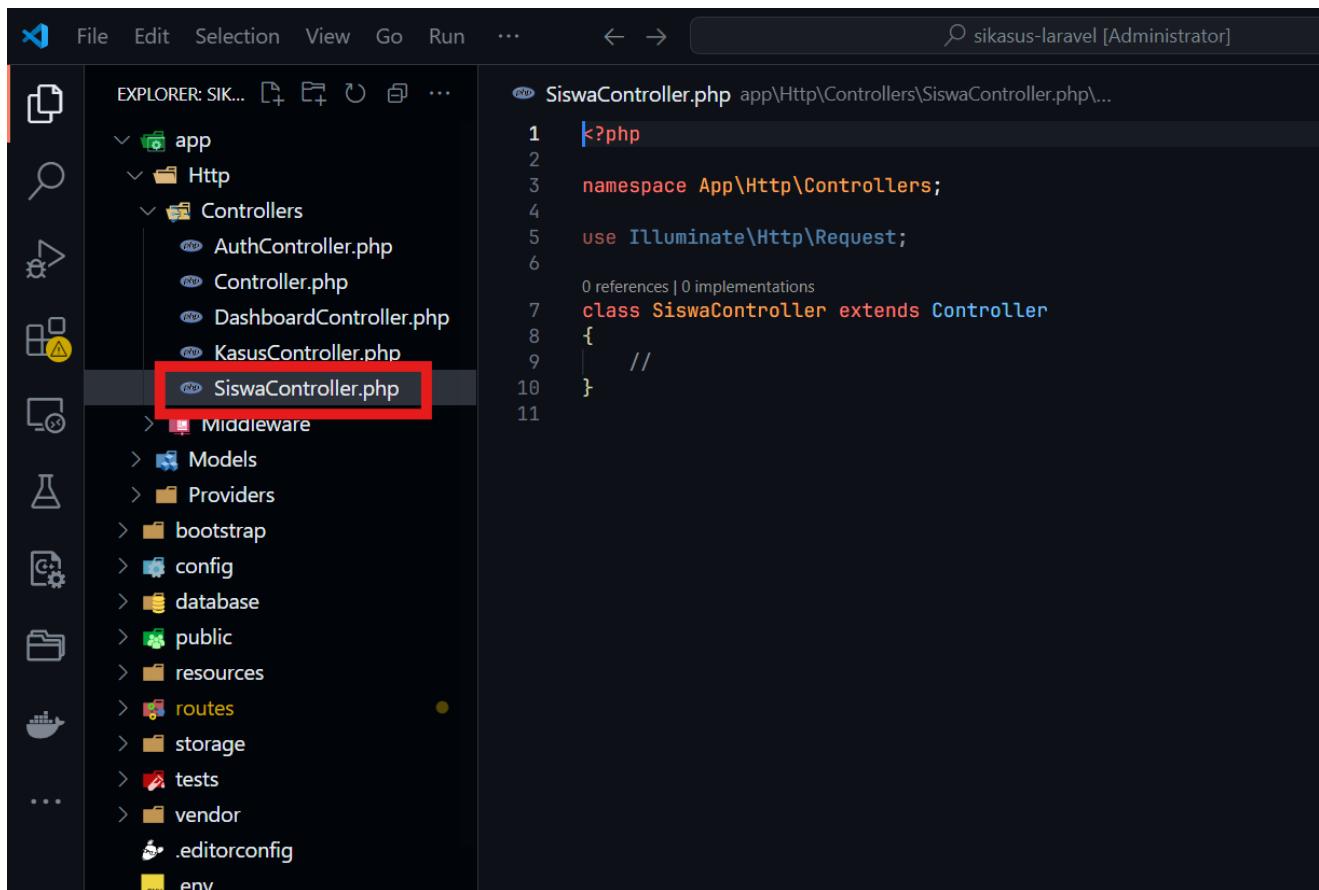
```
php artisan make:controller SiswaController
```



```
LENovo@Vicky MINGW64 /d/Dev/Latihan/Web/sekolah/sikasus-laravel
$ php artisan make:controller SiswaController
INFO Controller [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\app\Http\Controllers\SiswaController.php] created successfully.

LENovo@Vicky MINGW64 /d/Dev/Latihan/Web/sekolah/sikasus-laravel
$
```

Buka file `SiswaController.php` yang terletak di folder `app/Http/Controllers`. Di dalam controller ini, kita akan membuat berbagai method untuk menampilkan, menambah, mengedit, dan menghapus data **Siswa**.



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 0 references | 0 implementations
8 class SiswaController extends Controller
9 {
10     //
11 }
```

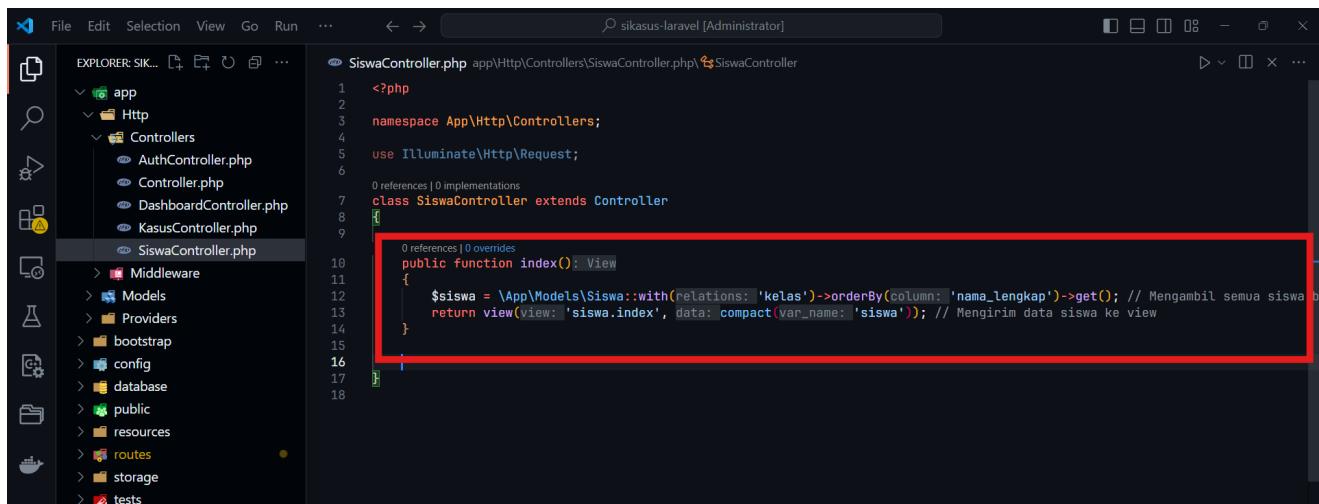
Menampilkan Daftar Siswa (Method index)

Untuk menampilkan daftar siswa, buat method `index` dalam controller yang mengambil semua data **Siswa** beserta informasi kelasnya dan mengirimnya ke view. Berikut adalah kode untuk fungsi `index` :

```

public function index()
{
    $siswa = \App\Models\Siswa::with('kelas')->orderBy('nama_lengkap')-
>get(); // Mengambil semua siswa beserta informasi kelasnya
    return view('siswa.index', compact('siswa')); // Mengirim data siswa ke
view
}

```



The screenshot shows a code editor interface with a dark theme. On the left is a sidebar labeled 'EXPLORER: SIK...' containing a tree view of project files under 'app'. The main area shows the 'SiswaController.php' file. The code is as follows:

```

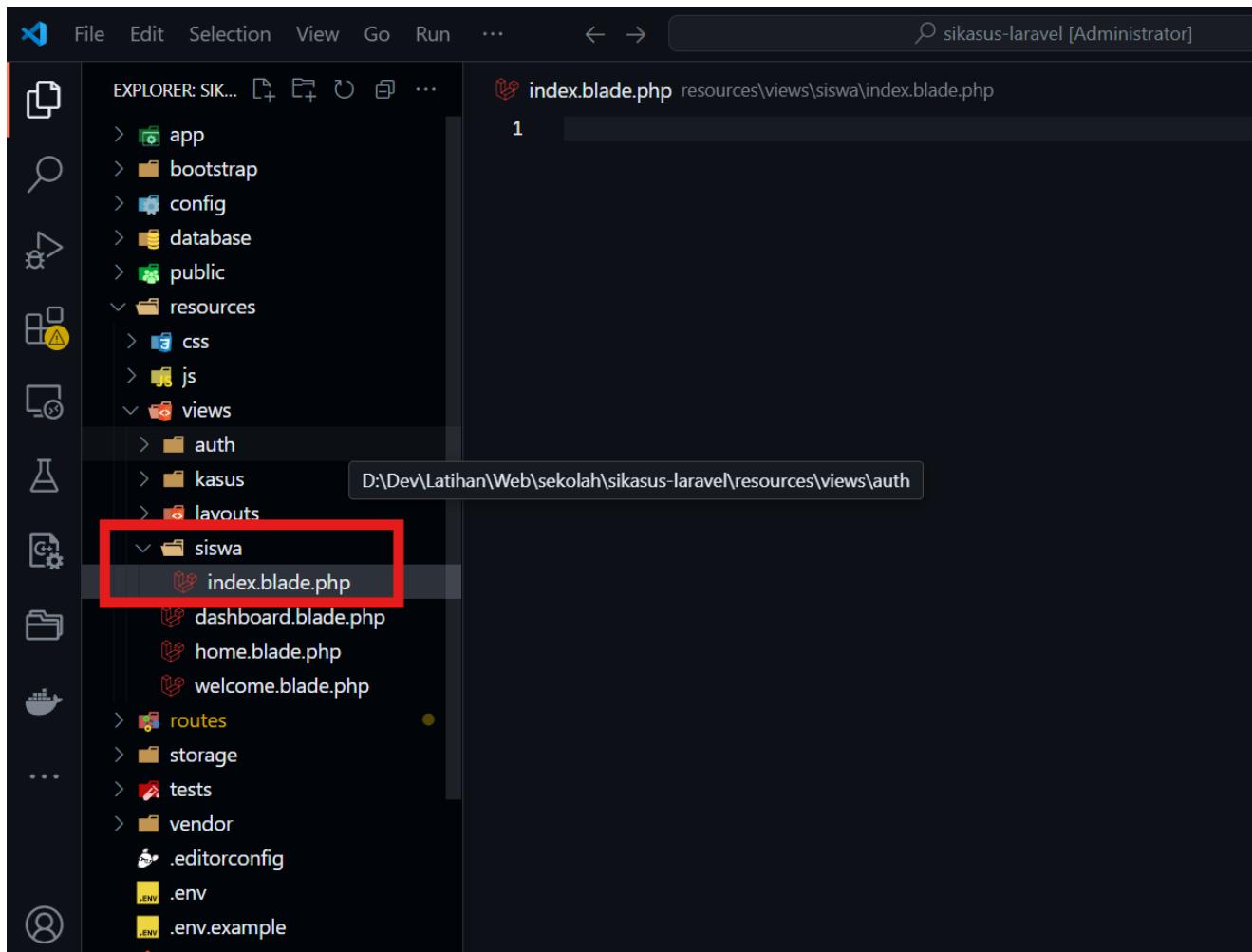
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class SiswaController extends Controller
{
    public function index(): View
    {
        $siswa = \App\Models\Siswa::with('kelas')->orderBy('nama_lengkap')->get(); // Mengambil semua siswa b
        return view('view: 'siswa.index', data: compact(var_name: 'siswa')) // Mengirim data siswa ke view
    }
}

```

The code block from line 10 to line 17 is highlighted with a red rectangle.

Membuat View untuk Menampilkan Siswa (View index)

Setelah controller dibuat, langkah selanjutnya adalah membuat tampilan untuk menampilkan daftar **Siswa**. Buat file `resources/views/siswa/index.blade.php`



Kemudian, buat kode menampilkan siswa :

```
@extends('layouts.app')

@section('title', 'Daftar Siswa')

@section('content')
    <h1>Daftar Siswa</h1>
    <a href="{{ route('siswa.create') }}" class="my-2 btn btn-primary">Tambah Siswa</a>

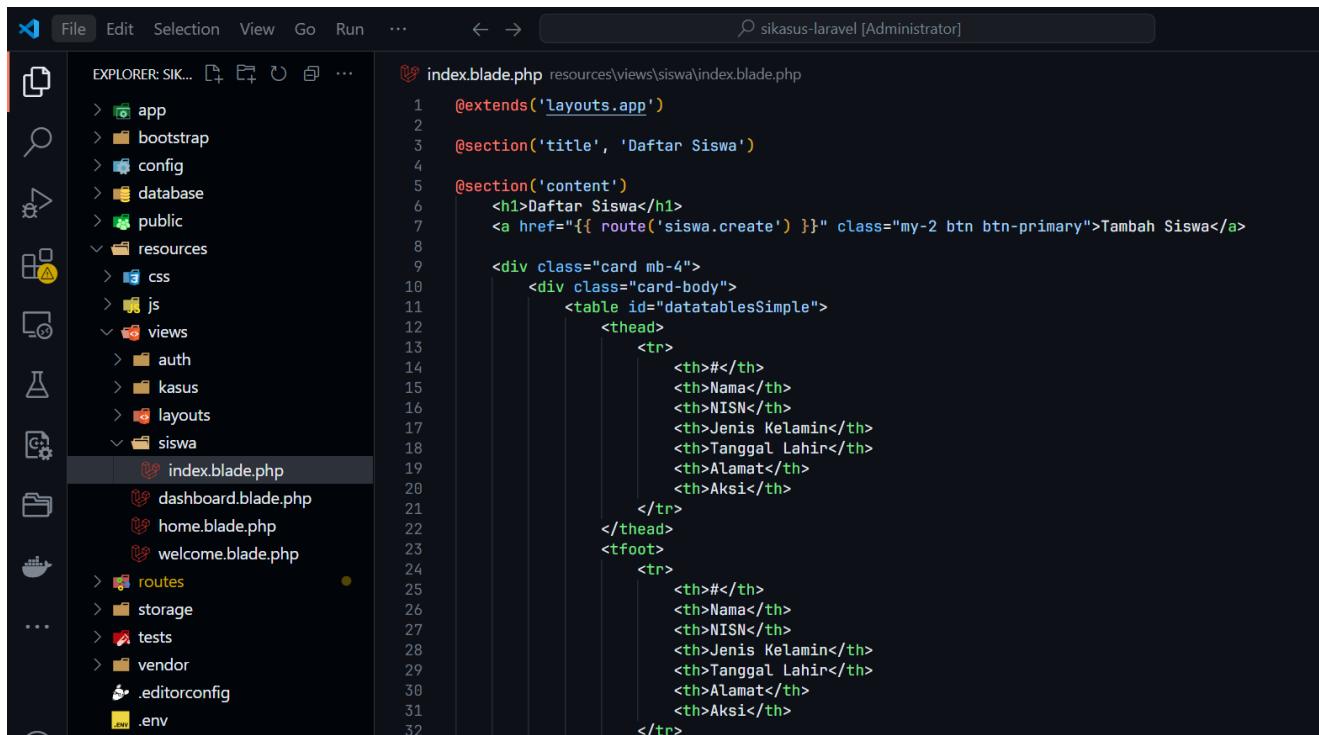
    <div class="card mb-4">
        <div class="card-body">
            <table id="datatablesSimple">
                <thead>
                    <tr>
                        <th>#</th>
                        <th>Nama</th>
                        <th>NISN</th>
                        <th>Jenis Kelamin</th>
                        <th>Tanggal Lahir</th>
                        <th>Alamat</th>
                        <th>Aksi</th>
                    </tr>
                </thead>
                <tbody>
                </tbody>
            </table>
        </div>
    </div>

```

```

</thead>
<tfoot>
    <tr>
        <th>#</th>
        <th>Nama</th>
        <th>NISN</th>
        <th>Jenis Kelamin</th>
        <th>Tanggal Lahir</th>
        <th>Alamat</th>
        <th>Aksi</th>
    </tr>
</tfoot>
<tbody>
    @foreach ($siswa as $item)
        <tr>
            <td>{{ $loop->iteration }}</td>
            <td>{{ $item->nama_lengkap }}</td>
            <td>{{ $item->nisn }}</td>
            <td>{{ $item->jenis_kelamin }}</td>
            <td>{{ $item->tanggal_lahir }}</td>
            <td>{{ $item->alamat }}</td>
            <td>
                <a href="{{ route('siswa.edit', $item->id) }}>Edit</a>
            </td>
        </tr>
    @endforeach
</tbody>
</table>
</div>
</div>
@endsection

```



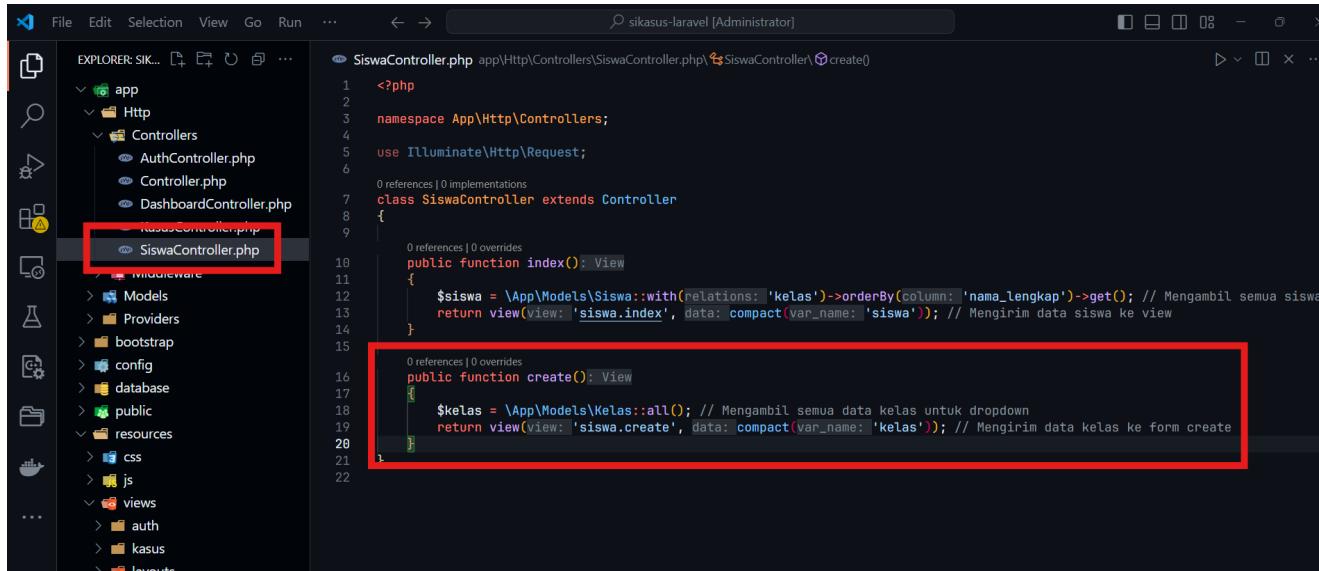
```
index.blade.php resources\views\siswa\index.blade.php
1  @extends('layouts.app')
2
3  @section('title', 'Daftar Siswa')
4
5  @section('content')
6      <h1>Daftar Siswa</h1>
7      <a href="{{ route('siswa.create') }}" class="my-2 btn btn-primary">Tambah Siswa</a>
8
9      <div class="card mb-4">
10         <div class="card-body">
11             <table id="datatablesSimple">
12                 <thead>
13                     <tr>
14                         <th>#</th>
15                         <th>Nama</th>
16                         <th>NISN</th>
17                         <th>Jenis Kelamin</th>
18                         <th>Tanggal Lahir</th>
19                         <th>Alamat</th>
20                         <th>Aksi</th>
21                     </tr>
22                 </thead>
23                 <tfoot>
24                     <tr>
25                         <th>#</th>
26                         <th>Nama</th>
27                         <th>NISN</th>
28                         <th>Jenis Kelamin</th>
29                         <th>Tanggal Lahir</th>
30                         <th>Alamat</th>
31                         <th>Aksi</th>
32                     </tr>

```

Menambahkan Siswa Baru ke Dalam Database (Method create dan store)

Setelah menampilkan daftar siswa, langkah selanjutnya adalah membuat form untuk menambah data **Siswa** baru. Pertama, buat method `create` di controller untuk mengambil data kelas dan mengirimkannya ke form.

```
public function create()
{
    $kelas = \App\Models\Kelas::all(); // Mengambil semua data kelas untuk dropdown
    return view('siswa.create', compact('kelas')); // Mengirim data kelas ke form create
}
```

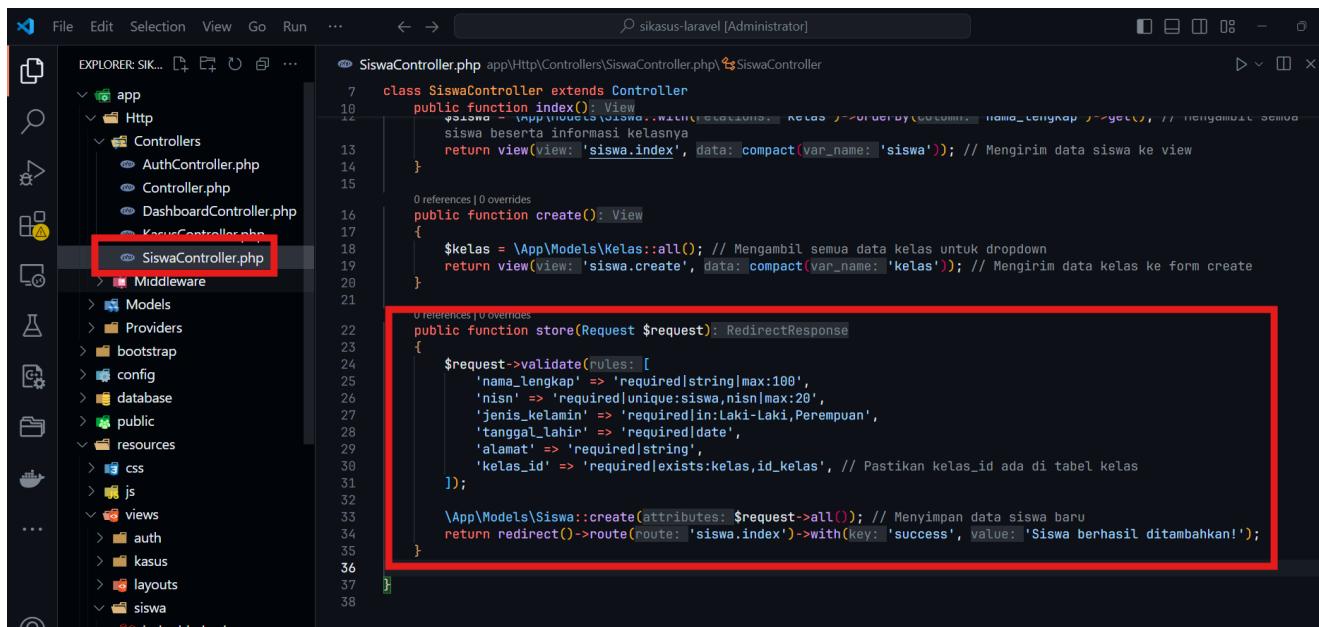


```
SiswaController.php app\Http\Controllers\SiswaController.php\create
1  <?php
2
3  namespace App\Http\Controllers;
4
5  use Illuminate\Http\Request;
6
7  0 references | 0 implementations
8  class SiswaController extends Controller
9  {
10
11     0 references | 0 overrides
12     public function index(): View
13     {
14         $siswa = \App\Models\Siswa::with('kelas')->orderBy(column: 'nama_lengkap')->get(); // Mengambil semua siswa
15         return view(view: 'siswa.index', data: compact(var_name: 'siswa')) // Mengirim data siswa ke view
16     }
17
18     0 references | 0 overrides
19     public function create(): View
20     {
21         $kelas = \App\Models\Kelas::all(); // Mengambil semua data kelas untuk dropdown
22         return view(view: 'siswa.create', data: compact(var_name: 'kelas')) // Mengirim data kelas ke form create
23     }
24
25
26
27
28
29
30
31
32
```

Selanjutnya, buat method `store` untuk menangani penyimpanan data **Siswa** yang baru ditambahkan.

```
public function store(Request $request)
{
    $request->validate([
        'nama_lengkap' => 'required|string|max:100',
        'nisn' => 'required|unique:siswa,nisn|max:20',
        'jenis_kelamin' => 'required|in:Laki-Laki,Perempuan',
        'tanggal_lahir' => 'required|date',
        'alamat' => 'required|string',
        'kelas_id' => 'required|exists:kelas,id_kelas', // Pastikan kelas_id ada di tabel kelas
    ]);

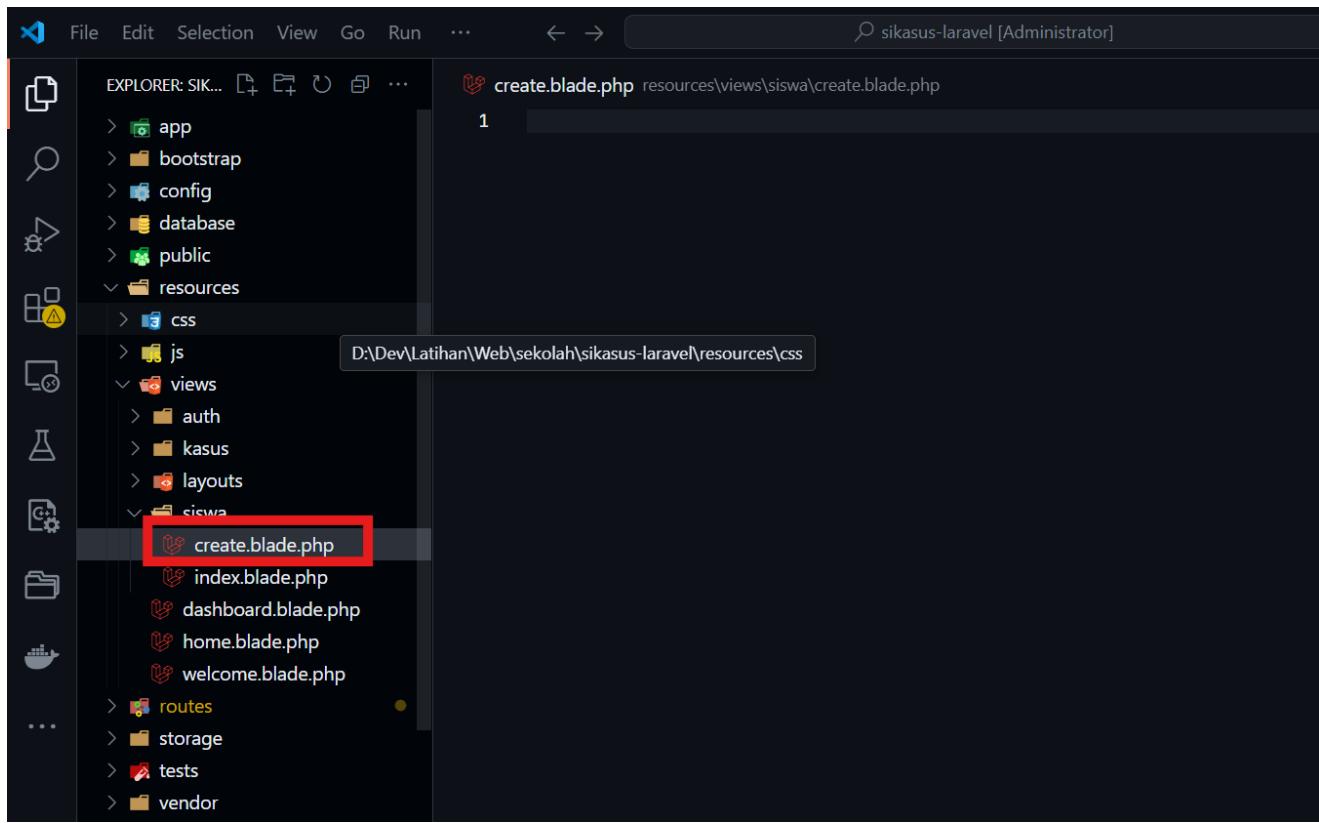
    \App\Models\Siswa::create($request->all()); // Menyimpan data siswa baru
    return redirect()->route('siswa.index')->with('success', 'Siswa berhasil ditambahkan!');
}
```



```
File Edit Selection View Go Run ... < > siklus-laravel [Administrator] D X
EXPLORER: SIK... E F U ... SiswaController.php app\Http\Controllers\SiswaController.php SiswaController
7 class SiswaController extends Controller
10 public function index(): View
11     {
12         $siswa = \App\Models\Siswa::with(['kelas'])->orderBy('id', 'asc');
13         return view('view: 'siswa.index', data: compact(var_name: 'siswa'));// Mengirim data siswa ke view
14     }
15
16     0 references | 0 overrides
17     public function create(): View
18     {
19         $kelas = \App\Models\Kelas::all(); // Mengambil semua data kelas untuk dropdown
20         return view('view: 'siswa.create', data: compact(var_name: 'kelas'));// Mengirim data kelas ke form create
21     }
22
23     0 references | 0 overrides
24     public function store(Request $request): RedirectResponse
25     {
26         $request->validate(rules: [
27             'nama_lengkap' => 'required|string|max:100',
28             'nisn' => 'required|unique:siswa,nisn|max:20',
29             'jenis_kelamin' => 'required|in:Laki-Laki,Perempuan',
30             'tanggal_lahir' => 'required|date',
31             'alamat' => 'required|string',
32             'kelas_id' => 'required|exists:kelas,id_kelas', // Pastikan kelas_id ada di tabel kelas
33         ]);
34
35         \App\Models\Siswa::create(attributes: $request->all()); // Menyimpan data siswa baru
36         return redirect()->route(route: 'siswa.index')->with(key: 'success', value: 'Siswa berhasil ditambahkan!');
37     }
38 }
```

Membuat Form untuk Menambah Siswa (View create)

Pertama, buat file `resources/views/siswa/create.blade.php` untuk menampilkan form pendaftaran **Siswa** baru.



Kemudian, membuat kode formulir pendaftaran siswa :

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <h1>Tambah Siswa Baru</h1>

        <form action="{{ route('siswa.store') }}" method="POST">
            @csrf
            <div class="form-group">
                <label for="nama_lengkap">Nama Lengkap</label>
                <input type="text" name="nama_lengkap" id="nama_lengkap" class="form-control" required>
            </div>

            <div class="form-group">
                <label for="nisn">NISN</label>
                <input type="text" name="nisn" id="nisn" class="form-control" required>
            </div>

            <div class="form-group">
                <label for="jenis_kelamin">Jenis Kelamin</label>
                <select name="jenis_kelamin" id="jenis_kelamin" class="form-control" required>
                    <option value="Laki-Laki">Laki-Laki</option>
                    <option value="Perempuan">Perempuan</option>
                </select>
            </div>
        </form>
    </div>

```

```

        </div>

        <div class="form-group">
            <label for="tanggal_lahir">Tanggal Lahir</label>
            <input type="date" name="tanggal_lahir" id="tanggal_lahir" class="form-control" required>
        </div>

        <div class="form-group">
            <label for="alamat">Alamat</label>
            <textarea name="alamat" id="alamat" class="form-control" required></textarea>
        </div>

        <div class="form-group">
            <label for="kelas_id">Kelas</label>
            <select name="kelas_id" id="kelas_id" class="form-control" required>
                @foreach ($kelas as $k)
                    <option value="{{ $k->id_kelas }}>{{ $k->nama_kelas }}</option>
                @endforeach
            </select>
        </div>

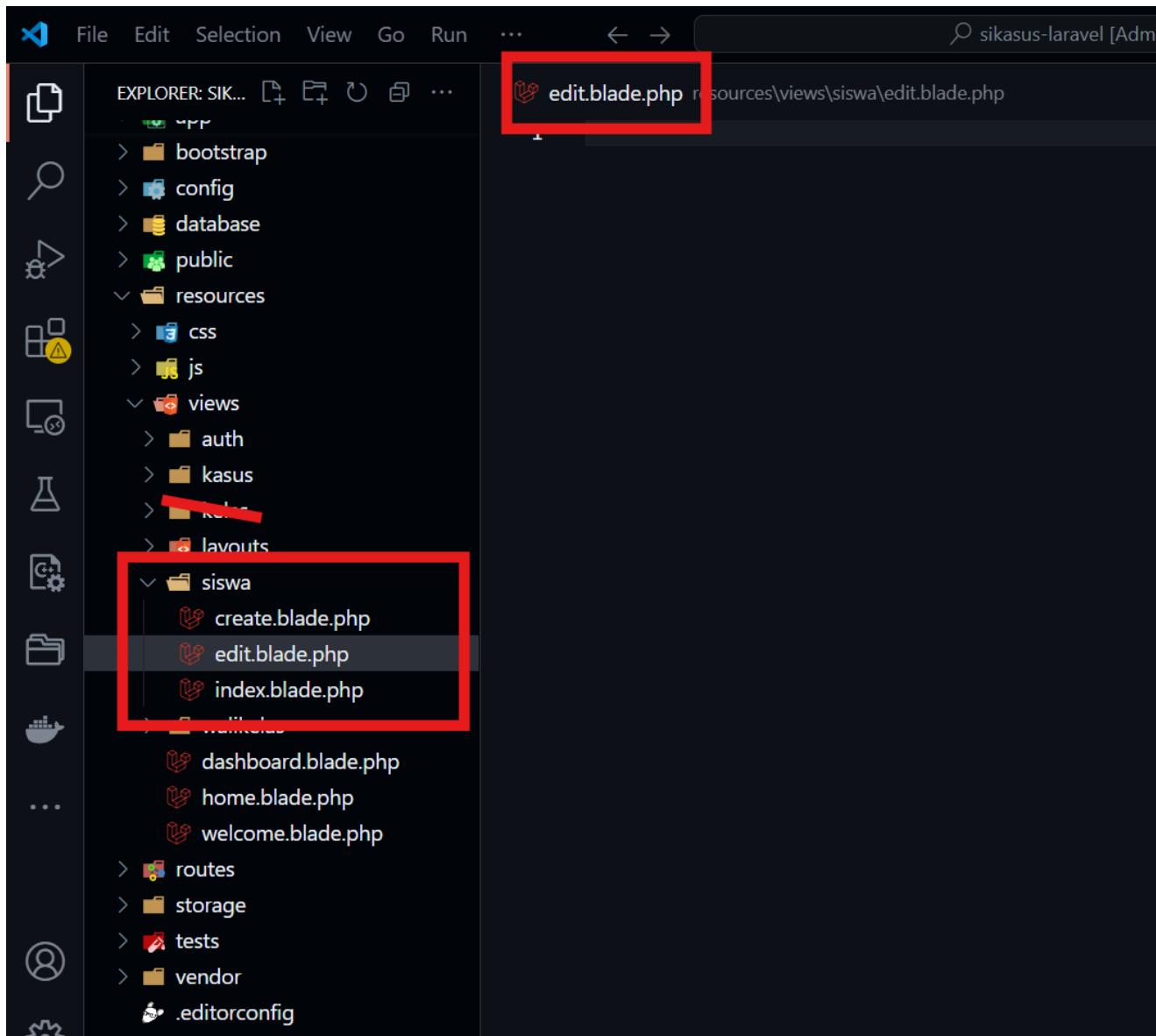
        <button type="submit" class="btn btn-primary mt-3">Simpan</button>
    </form>
</div>
@endsection

```

```

File Edit Selection View Go Run ... ← → ⌂ siskus-laravel [Administrator]
EXPLORER SIK... D+ ⌂ ⌂ ⌂ ...
    > app
    > bootstrap
    > config
    > database
    > public
    > resources
        > css
        > js
        > views
            > auth
            > kasus
            > layouts
                > siswa
                    > create.blade.php
                    > index.blade.php
                    > dashboard.blade.php
                    > home.blade.php
                    > welcome.blade.php
    > routes
    > storage
    > tests
    > vendor
    .editorconfig
create.blade.php resources\views\siswa\create.blade.php...
1  @extends('layouts.app')
2
3  @section('content')
4      <div class="container">
5          <h1>Tambah Siswa Baru</h1>
6
7          <form action="{{ route('siswa.store') }}" method="POST">
8              @csrf
9              <div class="form-group">
10                 <label for="nama_lengkap">Nama Lengkap</label>
11                 <input type="text" name="nama_lengkap" id="nama_lengkap" class="form-control" required>
12             </div>
13
14             <div class="form-group">
15                 <label for="nisn">NISN</label>
16                 <input type="text" name="nisn" id="nisn" class="form-control" required>
17             </div>
18
19             <div class="form-group">
20                 <label for="jenis_kelamin">Jenis Kelamin</label>
21                 <select name="jenis_kelamin" id="jenis_kelamin" class="form-control" required>
22                     <option value="Laki-Laki">Laki-Laki</option>
23                     <option value="Perempuan">Perempuan</option>
24                 </select>
25             </div>
26
27             <div class="form-group">
28                 <label for="tanggal_lahir">Tanggal Lahir</label>
29                 <input type="date" name="tanggal_lahir" id="tanggal_lahir" class="form-control" required>
30             </div>
31
32         <div class="form-group">
33             <button type="submit" class="btn btn-primary mt-3">Simpan</button>
34         </form>
35     </div>
36
37     @endsection

```



```
edit.blade.php resources\views\siswa\edit.blade.php
1 @extends('Layouts.app')
2
3 @section('content')
4     <div class="container">
5         <h1>Edit Data Siswa</h1>
6
7         <form action="{{ route('siswa.update', $siswa->id) }}" method="POST">
8             @csrf
9             @method('PUT')
10            <div class="form-group">
11                <label for="nama_lengkap">Nama Lengkap</label>
12                <input type="text" name="nama_lengkap" id="nama_lengkap" class="form-control" value="{{ $siswa->nama_lengkap }}" required>
13            </div>
14
15            <div class="form-group">
16                <label for="nisn">NISSN</label>
17                <input type="text" name="nisn" id="nisn" class="form-control" value="{{ $siswa->nism }}" required>
18            </div>
19
20            <div class="form-group">
21                <label for="jenis_kelamin">Jenis Kelamin</label>
22                <select name="jenis_kelamin" id="jenis_kelamin" class="form-control" required>
23                    <option value="Laki-Laki" {{ $siswa->jenis_kelamin == 'Laki-Laki' ? 'selected' : '' }}>Laki-Laki</option>
24                    <option value="Perempuan" {{ $siswa->jenis_kelamin == 'Perempuan' ? 'selected' : '' }}>Perempuan</option>
25                </select>
26            </div>
27
28            <div class="form-group">
29                <label for="tanggal_lahir">Tanggal Lahir</label>
30                <input type="date" name="tanggal_lahir" id="tanggal_lahir" class="form-control" value="{{ $siswa->tanggal_lahir }}" required>
31            </div>
32
33        <div class="form-group">
```

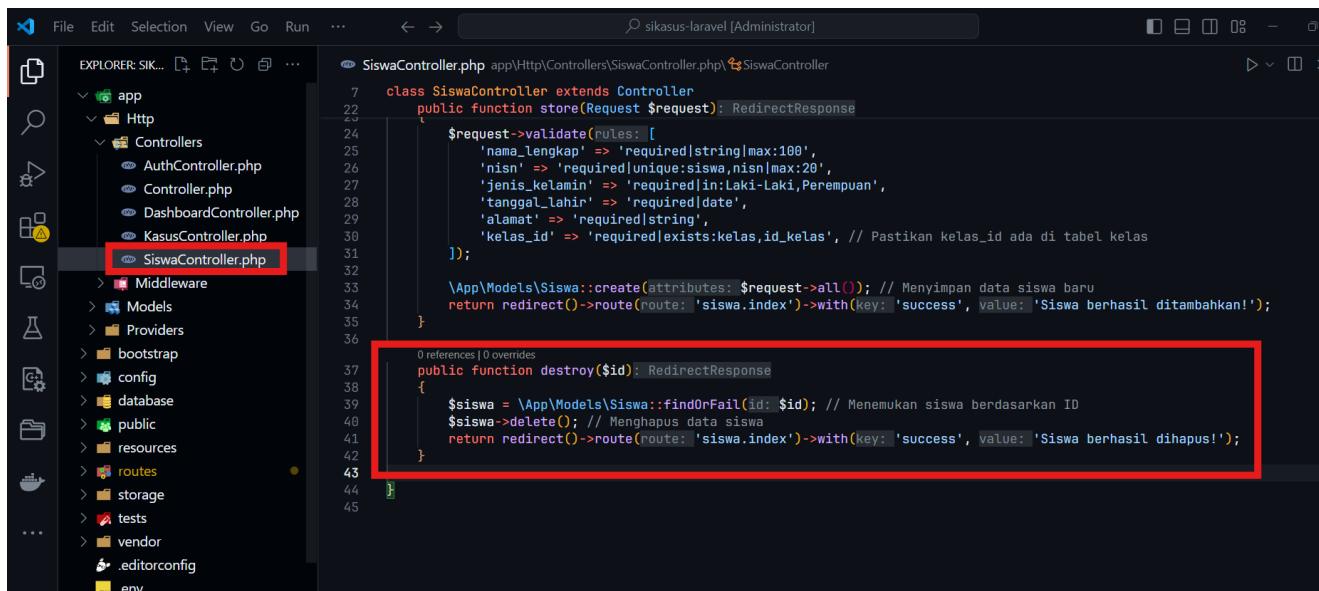
Menambahkan Method destroy di Controller

Untuk menghapus data siswa, kita perlu menambahkan method `destroy` di dalam controller **Siswa**. Method ini akan menangani permintaan untuk menghapus data siswa

berdasarkan ID yang diberikan.

Buka file `SiswaController.php`, kemudian tambahkan kode berikut untuk method `destroy`:

```
public function destroy($id)
{
    $siswa = \App\Models\Siswa::findOrFail($id); // Menemukan siswa berdasarkan ID
    $siswa->delete(); // Menghapus data siswa
    return redirect()->route('siswa.index')->with('success', 'Siswa berhasil dihapus!');
}
```



Menambahkan Tombol Hapus di View

Setelah menambahkan method `destroy` di controller, langkah selanjutnya adalah menambahkan tombol hapus di view yang menampilkan daftar siswa. Tombol ini akan memanggil action `destroy` di controller dan menghapus siswa yang dipilih.

Buka file `resources/views/siswa/index.blade.php` dan tambahkan form untuk menghapus data siswa, seperti berikut:

```
<form action="{{ route('siswa.destroy', $item->id) }}" method="POST"
style="display:inline-block;">
    @csrf
    @method('DELETE')
    <button type="submit" class="btn btn-danger btn-sm"
        onclick="return confirm('Yakin ingin menghapus?')>Hapus</button>
</form>
```

```

<div class="card mb-4">
    <div class="card-body">
        <table id="datatablesSimple">
            <tfoot>
                <tr>
                    <td>{{ $loop->iteration }}</td>
                    <td>{{ $item->nama_lengkap }}</td>
                    <td>{{ $item->nisn }}</td>
                    <td>{{ $item->jenis_kelamin }}</td>
                    <td>{{ $item->tanggal_lahir }}</td>
                    <td>{{ $item->alamat }}</td>
                    <td>
                        <a href="{{ route('siswa.edit', $item->id) }}" class="btn btn-warning btn-sm">EditHapusEditHapus

```

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration	1. Migration Kasus 2. Migration Siswa 3. Migration Kelas 4. Migration Walikelas	✓ ✓ ✓ ✓
Menambahkan Mass Assignment	1. Model Kasus 2. Model Siswa 3. Model Kelas 4. Model Walikelas	✓ ✓ ✓ ✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓

Topik	Sub Topik	Status
	Menampilkan Halaman Jenis Login	✓
	Menampilkan Form Login Berdasarkan Jenis	✓
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		✓
	Membuat Middleware untuk Autentikasi	✓
	Membuat Middleware untuk Role	✓
	Registrasi Middleware	✓
Membuat Blade Layout		✓
Membuat Tampilan Homepage		✓
Membuat Tampilan Dashboard Setelah Login		✓
Membuat Fitur Kasus		✓
	Menambahkan Routing untuk Kasus	✓
	Membuat Controller Kasus	✓
	Menampilkan Daftar Kasus (Method index)	✓
	Membuat View untuk Menampilkan Kasus (View index)	✓
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Kasus (View create)	✓
	Edit dan Update Kasus ke Database	✓
	Membuat View Form Edit Kasus (View edit)	✓
	Menambahkan Method Destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Siswa		✓
	Menambahkan Routing untuk Siswa	✓
	Membuat Controller Siswa	✓
	Menampilkan Daftar Siswa (Method index)	✓

Topik	Sub Topik	Status
	Membuat View untuk Menampilkan Siswa (View index)	<input checked="" type="checkbox"/>
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	<input checked="" type="checkbox"/>
	Membuat Form untuk Menambah Siswa (View create)	<input checked="" type="checkbox"/>
	Menambahkan Method destroy di Controller	<input checked="" type="checkbox"/>
	Menambahkan Tombol Hapus di View	<input checked="" type="checkbox"/>
Membuat Fitur Walikelas		
	Menambahkan Routing untuk Walikelas	
	Membuat Controller Walikelas	
	Menampilkan Daftar Walikelas (Method index)	
	Membuat View untuk Menampilkan Walikelas (View index)	
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Walikelas (View create)	
	Edit dan Update Walikelas ke Database (Method edit dan update)	
	Membuat View Form Edit Walikelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	

Topik	Sub Topik	Status
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk 'Siswa	
	Membuat Controller 'Siswa	

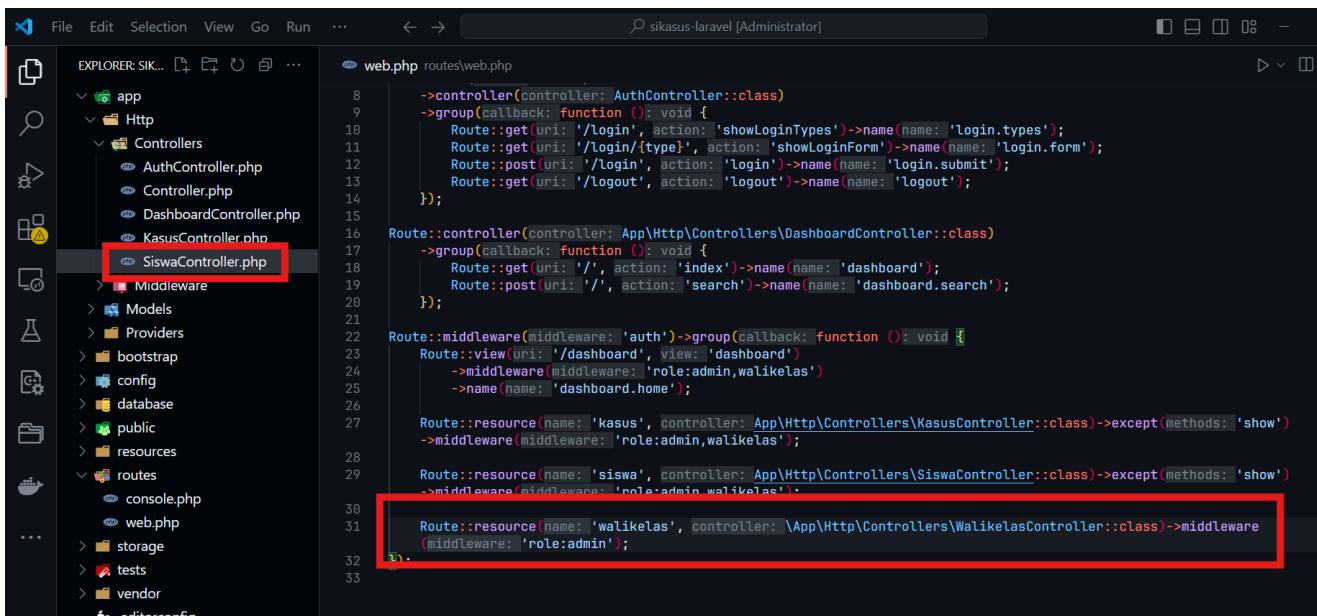
Membuat Fitur Walikelas

Menambahkan Routing untuk Walikelas

Langkah pertama dalam membuat bagian Kelas adalah menambahkan routing di file routes/web.php . Hal ini memungkinkan kita untuk menghubungkan URL dengan fungsi yang sesuai di dalam controller. Gunakan resource route untuk mengelola semua operasi CRUD di kelas.

Buka file `routes/web.php` dan tambahkan kode berikut:

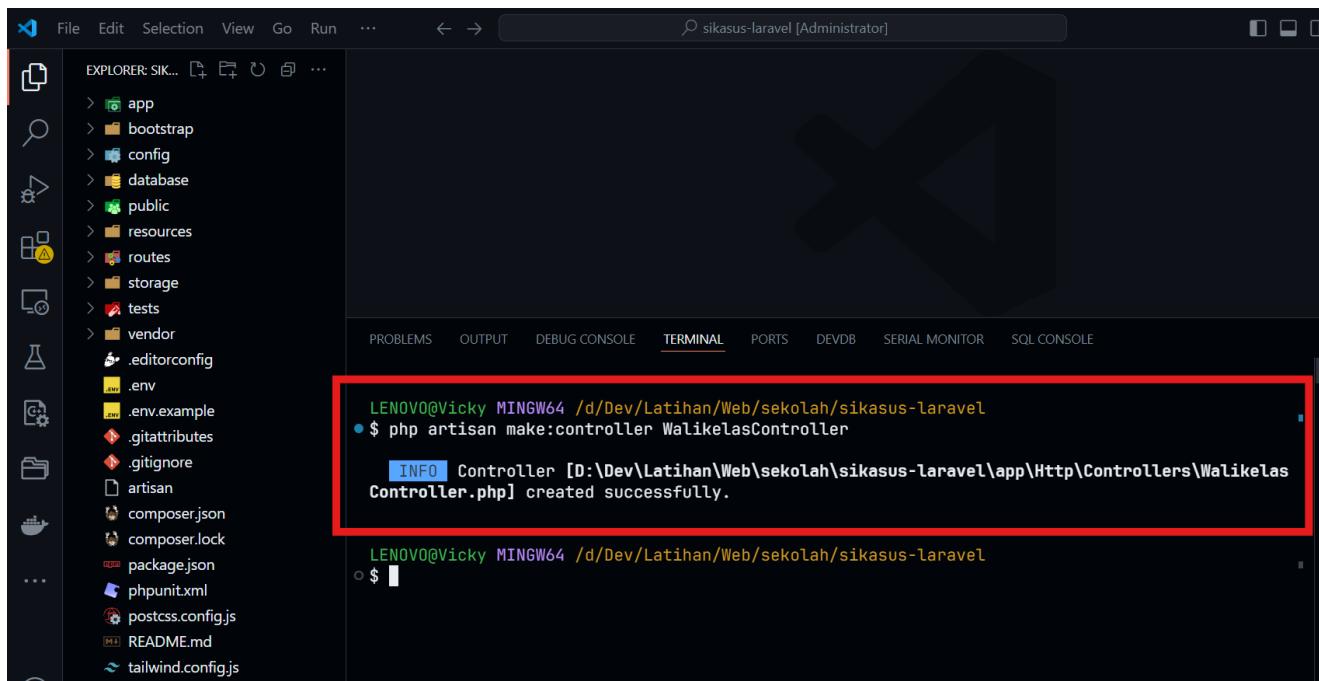
```
Route::resource('walikelas',  
    \App\Http\Controllers\WalikelasController::class)->middleware('role:admin');
```



Membuat Controller Walikelas

Selanjutnya, buat controller untuk menangani operasi CRUD terkait `Walikelas`. Gunakan perintah Artisan untuk membuat controller baru:

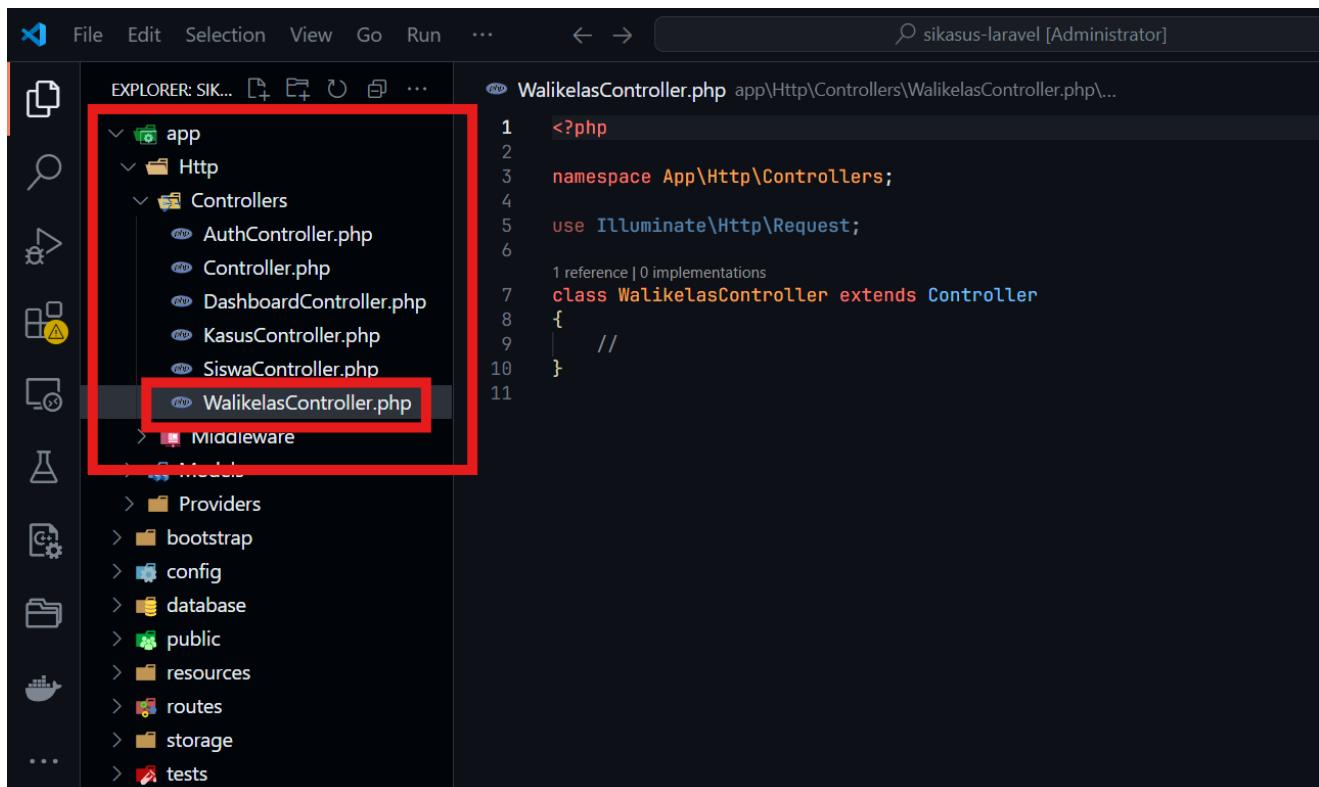
```
php artisan make:controller WalikelasController
```



```
LENovo@Vicky MINGW64 /d/Dev/Latihan/Web/sekolah/sikasus-laravel
$ php artisan make:controller WalikelasController

INFO Controller [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\app\Http\Controllers\WalikelasController.php] created successfully.
```

Setelah controller berhasil dibuat, buka file `WalikelasController.php` di `app/Http/Controllers` dan tambahkan method-method seperti `index` dsb.



```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class WalikelasController extends Controller
{
    //
}
```

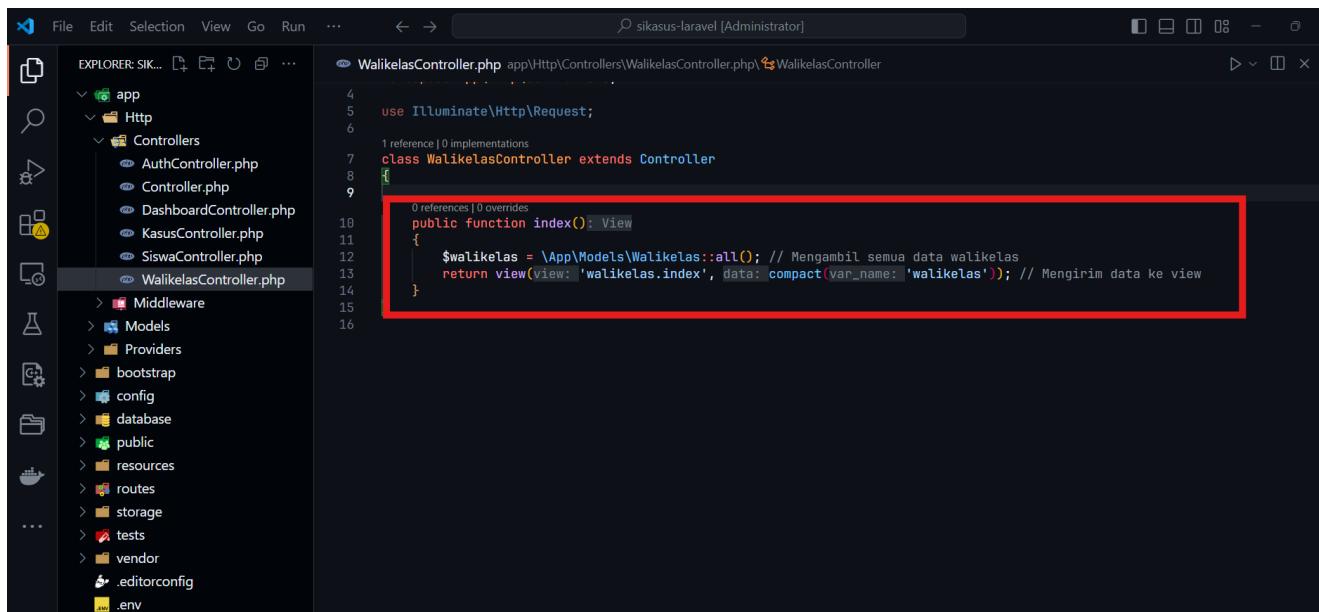
Menampilkan Daftar Walikelas (Method index)

Di dalam method `index`, kita akan mengambil semua data walikelas yang ada dalam database dan menampilkannya di view `walikelas.index`.

```

public function index()
{
    $walikelas = \App\Models\Walikelas::all(); // Mengambil semua data walikelas
    return view('walikelas.index', compact('walikelas')); // Mengirim data ke view
}

```



The screenshot shows a code editor interface with a dark theme. On the left is the file explorer sidebar showing the project structure. The main area displays the `WalikelasController.php` file. The `index()` method is highlighted with a red rectangular box.

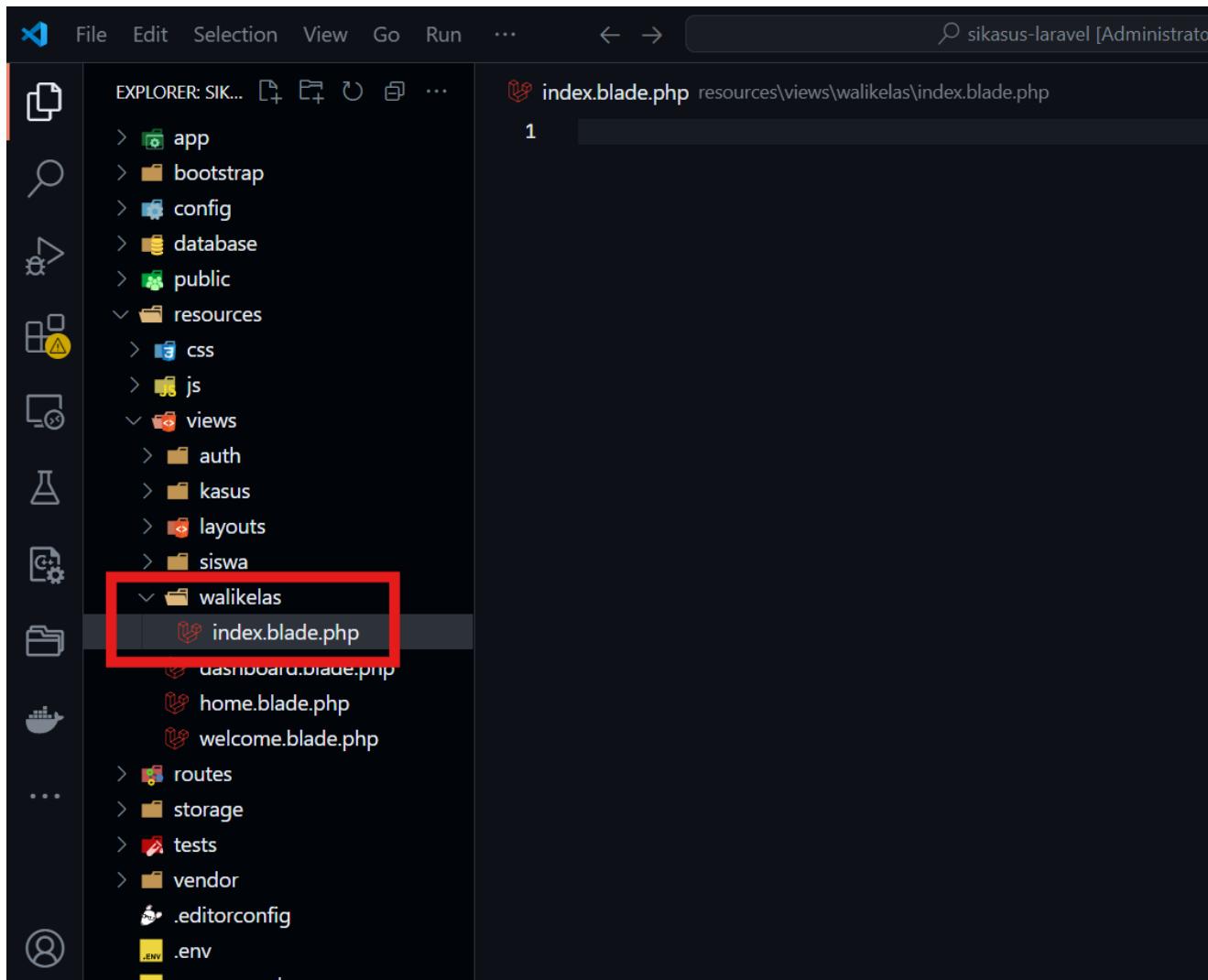
```

use Illuminate\Http\Request;
class WalikelasController extends Controller
{
    public function index(): View
    {
        $walikelas = \App\Models\Walikelas::all(); // Mengambil semua data walikelas
        return view('view: "walikelas.index", data: compact(var_name: "walikelas')); // Mengirim data ke view
    }
}

```

Membuat View untuk Menampilkan Walikelas (View index)

Buat view `index.blade.php` di dalam folder `resources/views/walikelas` yang menampilkan daftar wali kelas.



Kemudian, buat kode untuk menampilkan daftar walikelas :

```
@extends('layouts.app')

@section('title', 'Daftar Walikelas')

@section('content')
    <h1>Daftar Walikelas</h1>
    <a href="{{ route('walikelas.create') }}" class="my-2 btn btn-primary">Tambah Walikelas</a>

    <div class="card mb-4">
        <div class="card-body">
            <table id="datatablesSimple">
                <thead>
                    <tr>
                        <th>#</th>
                        <th>Nama</th>
                        <th>NIP</th>
                        <th>Jenis Kelamin</th>
                        <th>Alamat</th>
                        <th>Aksi</th>
                    </tr>
                </thead>
                <tbody>
                </tbody>
            </table>
        </div>
    </div>
</section>
```

```

        </thead>
        <tfoot>
            <tr>
                <th>#</th>
                <th>Nama</th>
                <th>NIP</th>
                <th>Jenis Kelamin</th>
                <th>Alamat</th>
                <th>Aksi</th>
            </tr>
        </tfoot>
        <tbody>
            @foreach ($walikelas as $item)
            <tr>
                <td>{{ $loop->iteration }}</td>
                <td>{{ $item->nama_walikelas }}</td>
                <td>{{ $item->nip }}</td>
                <td>{{ $item->jenis_kelamin }}</td>
                <td>{{ $item->alamat }}</td>
                <td>
                    <a href="{{ route('walikelas.edit', $item->id_walikelas) }}" class="btn btn-warning btn-sm">Edit</a>
                </td>
            </tr>
        @endforeach
    </tbody>
</table>
</div>
</div>

```

@endsection

```

EXPLORER: SIK... File Edit Selection View Go Run ...
index.blade.php resources\views\walikelas\index.blade.php
1  @extends('layouts.app')
2
3  @section('title', 'Daftar Walikelas')
4
5  @section('content')
6      <h1>Daftar Walikelas</h1>
7      <a href="{{ route('walikelas.create') }}" class="my-2 btn btn-primary">Tambah Walikelas</a>
8
9      <div class="card mb-4">
10         <div class="card-body">
11             <table id="datatablesSimple">
12                 <thead>
13                     <tr>
14                         <th>#</th>
15                         <th>Nama</th>
16                         <th>NIP</th>
17                         <th>Jenis Kelamin</th>
18                         <th>Alamat</th>
19                         <th>Aksi</th>
20                     </tr>
21                 </thead>
22                 <tfoot>
23                     <tr>
24                         <th>#</th>
25                         <th>Nama</th>
26                         <th>NIP</th>
27                         <th>Jenis Kelamin</th>
28                         <th>Alamat</th>
29                         <th>Aksi</th>
30                     </tr>
31                 </tfoot>
32             <tbody>
33                 @foreach ($walikelas as $item)
34                 <tr>
35                     <td>{{ $loop->iteration }}</td>

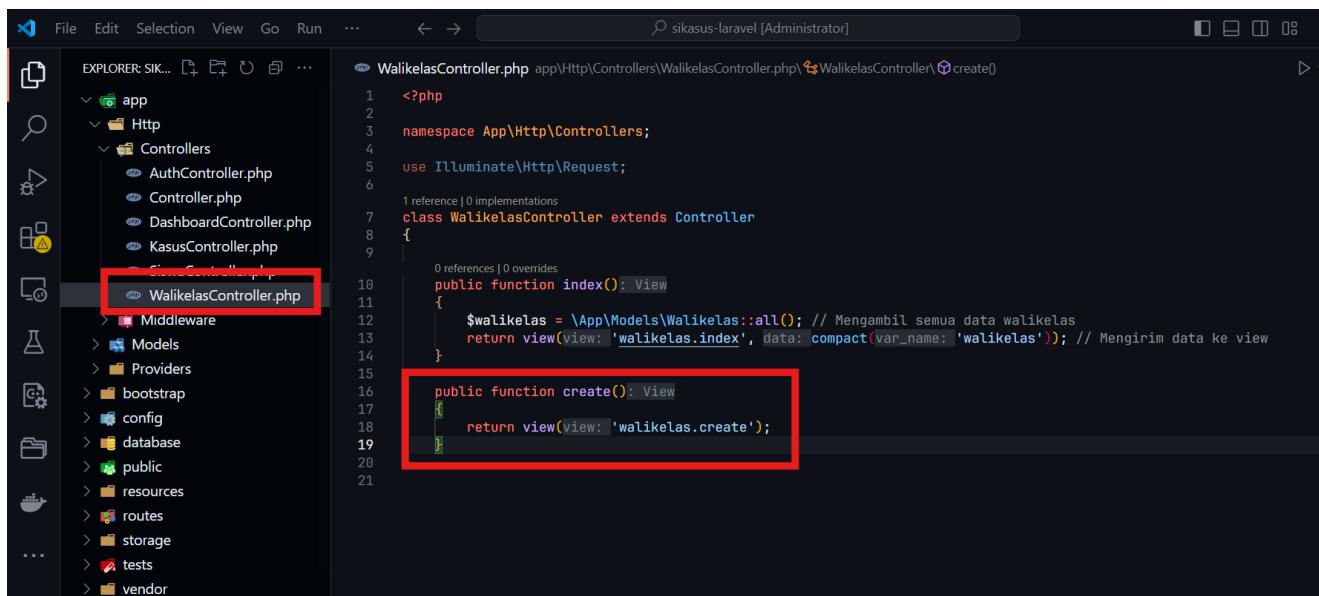
```

Menambahkan Wali Kelas Baru ke Dalam Database (Method create dan store)

Selanjutnya, kita akan menambahkan method `create` untuk menampilkan form tambah wali kelas, dan `store` untuk menyimpan data wali kelas baru ke database.

Di dalam `WalikelasController`, tambahkan method `create` untuk menampilkan form tambah wali kelas:

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

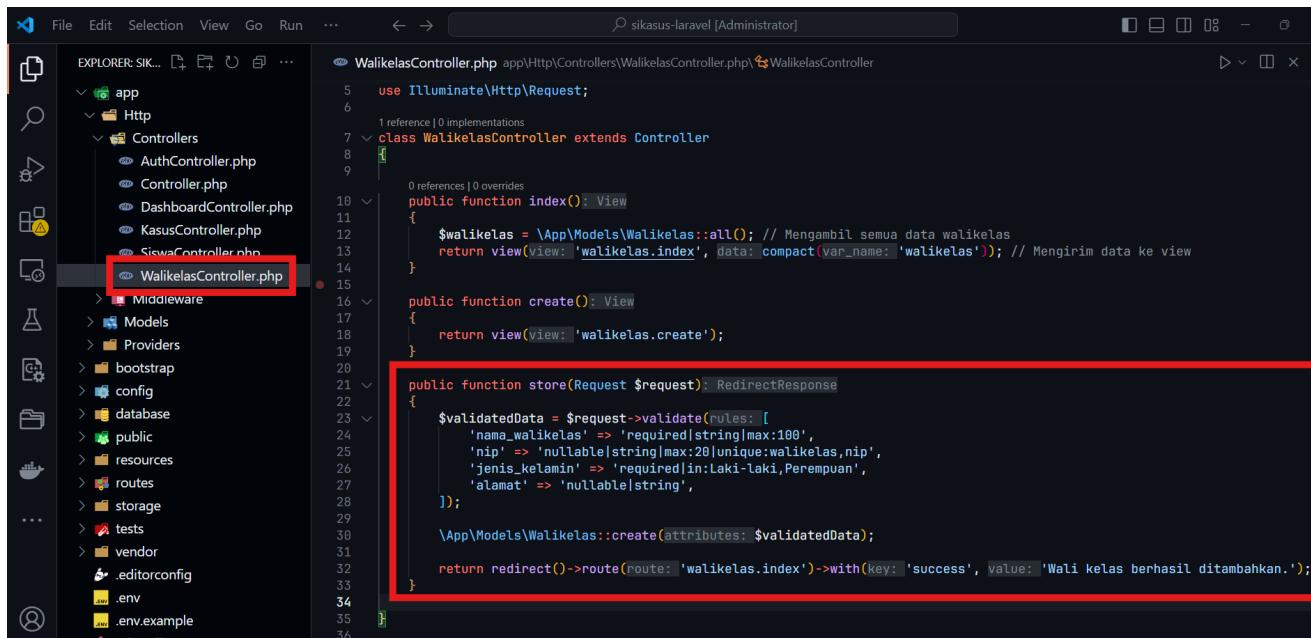


```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class WalikelasController extends Controller
{
    public function index(): View
    {
        $walikelas = \App\Models\Walikelas::all(); // Mengambil semua data walikelas
        return view('walikelas.index', compact('walikelas')); // Mengirim data ke view
    }
    public function create(): View
    {
        return view('walikelas.create');
    }
}
```

Kemudian, tambahkan method `store` untuk menyimpan data wali kelas yang dikirimkan melalui form:

```
public function store(Request $request)
{
    $validatedData = $request->validate([
        'nama_walikelas' => 'required|string|max:100',
        'nip' => 'nullable|string|max:20|unique:walikelas,nip',
        'jenis_kelamin' => 'required|in:Laki-laki,Perempuan',
        'alamat' => 'nullable|string',
    ]);
    \App\Models\Walikelas::create($validatedData);

    return redirect()->route('walikelas.index')->with('success', 'Wali kelas berhasil ditambahkan.');
}
```



```
use Illuminate\Http\Request;
class WalikelasController extends Controller
{
    public function index(): View
    {
        $walikelas = \App\Models\Walikelas::all(); // Mengambil semua data walikelas
        return view('view: "walikelas.index", data: compact(var_name: "walikelas')) // Mengirim data ke view
    }

    public function create(): View
    {
        return view('view: "walikelas.create");
    }

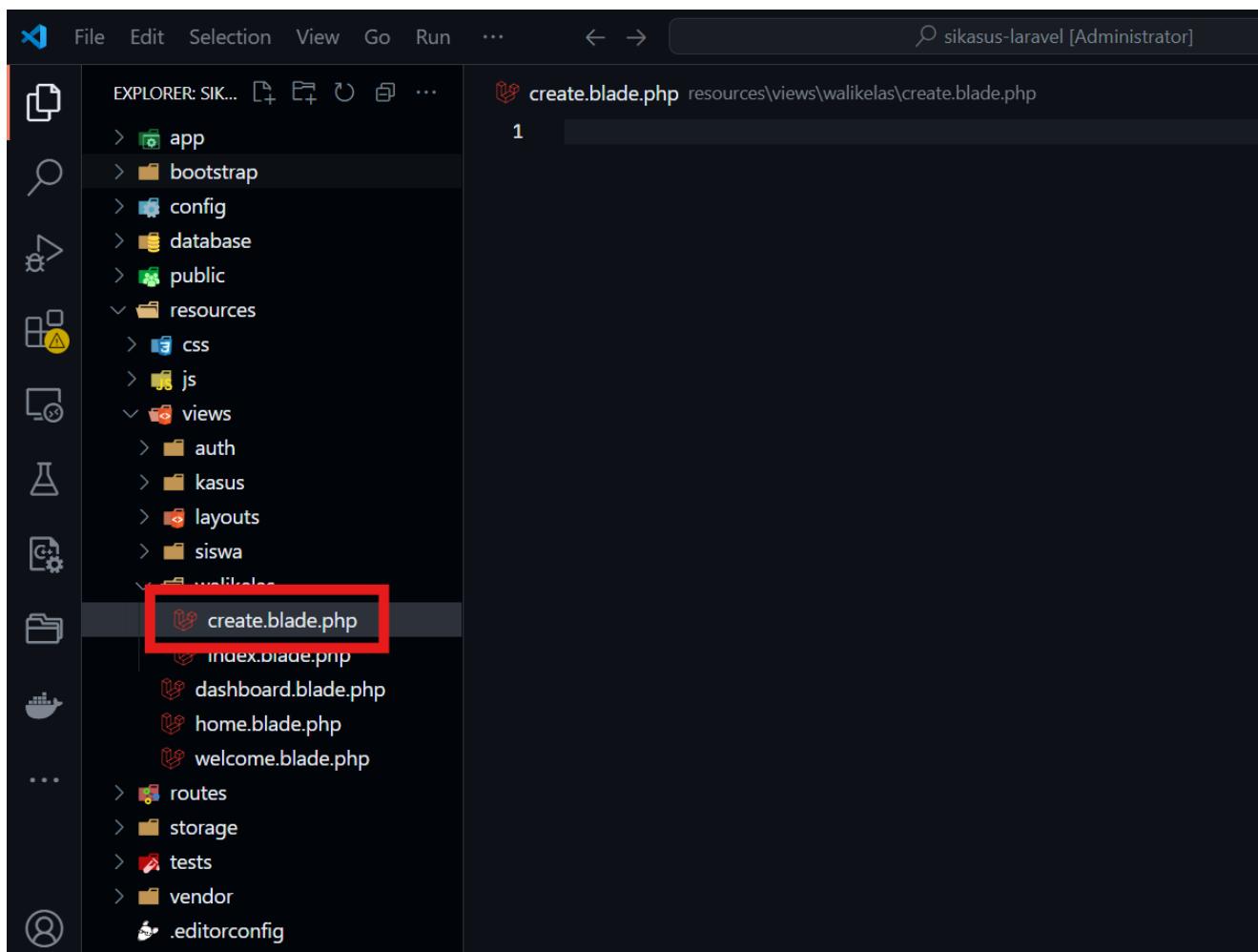
    public function store(Request $request): RedirectResponse
    {
        $validatedData = $request->validate(rules: [
            'nama_walikelas' => 'required|string|max:100',
            'nip' => 'nullable|string|max:20|unique:walikelas,nip',
            'jenis_kelamin' => 'required|in:Laki-laki,Perempuan',
            'alamat' => 'nullable|string',
        ]);

        \App\Models\Walikelas::create(attributes: $validatedData);

        return redirect()->route(route: "walikelas.index")->with(key: 'success', value: 'Wali kelas berhasil ditambahkan.');
    }
}
```

Membuat Form untuk Menambah Walikelas (View create)

Buat view untuk form tambah wali kelas. File ini akan berada di `resources/views/walikelas/create.blade.php`.



```
create.blade.php resources\views\walikelas\create.blade.php
```

Kemudian, buat kode untuk form tambah walikelas :

```
@extends('layouts.app')

@section('content')
    <div class="container">
        <h1>Tambah Wali Kelas Baru</h1>

        <form action="{{ route('walikelas.store') }}" method="POST">
            @csrf
            <div class="form-group">
                <label for="nama_walikelas">Nama Wali Kelas</label>
                <input type="text" name="nama_walikelas" id="nama_walikelas"
class="form-control" required>
            </div>

            <div class="form-group">
                <label for="nip">NIP</label>
                <input type="text" name="nip" id="nip" class="form-control"
required>
            </div>

            <div class="form-group">
                <label for="jenis_kelamin">Jenis Kelamin</label>
                <select name="jenis_kelamin" id="jenis_kelamin" class="form-
control" required>
                    <option value="Laki-laki">Laki-laki</option>
                    <option value="Perempuan">Perempuan</option>
                </select>
            </div>

            <div class="form-group">
                <label for="alamat">Alamat</label>
                <textarea name="alamat" id="alamat" class="form-control"
rows="3"></textarea>
            </div>

            <button type="submit" class="btn btn-primary mt-
3">Simpan</button>
        </form>
    </div>
@endsection
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows the project structure under "resources\views\walikelas". The "create.blade.php" file is currently selected.
- Code Editor (Right):** Displays the content of "create.blade.php". The code is a Blade template for creating a new student record. It includes sections for form fields like name, NIP, gender, address, and a submit button.

```
resources\views\walikelas\create.blade.php

1  @extends('layouts.app')
2
3  @section('content')
4      <div class="container">
5          <h1>Tambah Wali Kelas Baru</h1>
6
7          <form action="{{ route('walikelas.store') }}" method="POST">
8              @csrf
9              <div class="form-group">
10                  <label for="nama_walikelas">Nama Wali Kelas</label>
11                  <input type="text" name="nama_walikelas" id="nama_walikelas" class="form-control" required>
12              </div>
13
14              <div class="form-group">
15                  <label for="nip">NIP</label>
16                  <input type="text" name="nip" id="nip" class="form-control" required>
17              </div>
18
19              <div class="form-group">
20                  <label for="jenis_kelamin">Jenis Kelamin</label>
21                  <select name="jenis_kelamin" id="jenis_kelamin" class="form-control" required>
22                      <option value="Laki-laki">Laki-laki</option>
23                      <option value="Perempuan">Perempuan</option>
24                  </select>
25              </div>
26
27              <div class="form-group">
28                  <label for="alamat">Alamat</label>
29                  <textarea name="alamat" id="alamat" class="form-control" rows="3"></textarea>
30              </div>
31
32              <button type="submit" class="btn btn-primary mt-3">Simpan</button>
33          </form>
34      </div>
35  @endsection
```

Edit dan Update Walikelas ke Database (Method edit dan update)

Untuk melakukan edit data wali kelas, kita perlu membuat method `edit` dan `update`.

Method `edit` digunakan untuk menampilkan form edit yang sudah terisi dengan data wali kelas, sedangkan method `update` digunakan untuk menyimpan perubahan data ke dalam database.

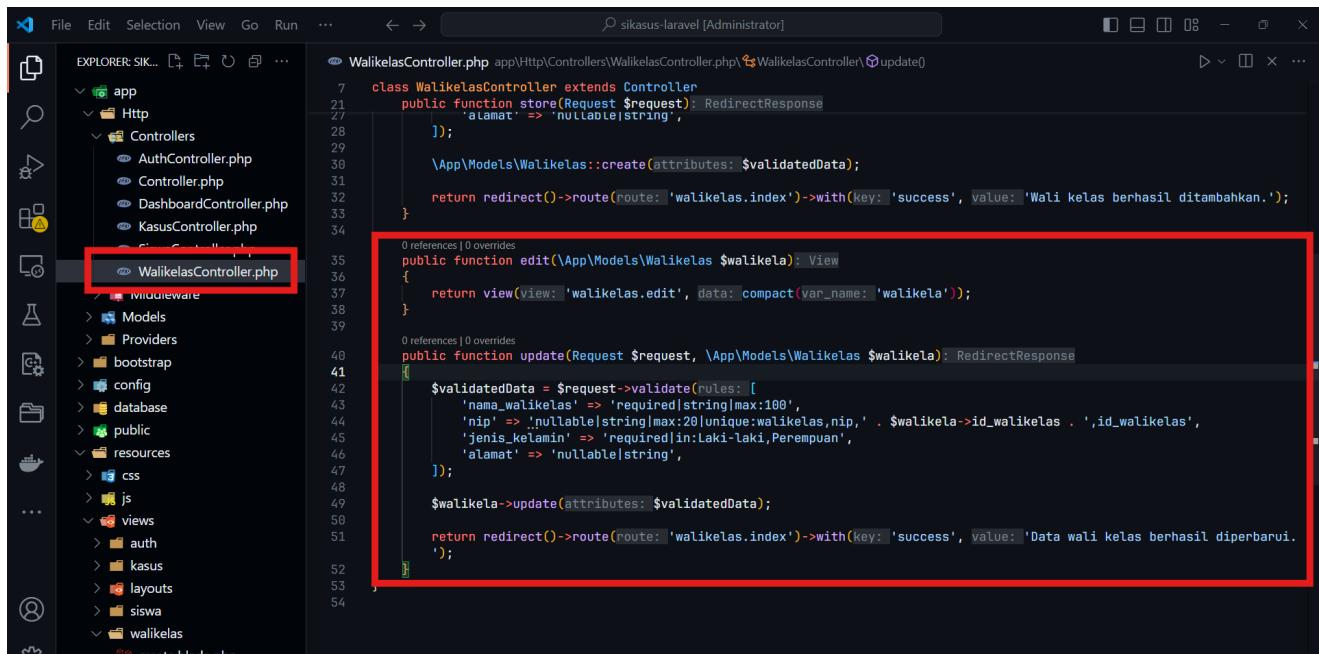
Tambahkan method edit dan update pada WalikelasController:

```
public function edit(\App\Models\Walikelas $walikelas)
{
    return view('walikelas.edit', compact('walikelas'));
}

public function update(Request $request, \App\Models\Walikelas $walikelas)
{
    $validatedData = $request->validate([
        'nama_walikelas' => 'required|string|max:100',
        'nip' => 'nullable|string|max:20|unique:walikelas,nip,' . $walikelas->id_walikelas . ',id_walikelas',
        'jenis_kelamin' => 'required|in:Laki-laki,Perempuan',
        'alamat' => 'nullable|string',
    ]);

    $walikelas->update($validatedData);

    return redirect()->route('walikelas.index')->with('success', 'Data wali
kelas berhasil diperbarui.');
}
```



```
class WalikelasController extends Controller
{
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'alamat' => 'nullable|string',
        ]);

        \App\Models\Walikelas::create($request->all());

        return redirect()->route('walikelas.index')->with(['key' => 'success', 'value' => 'Wali kelas berhasil ditambahkan.']);
    }

    public function edit(\App\Models\Walikelas $walikelas): View
    {
        return view('walikelas.edit', compact('walikelas'));
    }

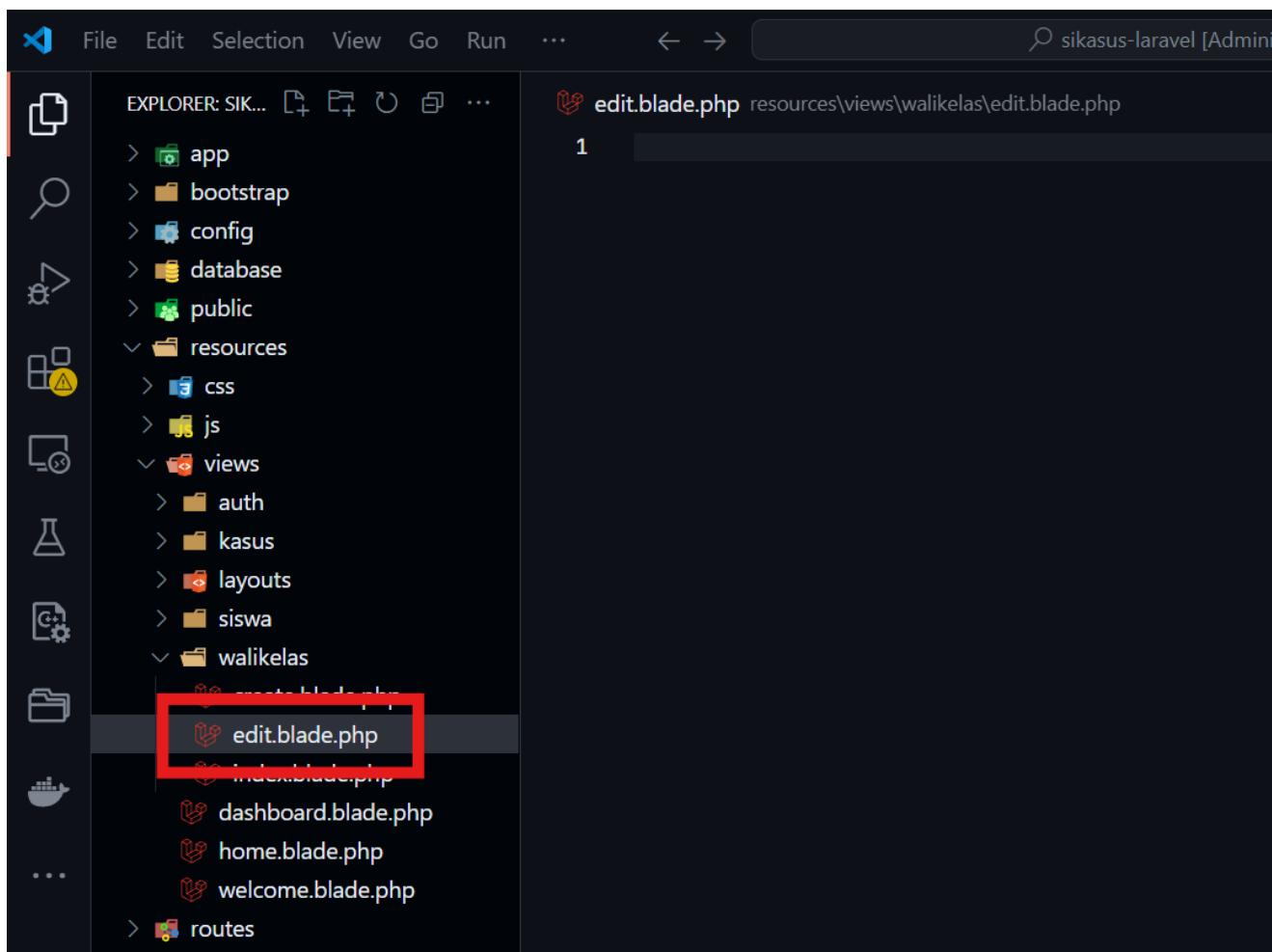
    public function update(Request $request, \App\Models\Walikelas $walikelas): RedirectResponse
    {
        $validatedData = $request->validate([
            'nama_walikelas' => 'required|string|max:100',
            'nip' => 'nullable|string|max:20|unique:walikelas,nip,' . $walikelas->id_walikelas . ',id_walikelas',
            'jenis_kelamin' => 'required|in:Laki-laki,Perempuan',
            'alamat' => 'nullable|string',
        ]);

        $walikelas->update($validatedData);

        return redirect()->route('walikelas.index')->with(['key' => 'success', 'value' => 'Data wali kelas berhasil diperbarui.']);
    }
}
```

Membuat View Form Edit Wali kelas (View edit)

Sekarang, buat view untuk form edit wali kelas di `resources/views/walikelas/edit.blade.php`.



```
1
```

Kemudian, membuat kode form edit walikelas :

```

@extends('layouts.app')

@section('content')
    <div class="container">
        <h1>Edit Data Wali Kelas</h1>

        <form action="{{ route('walikelas.update', $walikelala->id_walikelas) }}" method="POST">
            @csrf
            @method('PUT')
            <div class="form-group">
                <label for="nama_walikelas">Nama Wali Kelas</label>
                <input type="text" name="nama_walikelas" id="nama_walikelas" class="form-control" value="{{ $walikelala->nama_walikelas }}" required>
            </div>
            <div class="form-group">
                <label for="nip">NIP</label>
                <input type="text" name="nip" id="nip" class="form-control" value="{{ $walikelala->nip }}" required>
            </div>

            <div class="form-group">
                <label for="jenis_kelamin">Jenis Kelamin</label>
                <select name="jenis_kelamin" id="jenis_kelamin" class="form-control" required>
            <div class="form-group">
                <label for="alamat">Alamat</label>
                <textarea name="alamat" id="alamat" class="form-control" rows="3">{{ $walikelala->alamat }}</textarea>
            </div>

            <button type="submit" class="btn btn-warning mt-3">Perbarui</button>
        </form>
    </div>

@endsection

```

```

1 @extends('layouts.app')
2
3 @section('content')
4     <div class="container">
5         <h1>Edit Data Wali Kelas</h1>
6
7         <form action="{{ route('walikelas.update', $walikela->id_walikelas) }}" method="POST">
8             @csrf
9             @method('PUT')
10            <div class="form-group">
11                <label for="nama_walikelas">Nama Wali Kelas</label>
12                <input type="text" name="nama_walikelas" id="nama_walikelas" class="form-control" value="{{ $walikela->nama_walikelas }}" required>
13            </div>
14            <div class="form-group">
15                <label for="nip">NIP</label>
16                <input type="text" name="nip" id="nip" class="form-control" value="{{ $walikela->nip }}" required>
17            </div>
18
19            <div class="form-group">
20                <label for="jenis_kelamin">Jenis Kelamin</label>
21                <select name="jenis_kelamin" id="jenis_kelamin" class="form-control" required>
22
23                <div class="form-group">
24                    <label for="alamat">Alamat</label>
25                    <textarea name="alamat" id="alamat" class="form-control" rows="3">{{ $walikela->alamat }}</textarea>
26                </div>
27
28            <button type="submit" class="btn btn-warning mt-3">Perbarui</button>
29        </form>
30    </div>
31
32 @endsection
33

```

Menambahkan Method destroy di Controller

Untuk menghapus data wali kelas, kita memerlukan method `destroy` di controller. Method ini akan menghapus data wali kelas dari database.

```

public function destroy(\App\Models\Walikelas $walikela)
{
    $walikela->delete(); // Menghapus data wali kelas
    return redirect()->route('walikelas.index')->with('success', 'Wali kelas berhasil dihapus.');
}

```

```

class WalikelasController extends Controller
{
    public function edit(\App\Models\Walikelas $walikela): View
    {
        return view('walikelas.edit', data: compact(var_name: 'walikela'));
    }

    public function update(Request $request, \App\Models\Walikelas $walikela): RedirectResponse
    {
        $validatedData = $request->validate(rules: [
            'nama_walikelas' => 'required|string|max:100',
            'nip' => 'nullable|string|max:20|unique:walikelas,nip,' . $walikela->id_walikelas . ',id_walikelas',
            'jenis_kelamin' => 'required|in:Laki-laki,Perempuan',
            'alamat' => 'nullable|string',
        ]);

        $walikela->update(attributes: $validatedData);

        return redirect()->route('walikelas.index')->with(key: 'success', value: 'Data wali kelas berhasil diperbarui');
    }

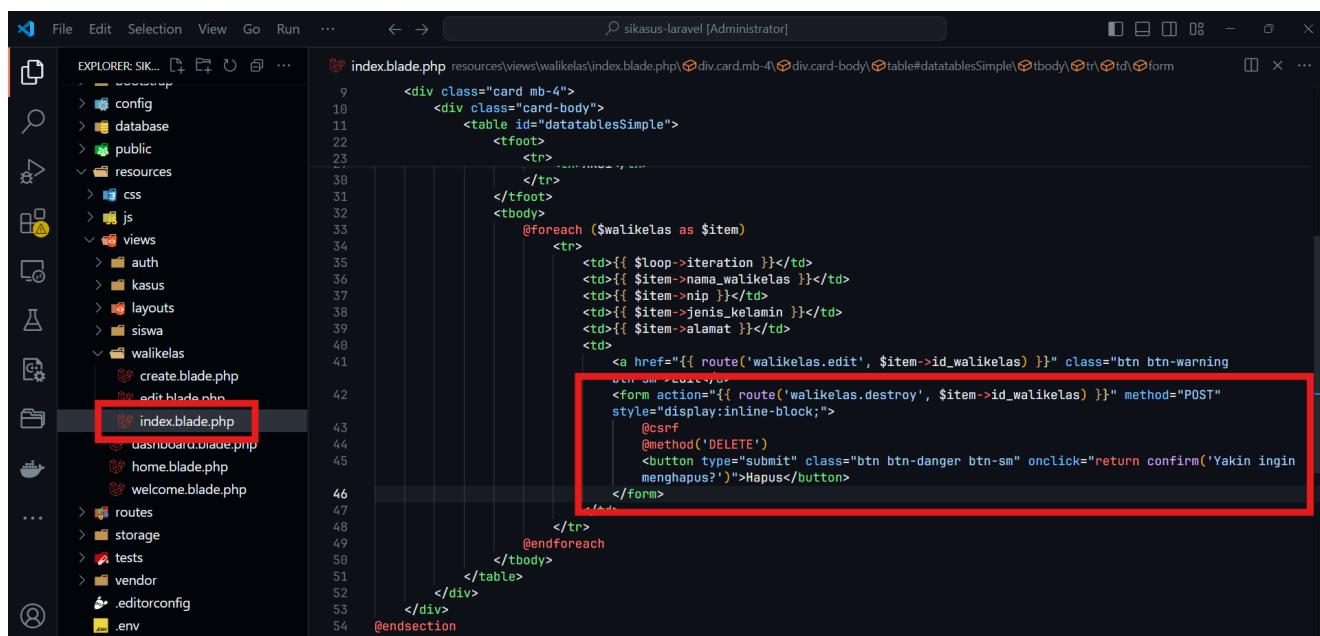
    public function destroy(\App\Models\Walikelas $walikela): RedirectResponse
    {
        $walikela->delete(); // Menghapus data wali kelas
        return redirect()->route('walikelas.index')->with(key: 'success', value: 'Wali kelas berhasil dihapus.');
    }
}

```

Menambahkan Tombol Hapus di View

Tombol hapus sudah ditambahkan di view `walikelas.index.blade.php`, namun pastikan form penghapusan sudah menggunakan `method="DELETE"` dan csrf token agar mengirimkan request yang aman.

```
<form action="{{ route('walikelas.destroy', $item->id_walikelas) }}" method="POST" style="display:inline-block;">
    @csrf
    @method('DELETE')
    <button type="submit" class="btn btn-danger btn-sm" onclick="return confirm('Yakin ingin menghapus?')">Hapus</button>
</form>
```



The screenshot shows a code editor interface with the file `index.blade.php` open. The file contains a table with data and a delete button for each row. The delete button's form is highlighted with a red box. The code within the highlighted box is as follows:

```
<form action="{{ route('walikelas.destroy', $item->id_walikelas) }}" method="POST" style="display:inline-block;">
    @csrf
    @method('DELETE')
    <button type="submit" class="btn btn-danger btn-sm" onclick="return confirm('Yakin ingin menghapus?')">Hapus</button>
</form>
```

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration	1. Migration Kasus 2. Migration Siswa 3. Migration Kelas 4. Migration Walikelas	✓ ✓ ✓ ✓
Menambahkan Mass Assignment	1. Model Kasus 2. Model Siswa	✓ ✓

Topik	Sub Topik	Status
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓
	Menampilkan Halaman Jenis Login	✓
	Menampilkan Form Login Berdasarkan Jenis	✓
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		✓
	Membuat Middleware untuk Autentikasi	✓
	Membuat Middleware untuk Role	✓
	Registrasi Middleware	✓
Membuat Blade Layout		✓
Membuat Tampilan Homepage		✓
Membuat Tampilan Dashboard Setelah Login		✓
Membuat Fitur Kasus		✓
	Menambahkan Routing untuk Kasus	✓
	Membuat Controller Kasus	✓
	Menampilkan Daftar Kasus (Method index)	✓
	Membuat View untuk Menampilkan Kasus (View index)	✓
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Kasus (View create)	✓
	Edit dan Update Kasus ke Database	✓
	Membuat View Form Edit Kasus (View edit)	✓

Topik	Sub Topik	Status
	Menambahkan Method Destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Siswa		✓
	Menambahkan Routing untuk Siswa	✓
	Membuat Controller Siswa	✓
	Menampilkan Daftar Siswa (Method index)	✓
	Membuat View untuk Menampilkan Siswa (View index)	✓
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Siswa (View create)	✓
	Menambahkan Method destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Walikelas		✓
	Menambahkan Routing untuk Walikelas	✓
	Membuat Controller Walikelas	✓
	Menampilkan Daftar Walikelas (Method index)	✓
	Membuat View untuk Menampilkan Walikelas (View index)	✓
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Walikelas (View create)	✓
	Edit dan Update Walikelas ke Database (Method edit dan update)	✓
	Membuat View Form Edit Walikelas (View edit)	✓
	Menambahkan Method destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Kelas		
	Menambahkan Routing untuk Kelas	
	Membuat Controller Kelas	
	Menampilkan Daftar Kelas (Method index)	
	Membuat View untuk Menampilkan Kelas (View index)	

Topik	Sub Topik	Status
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	
	Membuat Form untuk Menambah Kelas (View create)	
	Edit dan Update Kelas ke Database	
	Membuat View Form Edit Kelas (View edit)	
	Menambahkan Method destroy di Controller	
	Menambahkan Tombol Hapus di View	
Membuat Dashboard untuk Siswa		
	Menambahkan Routing untuk 'Siswa'	
	Membuat Controller 'Siswa'	

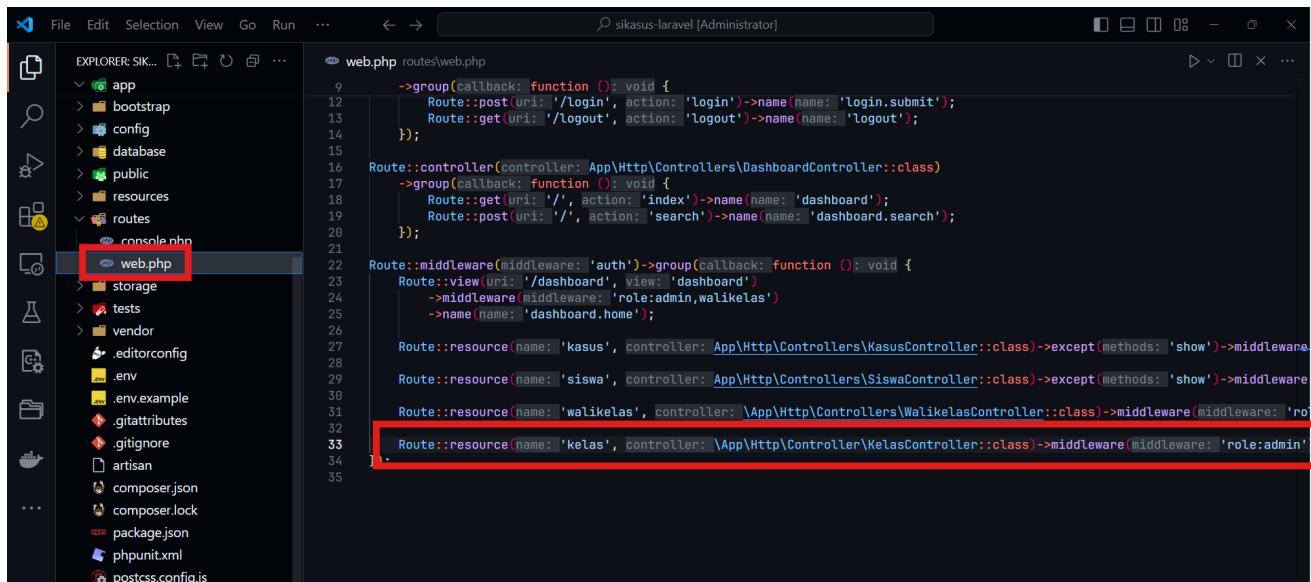
Membuat Fitur Kelas

Menambahkan Routing untuk Kelas

Langkah pertama dalam membuat bagian Kelas adalah menambahkan routing di file `routes/web.php`. Hal ini memungkinkan kita untuk menghubungkan URL dengan fungsi yang sesuai di dalam controller. Gunakan resource route untuk mengelola semua operasi CRUD di kelas.

Tambahkan kode berikut ke dalam file `routes/web.php`:

```
Route::resource('kelas', \App\Http\Controllers\KelasController::class)-
>middleware('role:admin');
};
```



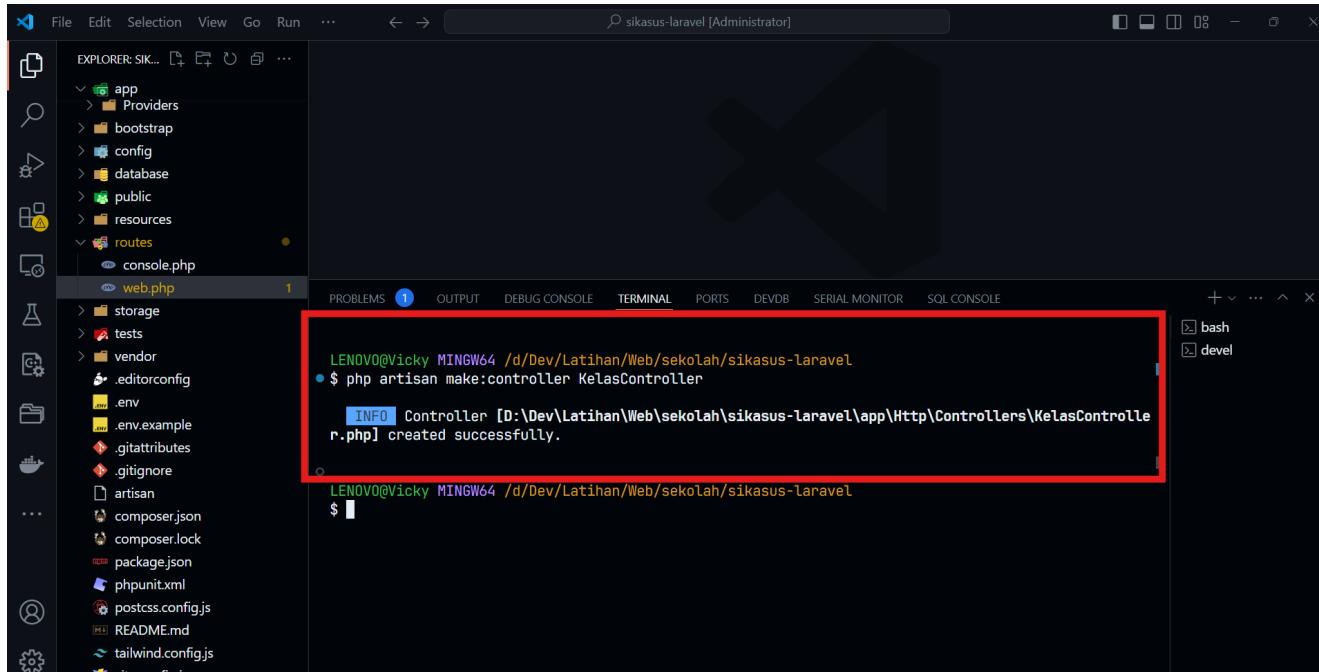
```
9      ->group(callback: function (): void {
10          Route::post(Uri: '/login', action: 'login')->name(name: 'login.submit');
11          Route::get(Uri: '/Logout', action: 'logout')->name(name: 'logout');
12      );
13
14      Route::controller(Controller: App\Http\Controllers\DashboardController::class)
15          ->group(callback: function (): void {
16              Route::get(Uri: '/', action: 'index')->name(name: 'dashboard');
17              Route::post(Uri: '/', action: 'search')->name(name: 'dashboard.search');
18          );
19
20      Route::middleware(middleware: 'auth')->group(callback: function (): void {
21          Route::view(Uri: '/dashboard', view: 'dashboard')
22              ->middleware(middleware: 'role:admin,walikelas')
23              ->name(name: 'dashboard.home');
24
25          Route::resource(name: 'kasus', controller: App\Http\Controllers\KasusController::class)->except(methods: 'show')->middleware(
26              middleware: 'role:admin'
27          );
28
29          Route::resource(name: 'siswa', controller: App\Http\Controllers\SiswaController::class)->except(methods: 'show')->middleware(
30              middleware: 'role:admin'
31          );
32
33          Route::resource(name: 'walikelas', controller: \App\Http\Controllers\WalikelasController::class)->middleware(middleware: 'role:admin')
34
35      );
36  
```

Membuat Controller Kelas

Langkah berikutnya adalah membuat controller Kelas . Kita akan menggunakan Artisan untuk membuat controller ini, yang akan menangani operasi CRUD untuk data kelas.

Gunakan perintah Artisan berikut:

```
php artisan make:controller KelasController
```



```
LENovo@Vicky MINGW64 /d/Dev/Latihan/Web/sekolah/sikasus-laravel
$ php artisan make:controller KelasController
[INFO] Controller [D:\Dev\Latihan\Web\sekolah\sikasus-laravel\app\Http\Controllers\KelasController.php] created successfully.
```

Setelah controller dibuat, buka file KelasController.php di folder app/Http/Controllers

```
File Edit Selection View Go Run ... ← → ⌘ sikasus-laravel [Administrator]
EXPLORER: SIK... + ⌂ ⌂ ⌂ ...
app
  Http
    Controllers
      AuthController.php
      Controller.php
      DashboardController.php
      KasusController.php
      KelasController.php
      SiswaController.php
      WalikelasController.php
    Middleware
  Models
  Providers
  bootstrap
  config
  database
  public
  resources
  routes
  storage
  tests
  vendor
```

```
KelasController.php app\Http\Controllers\KelasController.php...
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 1 reference | 0 implementations
8 class KelasController extends Controller
9 {
10     // ...
11 }
```

Kemudian, tambahkan method `index` untuk menampilkan tampilan data kelas.

```
public function index()
{
    $kelas = \App\Models\Kelas::with('walikelas')->get(); // Mengambil semua
    data kelas beserta wali kelasnya
    return view('kelas.index', compact('kelas'));
}
```

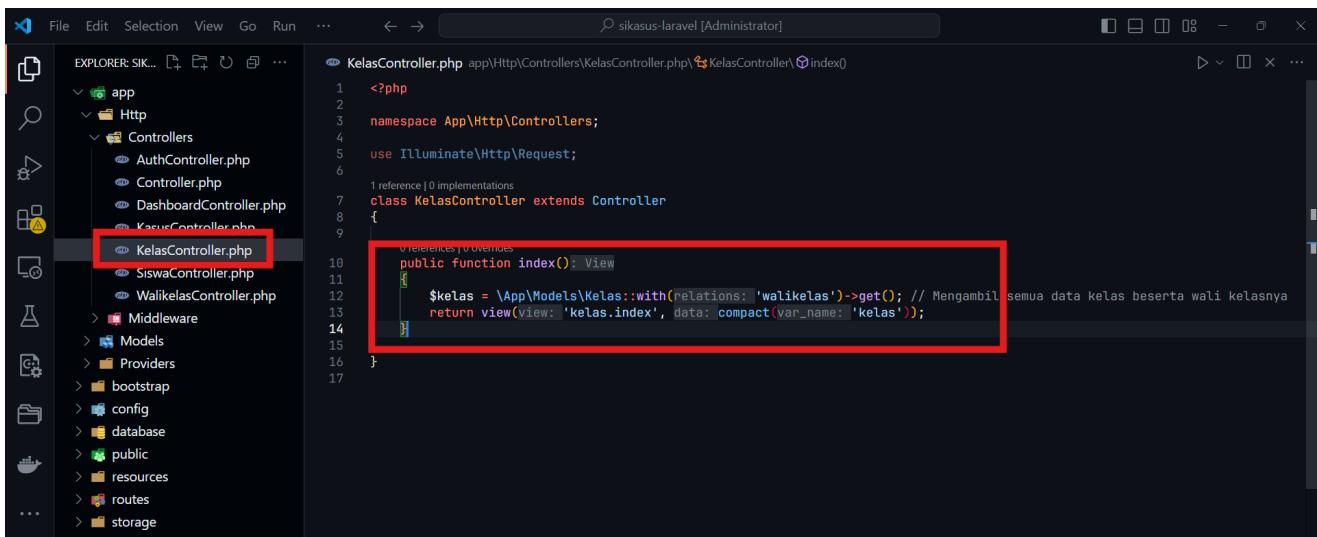
```
File Edit Selection View Go Run ... ← → ⌘ sikasus-laravel [Administrator]
EXPLORER: SIK... + ⌂ ⌂ ⌂ ...
app
  Http
    Controllers
      AuthController.php
      Controller.php
      DashboardController.php
      KasusController.php
      KelasController.php
      SiswaController.php
      WalikelasController.php
    Middleware
  Models
  Providers
  bootstrap
  config
  database
  public
  resources
  routes
  storage
  tests
  vendor
```

```
KelasController.php app\Http\Controllers\KelasController.php\index()
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 1 reference | 0 implementations
8 class KelasController extends Controller
9 {
10
11     0 references | 0 overrides
12     public function index(): View
13     {
14         $kelas = \App\Models\Kelas::with('walikelas')->get(); // Mengambil semua data kelas beserta wali kelasnya
15         return view('kelas.index', compact('kelas'));
16     }
17 }
```

Menampilkan Daftar Kelas (Method `index`)

Setelah method `index` ditambahkan, method ini akan digunakan untuk menampilkan daftar kelas di dalam view. Untuk itu, pastikan untuk memanggil method `index` di route `kelas.index`. Method ini akan mengambil data kelas dan mengirimkannya ke view `kelas.index`.

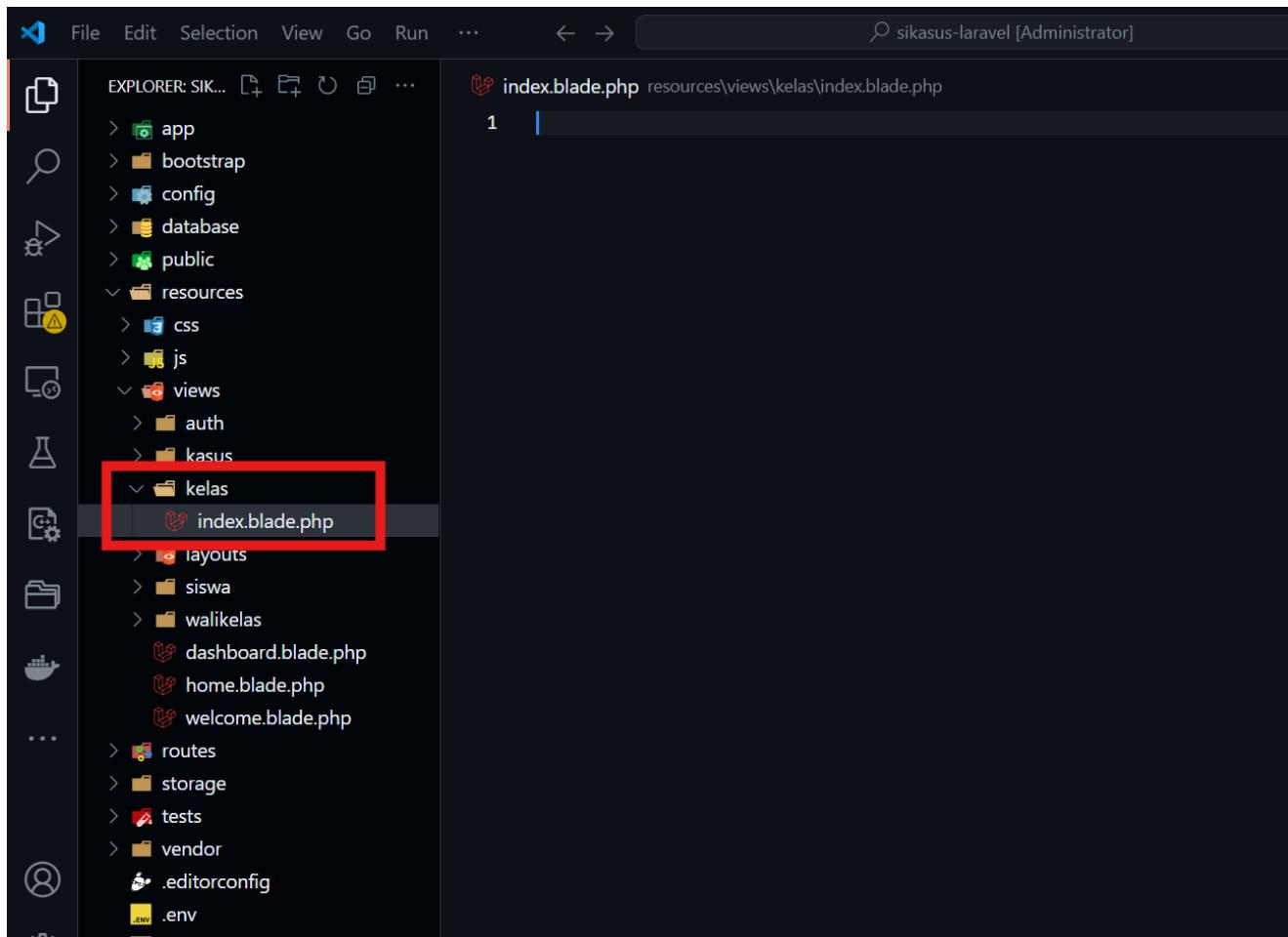
```
public function index()
{
    $kelas = \App\Models\Kelas::with('walikelas')->get(); // Mengambil semua kelas yang terhubung dengan wali kelas
    return view('kelas.index', compact('kelas')); // Mengirim data kelas ke view
}
```



```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
class KelasController extends Controller
{
    public function index(): View
    {
        $kelas = \App\Models\Kelas::with('walikelas')->get(); // Mengambil semua data kelas beserta wali kelasnya
        return view('kelas.index', compact('kelas'));
    }
}
```

Membuat View untuk Menampilkan Kelas (View index)

Selanjutnya, kita buat view `index.blade.php` di folder `resources/views/kelas`. Di sini, kita akan menampilkan daftar kelas yang telah diambil dari database.



Buka folder `resources/views` dan buat folder `kelas`, lalu buat file `index.blade.php` di dalamnya. Berikut adalah tampilan untuk daftar kelas:

```
@extends('layouts.app')

@section('title', 'Daftar kelas')

@section('content')
    <h1>Daftar kelas</h1>
    <a href="{{ route('kelas.create') }}" class="my-2 btn btn-primary">Tambah kelas</a>

    <div class="card mb-4">
        <div class="card-body">
            <table id="datatablesSimple">
                <thead>
                    <tr>
                        <th>#</th>
                        <th>Nama Kelas</th>
                        <th>Walikelas</th>
                        <th>Aksi</th>
                    </tr>
                </thead>
                <tfoot>
                    <tr>
                </tr>
            </tfoot>
        </div>
    </div>
```

```

<th>#</th>
<th>Nama Kelas</th>
<th>Walikelas</th>
<th>Aksi</th>
</tr>
</tfoot>
<tbody>
@foreach ($kelas as $item)
<tr>
<td>{{ $loop->iteration }}</td>
<td>{{ $item->nama_kelas }}</td>
<td>{{ $item->walikelas->nama_walikelas ?? '' }}</td>
<td>
<a href="{{ route('kelas.edit', $item->id_kelas) }}" class="btn btn-warning btn-sm">Edit</a>
</td>
</tr>
@endforeach
</tbody>
</table>
</div>
</div>
@endsection

```

```

File Edit Selection View Go Run ... < > sikasus-laravel [Administrator]
EXPLORER: SIK... D+ ⌂ ⌂ ⌂ ⌂ ⌂ ...
> app
> bootstrap
> config
> database
> public
resources
> css
> js
views
> auth
> kasus
> kelas
index.blade.php resources\views\kelas\index.blade.php
1 @extends('layouts.app')
2
3 @section('title', 'Daftar kelas')
4
5 @section('content')
6   <h1>Daftar kelas</h1>
7   <a href="{{ route('kelas.create') }}" class="my-2 btn btn-primary">Tambah kelas</a>
8
9   <div class="card mb-4">
10    <div class="card-body">
11      <table id="datatablesSimple">
12        <thead>
13          <tr>
14            <th>#</th>
15            <th>Nama Kelas</th>
16            <th>Walikelas</th>
17            <th>Aksi</th>
18          </tr>
19        </thead>
20        <tfoot>
21          <tr>
22            <th>#</th>
23            <th>Nama Kelas</th>
24            <th>Walikelas</th>
25            <th>Aksi</th>
26          </tr>
27        </tfoot>
28        <tbody>
29          @foreach ($kelas as $item)
30            <tr>
31              <td>{{ $loop->iteration }}</td>
32              <td>{{ $item->nama_kelas }}</td>

```

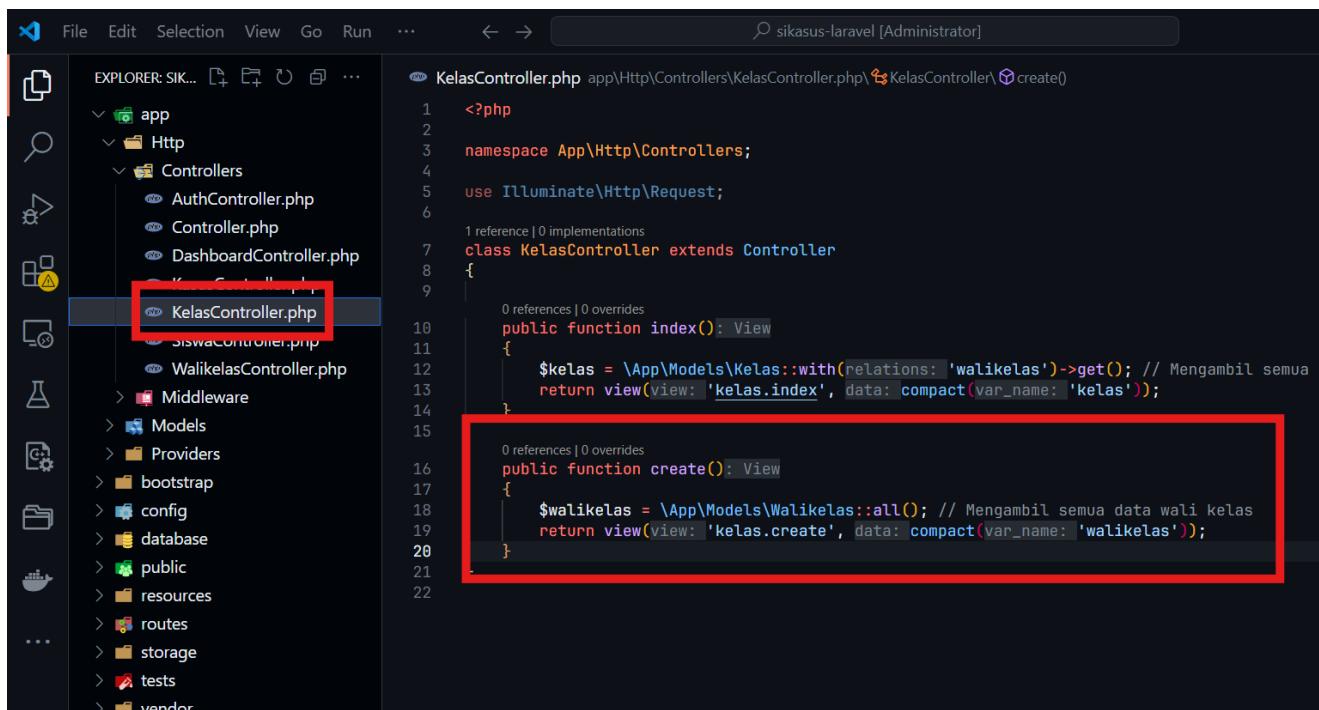
Menambahkan Kelas Baru ke Dalam Database (Method create dan store)

Setelah menampilkan daftar kelas, selanjutnya adalah membuat fungsi untuk menambahkan kelas baru. Kita akan membuat form untuk menambah data kelas di view `create`.

Controller akan menangani penyimpanan data kelas ke dalam database.

Tambahkan method `create` di `KelasController` untuk menampilkan form tambah kelas:

```
public function create()
{
    $walikelas = \App\Models\Walikelas::all(); // Mengambil semua data wali kelas
    return view('kelas.create', compact('walikelas'));
}
```



```
File Edit Selection View Go Run ... ← → sikasus-laravel [Administrator]
EXPLORER: SIK...
app
  Http
    Controllers
      AuthController.php
      Controller.php
      DashboardController.php
      KelasController.php
      KelasController.php (selected)
      SiswaController.php
      WalikelasController.php
    Middleware
  Models
  Providers
  bootstrap
  config
  database
  public
  resources
  routes
  storage
  tests
  vendor

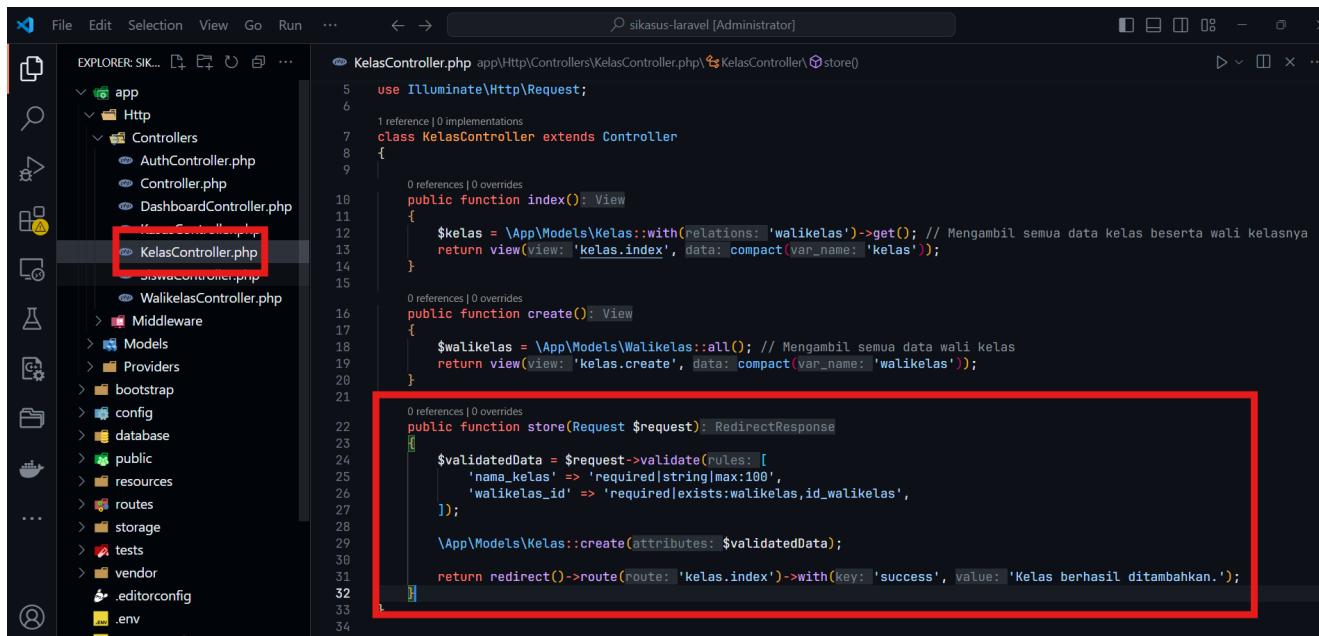
KelasController.php app\Http\Controllers\KelasController.php\create()
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6
7 1 reference | 0 implementations
8 class KelasController extends Controller
9 {
10
11     0 references | 0 overrides
12     public function index(): View
13     {
14         $kelas = \App\Models\Kelas::with(relations: 'walikelas')->get(); // Mengambil semua
15         return view(view: 'kelas.index', data: compact(var_name: 'kelas'));
16     }
17
18     0 references | 0 overrides
19     public function create(): View
20     {
21         $walikelas = \App\Models\Walikelas::all(); // Mengambil semua data wali kelas
22         return view(view: 'kelas.create', data: compact(var_name: 'walikelas'));
23     }
24 }
```

Kemudian, tambahkan method `store` untuk menyimpan kelas baru ke dalam database:

```
public function store(Request $request)
{
    $validatedData = $request->validate([
        'nama_kelas' => 'required|string|max:100',
        'walikelas_id' => 'required|exists:walikelas,id_walikelas',
    ]);

    \App\Models\Kelas::create($validatedData);

    return redirect()->route('kelas.index')->with('success', 'Kelas berhasil ditambahkan.');
}
```



```
use Illuminate\Http\Request;
class KelasController extends Controller
{
    public function index(): View
    {
        $kelas = \App\Models\Kelas::with('relations:walikelas')->get(); // Mengambil semua data kelas beserta wali kelasnya
        return view('kelas.index', compact('kelas'));
    }

    public function create(): View
    {
        $walikelas = \App\Models\Walikelas::all(); // Mengambil semua data wali kelas
        return view('kelas.create', compact('walikelas'));
    }

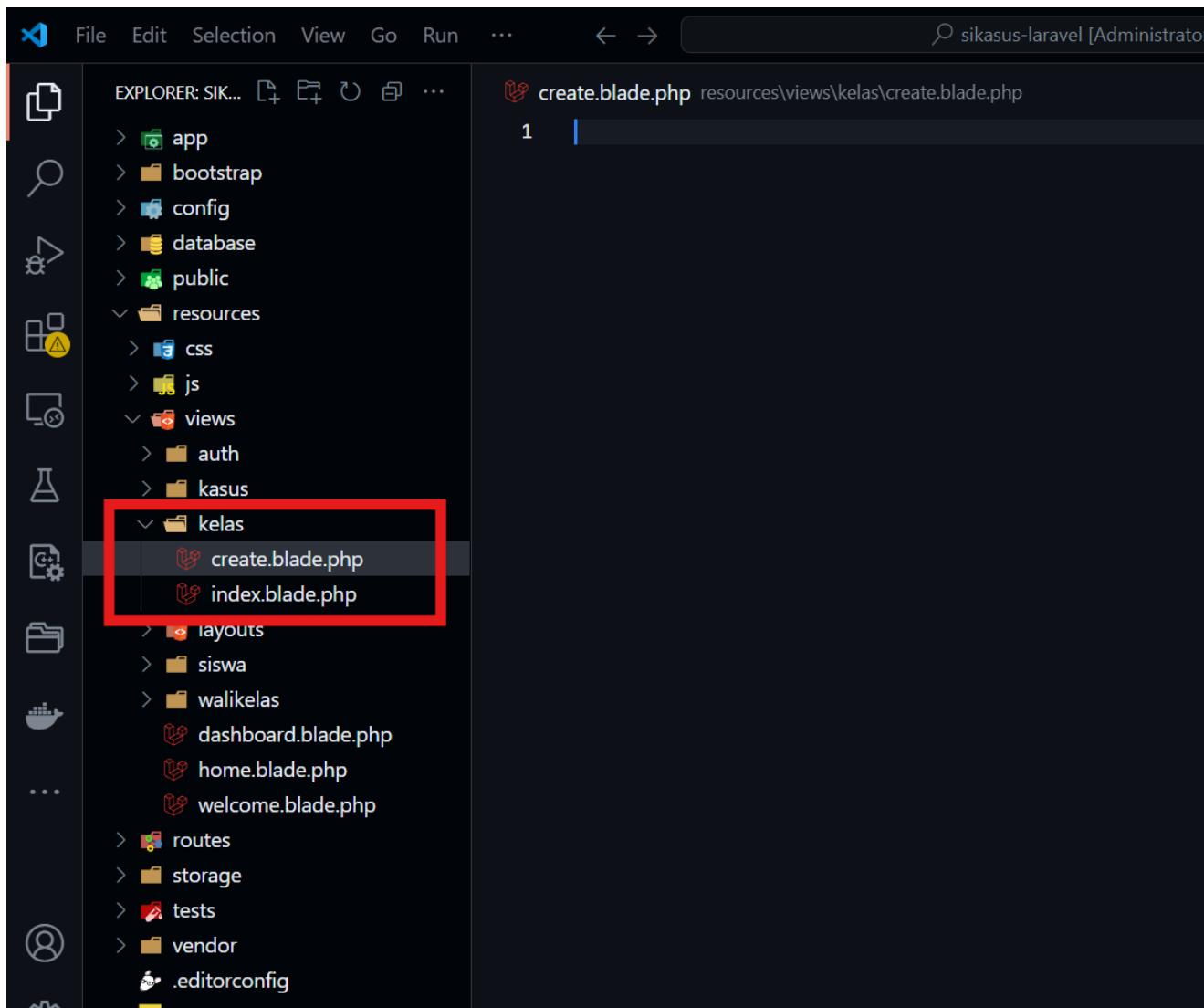
    public function store(Request $request): RedirectResponse
    {
        $validatedData = $request->validate([
            'nama_kelas' => 'required|string|max:100',
            'walikelas_id' => 'required|exists:walikelas,id_walikelas',
        ]);

        \App\Models\Kelas::create($validatedData);

        return redirect()->route('kelas.index')->with(['key' => 'success', 'value' => 'Kelas berhasil ditambahkan.']);
    }
}
```

Membuat Form untuk Menambah Kelas (View create)

Setelah method `create` dibuat, buat view `create.blade.php` untuk menampilkan form tambah kelas.



```
resources\views\kelas\create.blade.php
```

Berikut adalah kode untuk `create.blade.php` :

```

@extends('layouts.app')

@section('content')
    <div class="container">
        <h1>Tambah Kelas Baru</h1>

        <form action="{{ route('kelas.store') }}" method="POST">
            @csrf
            <div class="form-group">
                <label for="nama_kelas">Nama Kelas</label>
                <input type="text" name="nama_kelas" id="nama_kelas" class="form-control" required>
            </div>

            <div class="form-group">
                <label for="walikelas_id">Wali Kelas</label>
                <select name="walikelas_id" id="walikelas_id" class="form-control" required>
                    <option value="">Pilih Wali Kelas</option>
                    @foreach ($walikelas as $walikelasItem)
                        <option value="{{ $walikelasItem->id_walikelas }}>{{ $walikelasItem->nama_walikelas }}</option>
                    @endforeach
                </select>
            </div>

            <button type="submit" class="btn btn-primary mt-3">Simpan</button>
        </form>
    </div>
@endsection

```

```

CREATE: resources\views\kelas\create.blade.php
1  @extends('layouts.app')
2
3  @section('content')
4      <div class="container">
5          <h1>Tambah Kelas Baru</h1>
6
7          <form action="{{ route('kelas.store') }}" method="POST">
8              @csrf
9              <div class="form-group">
10                 <label for="nama_kelas">Nama Kelas</label>
11                 <input type="text" name="nama_kelas" id="nama_kelas" class="form-control" required>
12             </div>
13
14             <div class="form-group">
15                 <label for="walikelas_id">Wali Kelas</label>
16                 <select name="walikelas_id" id="walikelas_id" class="form-control" required>
17                     <option value="">Pilih Wali Kelas</option>
18                     @foreach ($walikelas as $walikelasItem)
19                         <option value="{{ $walikelasItem->id_walikelas }}>{{ $walikelasItem->nama_walikelas }}</option>
20                     @endforeach
21                 </select>
22             </div>
23
24             <button type="submit" class="btn btn-primary mt-3">Simpan</button>
25         </form>
26     </div>
27 @endsection
28

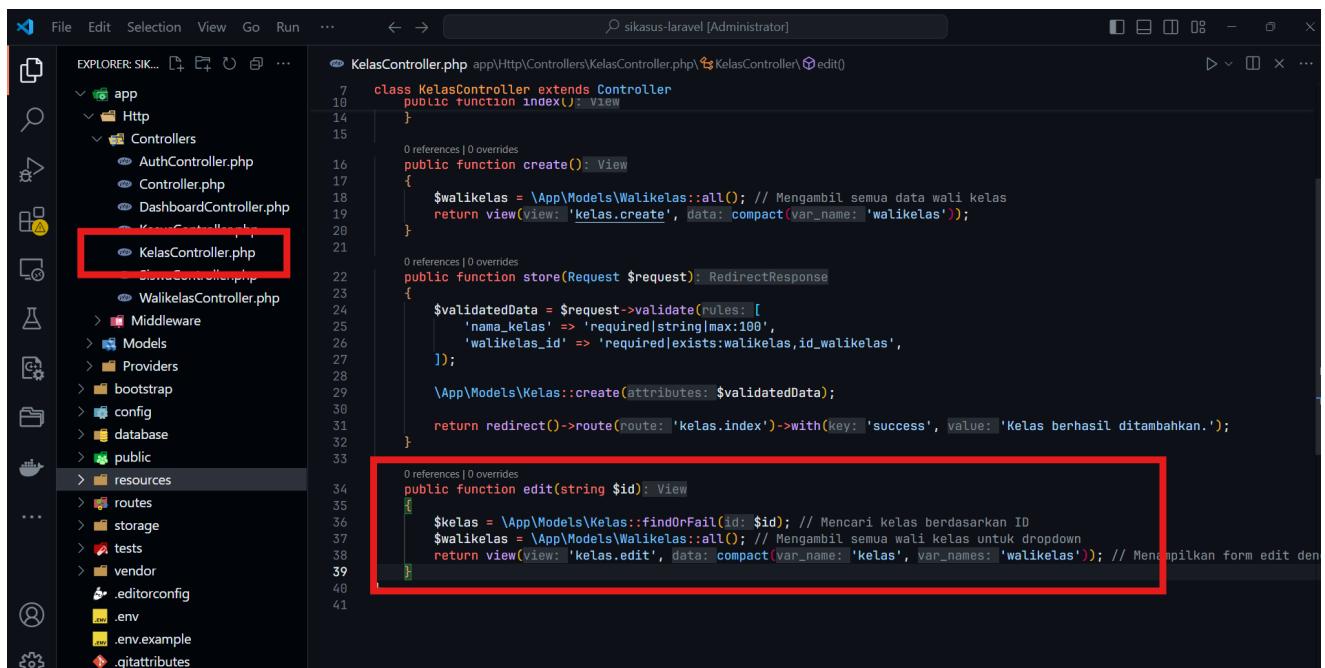
```

Edit dan Update Kelas ke Database

Langkah pertama adalah menambahkan dua method di dalam `KelasController` untuk menangani proses pengeditan dan pembaruan data kelas. Method `edit` digunakan untuk menampilkan form edit, sementara method `update` digunakan untuk memperbarui data kelas yang telah diedit.

Tambahkan method `edit` di dalam `KelasController`:

```
public function edit(string $id)
{
    $kelas = \App\Models\Kelas::findOrFail($id); // Mencari kelas berdasarkan ID
    $walikelas = \App\Models\Walikelas::all(); // Mengambil semua wali kelas untuk dropdown
    return view('kelas.edit', compact('kelas', 'walikelas')); // Menampilkan form edit dengan data kelas dan wali kelas
}
```



Kemudian, tambahkan method `update` untuk memproses pembaruan data kelas yang telah diedit:

```
public function update(Request $request, string $id)
{
    // Validasi input dari form edit
    $validatedData = $request->validate([
        'nama_kelas' => 'required|string|max:100',
        'walikelas_id' => 'required|exists:walikelas,id_walikelas',
    ]);

    $kelas = \App\Models\Kelas::findOrFail($id); // Menemukan kelas
```

berdasarkan ID

```
$kelas->update($validatedData); // Memperbarui data kelas dengan data  
yang baru  
  
return redirect()->route('kelas.index')->with('success', 'Kelas berhasil  
diperbarui.');// Redirect ke daftar kelas dengan pesan sukses  
}
```

```
class KelasController extends Controller
{
    public function store(Request $request): RedirectResponse
    {
        $validatedData = $request->validate([
            'nama_kelas' => 'required|string|max:100',
            'walikelas_id' => 'required|exists:walikelas,id_walikelas',
        ]);

        $kelas = \App\Models\Kelas::findOrFail($id); // Menemukan kelas berdasarkan ID
        $kelas->update($validatedData); // Memperbarui data kelas dengan data yang baru

        return redirect()->route('kelas.index')->with(key: 'success', value: 'Kelas berhasil diperbarui.');
    }

    public function edit(string $id): View
    {
        $kelas = \App\Models\Kelas::findOrFail($id); // Mencari kelas berdasarkan ID
        $walikelas = \App\Models\Walikelas::all(); // Mengambil semua wali kelas untuk dropdown
        return view(view: 'kelas.edit', data: compact(var_name: 'kelas', var_names: 'walikelas')) // Menampilkan form edit
    }

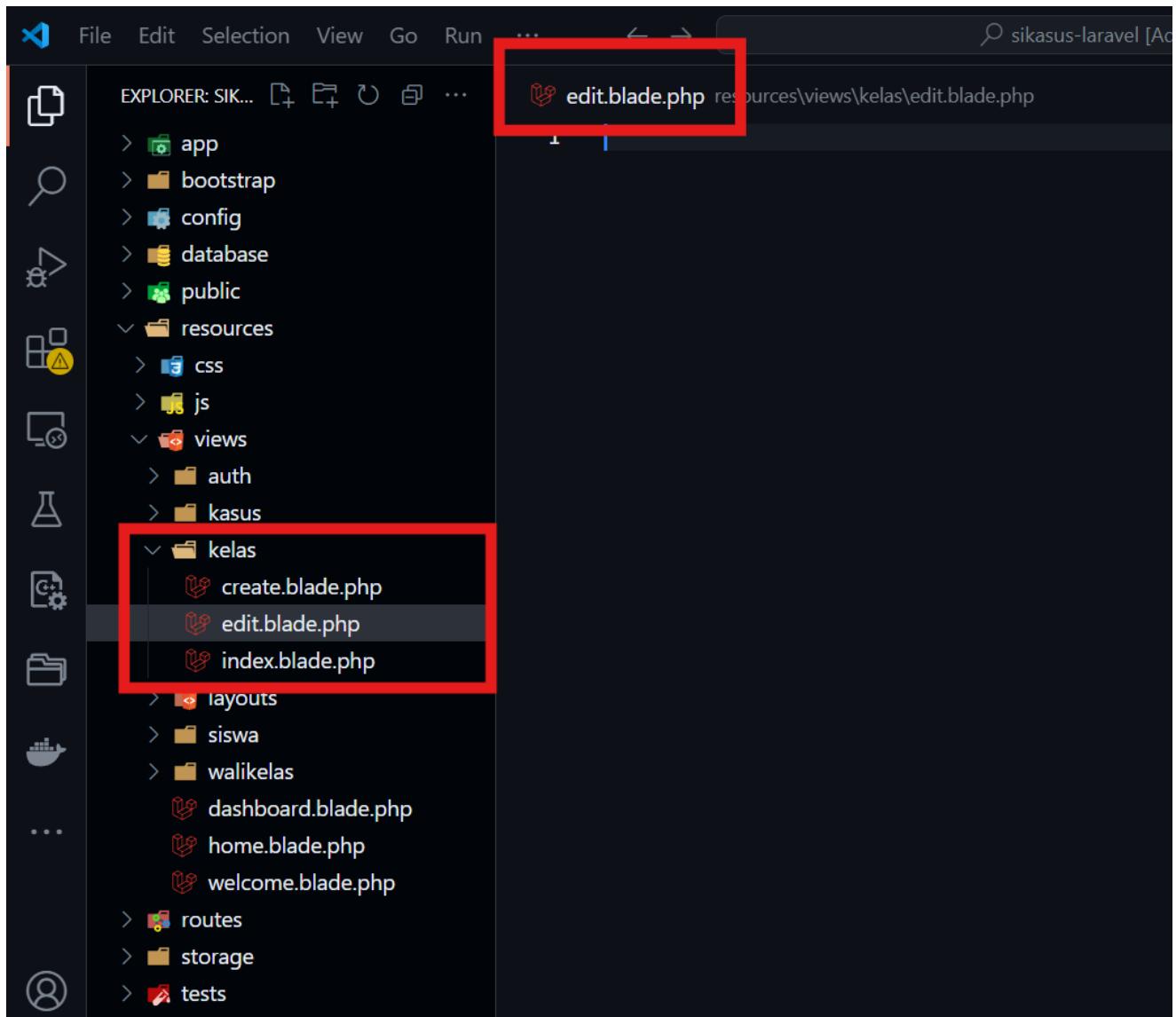
    public function update(Request $request, string $id): RedirectResponse
    {
        // Validasi input dari form edit
        $validatedData = $request->validate(rules: [
            'nama_kelas' => 'required|string|max:100',
            'walikelas_id' => 'required|exists:walikelas,id_walikelas',
        ]);

        $kelas = \App\Models\Kelas::findOrFail($id); // Menemukan kelas berdasarkan ID
        $kelas->update($validatedData); // Memperbarui data kelas dengan data yang baru

        return redirect()->route('kelas.index')->with(key: 'success', value: 'Kelas berhasil diperbarui.');
    }
}
```

Membuat View Form Edit Kelas (View edit)

Selanjutnya, buat tampilan untuk form edit kelas di dalam `resources/views/kelas/edit.blade.php`. View ini akan menampilkan data kelas yang sudah ada dan memungkinkan pengguna untuk memperbarui data kelas.



Buka folder `resources/views/kelas` , dan buat file baru `edit.blade.php` . Berikut adalah contoh kode untuk view `edit` :

```
@extends('layouts.app')

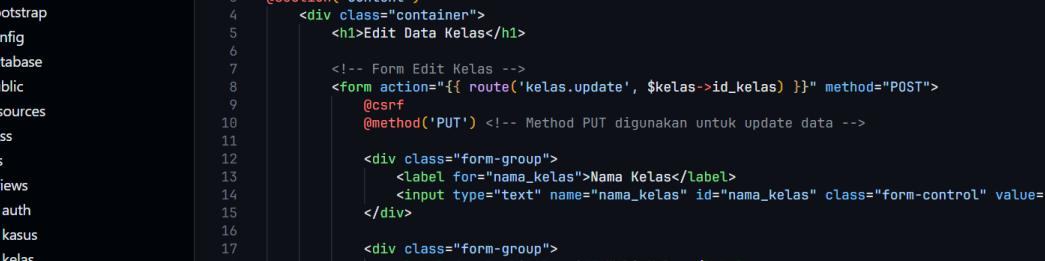
@section('content')
    <div class="container">
        <h1>Edit Data Kelas</h1>

        <!-- Form Edit Kelas -->
        <form action="{{ route('kelas.update', $kelas->id_kelas) }}">
            @method("POST")
            @csrf
            @method('PUT') <!-- Method PUT digunakan untuk update data -->

            <div class="form-group">
                <label for="nama_kelas">Nama Kelas</label>
                <input type="text" name="nama_kelas" id="nama_kelas" class="form-control" value="{{ $kelas->nama_kelas }}" required>
            </div>
        </form>
    </div>
</div>
```

```
<div class="form-group">
    <label for="walikelas_id">Wali Kelas</label>
    <select name="walikelas_id" id="walikelas_id" class="form-control" required>
        <option value="">Pilih Wali Kelas</option>
        @foreach ($walikelas as $walikelasItem)
            <option value="{{ $walikelasItem->id_walikelas }}"
                   {{ $kelas->walikelas_id == $walikelasItem-
                   id_walikelas ? 'selected' : '' }}>
                {{ $walikelasItem->nama_walikelas }}
            </option>
        @endforeach
    </select>
</div>

<button type="submit" class="btn btn-warning mt-3">Perbarui</button>
</form>
</div>
@endsection
```



```
edit.blade.php resources\views\kelas\edit.blade.php

2
3 @section('content')
4     <div class="container">
5         <h1>Edit Data Kelas</h1>
6
7         <!-- Form Edit Kelas -->
8         <form action="{{ route('kelas.update', $kelas->id_kelas) }}" method="POST">
9             @csrf
10            @method('PUT') <!-- Method PUT digunakan untuk update data -->
11
12             <div class="form-group">
13                 <label for="nama_kelas">Nama Kelas</label>
14                 <input type="text" name="nama_kelas" id="nama_kelas" class="form-control" value="{{ $kelas->nama_kelas }}>
15             </div>
16
17             <div class="form-group">
18                 <label for="walikelas_id">Wali Kelas</label>
19                 <select name="walikelas_id" id="walikelas_id" class="form-control" required>
20                     <option value="">Pilih Wali Kelas</option>
21                     @foreach ($walikelas as $walikelasItem)
22                         <option value="{{ $walikelasItem->id_walikelas }}>
23                             {{ $kelas->walikelas_id == $walikelasItem->id_walikelas ? 'selected' : '' }}>
24                             {{ $walikelasItem->nama_walikelas }}>
25                         </option>
26                     @endforeach
27                 </select>
28             </div>
29
30             <button type="submit" class="btn btn-warning mt-3">Perbarui</button>
31         </form>
32     </div>
33     @endsection
```

Menambahkan Method destroy di Controller

Setelah berhasil menambahkan dan mengedit kelas, langkah terakhir adalah menambahkan method untuk menghapus kelas. Tambahkan method `destroy` pada controller `KelasController` untuk menangani penghapusan data kelas dari database:

```
public function destroy(string $id)  
{
```

```

$kelas = \App\Models\Kelas::findOrFail($id);
$kelas->delete();

return redirect()->route('kelas.index')->with('success', 'Kelas berhasil
dihapus.');
}

```

```

class KelasController extends Controller
{
    public function update(Request $request, string $id): RedirectResponse
    {
        // Validasi input dari form edit
        $validatedData = $request->validate([
            'nama_kelas' => 'required|string|max:100',
            'walikelas_id' => 'required|exists:walikelas,id_walikelas',
        ]);

        $kelas = \App\Models\Kelas::findOrFail(id: $id); // Menemukan kelas berdasarkan ID
        $kelas->update($validatedData); // Memperbarui data kelas dengan data yang baru

        return redirect()->route(route: 'kelas.index')->with(key: 'success', value: 'Kelas berhasil diperbarui.');
    }

    public function destroy(string $id): RedirectResponse
    {
        $kelas = \App\Models\Kelas::findOrFail(id: $id);
        $kelas->delete();

        return redirect()->route(route: 'kelas.index')->with(key: 'success', value: 'Kelas berhasil dihapus.');
    }
}

```

Menambahkan Tombol Hapus di View

Untuk memungkinkan pengguna menghapus data kelas, tambahkan tombol hapus di view index.blade.php . Sudah ada form dengan metode `DELETE` yang akan mengirimkan permintaan untuk menghapus kelas:

```

<form action="{{ route('kelas.destroy', $item->id_kelas) }}" method="POST"
style="display:inline-block;">
    @csrf
    @method('DELETE')
    <button type="submit" class="btn btn-danger btn-sm" onclick="return
    confirm('Yakin ingin menghapus?')>Hapus</button>
</form>

```

```

File Edit Selection View Go Run ... ← → ⌘ sikasus-laravel (Administrator)
EXPLORER: SIKASUS-LARAVEL
resources\views\kelas\index.blade.php
<?php
...
10    <div class="card mb-4">
11        <div class="card-body">
12            <table id="datatablesSimple">
13                <thead>
14                    <tr>
15                        <th>Nama Kelas</th>
16                        <th>Aksi</th>
17                    </tr>
18                </thead>
19                <tbody>
20                    @foreach ($kelas as $item)
21                        <tr>
22                            <td>{{ $loop->iteration }}</td>
23                            <td>{{ $item->nama_kelas }}</td>
24                            <td>{{ $item->walikelas->nama_walikelas ?? '' }}</td>
25                            <td>
26                                <a href="{{ route('kelas.edit', $item->id_kelas) }}" class="btn btn-warning btn-sm">Edit
27                                <form action="{{ route('kelas.destroy', $item->id_kelas) }}" method="POST" style="display:inline-block;">
28                                    @csrf
29                                    @method('DELETE')
30                                    <button type="submit" class="btn btn-danger btn-sm" onclick="return confirm('Yakin ingin menghapus?')>">Hapus</button>
31                                </form>
32                            </td>
33                        </tr>
34                    @endforeach
35                </tbody>
36            </table>
37        </div>
38    </div>
39    @endsection
40
41
42
43
44
45
46
47
48
49

```

Topik	Sub Topik	Status
Menginstal Laravel 11		✓
Konfigurasi Proyek Laravel		✓
Membuat Model dan Migration		✓
Menambahkan Kolom di Dalam Migration		✓
	1. Migration Kasus	✓
	2. Migration Siswa	✓
	3. Migration Kelas	✓
	4. Migration Walikelas	✓
Menambahkan Mass Assignment		✓
	1. Model Kasus	✓
	2. Model Siswa	✓
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓
	Menampilkan Halaman Jenis Login	✓
	Menampilkan Form Login Berdasarkan Jenis	✓

Topik	Sub Topik	Status
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		✓
	Membuat Middleware untuk Autentikasi	✓
	Membuat Middleware untuk Role	✓
	Registrasi Middleware	✓
Membuat Blade Layout		✓
Membuat Tampilan Homepage		✓
Membuat Tampilan Dashboard Setelah Login		✓
Membuat Fitur Kasus		✓
	Menambahkan Routing untuk Kasus	✓
	Membuat Controller Kasus	✓
	Menampilkan Daftar Kasus (Method index)	✓
	Membuat View untuk Menampilkan Kasus (View index)	✓
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Kasus (View create)	✓
	Edit dan Update Kasus ke Database	✓
	Membuat View Form Edit Kasus (View edit)	✓
	Menambahkan Method Destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Siswa		✓
	Menambahkan Routing untuk Siswa	✓
	Membuat Controller Siswa	✓
	Menampilkan Daftar Siswa (Method index)	✓
	Membuat View untuk Menampilkan Siswa (View index)	✓

Topik	Sub Topik	Status
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Siswa (View create)	✓
	Menambahkan Method destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Walikelas		✓
	Menambahkan Routing untuk Walikelas	✓
	Membuat Controller Walikelas	✓
	Menampilkan Daftar Walikelas (Method index)	✓
	Membuat View untuk Menampilkan Walikelas (View index)	✓
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Walikelas (View create)	✓
	Edit dan Update Walikelas ke Database (Method edit dan update)	✓
	Membuat View Form Edit Walikelas (View edit)	✓
	Menambahkan Method destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Kelas		✓
	Menambahkan Routing untuk Kelas	✓
	Membuat Controller Kelas	✓
	Menampilkan Daftar Kelas (Method index)	✓
	Membuat View untuk Menampilkan Kelas (View index)	✓
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Kelas (View create)	✓
	Edit dan Update Kelas ke Database	✓
	Membuat View Form Edit Kelas (View edit)	✓
	Menambahkan Method destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓

Topik	Sub Topik	Status
Membuat Dashboard untuk Siswa	Menambahkan Routing untuk 'Siswa'	
	Membuat Controller 'Siswa'	

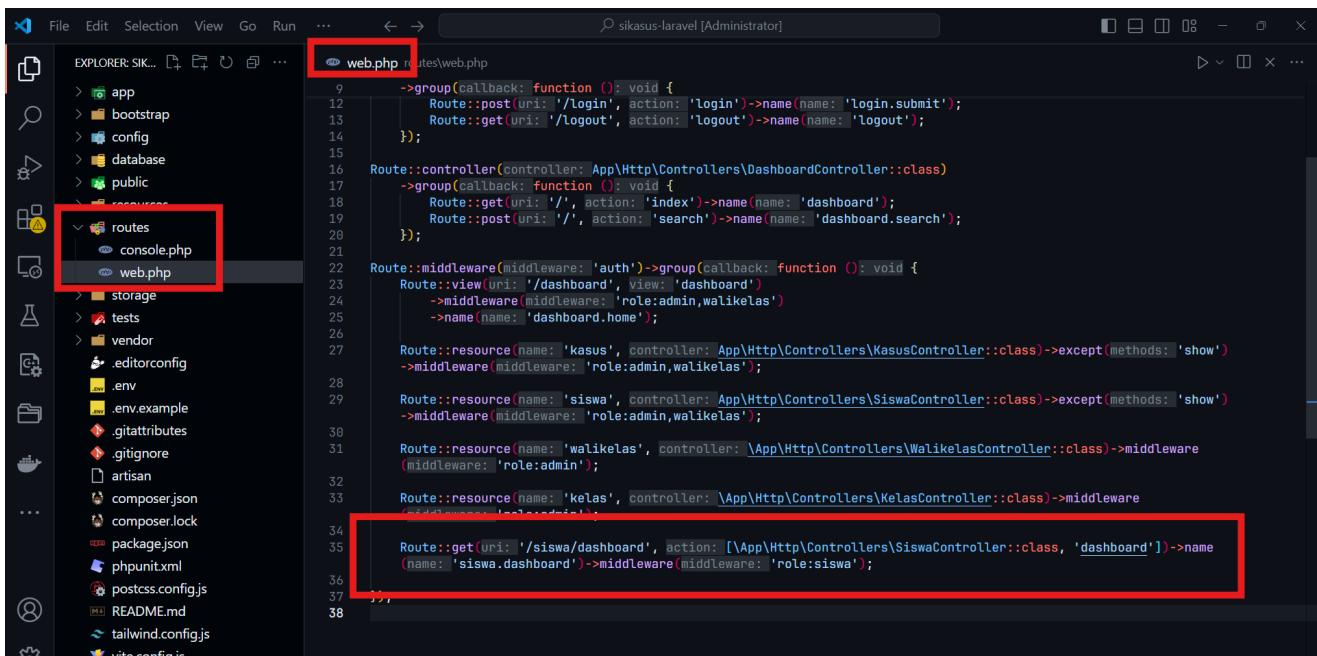
Membuat Dashboard untuk Siswa

Menambahkan Routing untuk 'Siswa'

Langkah pertama adalah menambahkan routing yang akan mengarahkan pengguna ke halaman dashboard siswa. Pastikan untuk menambahkan routing di `routes/web.php` yang memungkinkan siswa mengakses halaman dashboard mereka.

Buka file `routes/web.php` dan tambahkan kode berikut:

```
Route::get('/siswa/dashboard', [\App\Http\Controllers\SiswaController::class,
    'dashboard'])->name('siswa.dashboard')->middleware('role:siswa');
```



```
Route::group(callback: function () {
    Route::post(Uri: '/Login', action: 'login')->name(name: 'login.submit');
    Route::get(Uri: '/Logout', action: 'logout')->name(name: 'logout');
});

Route::controller(controller: App\Http\Controllers\DashboardController::class)
    ->group(callback: function () {
        Route::get(Uri: '/', action: 'index')->name(name: 'dashboard');
        Route::post(Uri: '/', action: 'search')->name(name: 'dashboard.search');
    });

Route::middleware(middleware: 'auth')->group(callback: function () {
    Route::view(Uri: '/dashboard', view: 'dashboard')
        ->middleware(middleware: 'role:admin,walikelas')
        ->name(name: 'dashboard.home');

    Route::resource(name: 'kasus', controller: App\Http\Controllers\KasusController::class)->except(methods: 'show')
        ->middleware(middleware: 'role:admin,walikelas');

    Route::resource(name: 'siswa', controller: App\Http\Controllers\SiswaController::class)->except(methods: 'show')
        ->middleware(middleware: 'role:admin,walikelas');

    Route::resource(name: 'walikelas', controller: \App\Http\Controllers\WalikelasController::class)->middleware
        (middleware: 'role:admin');

    Route::resource(name: 'kelas', controller: \App\Http\Controllers\KelasController::class)->middleware
        (middleware: 'role:admin');
});

Route::get(Uri: '/siswa/dashboard', action: [\App\Http\Controllers\SiswaController::class, 'dashboard'])->name
    (name: 'siswa.dashboard')->middleware(middleware: 'role:siswa');
```

Membuat Controller 'Siswa'

Selanjutnya, kita akan menambahkan method `dashboard` di controller `SiswaController`. Method ini akan mengambil data siswa yang sedang login, termasuk kasus-kasus yang terkait, dan menampilkannya di halaman dashboard.

Buka file `SiswaController.php` dan tambahkan kode berikut untuk method `dashboard`:

```

public function dashboard()
{
    $siswa = \App\Models\Siswa::with(['kasus', 'kelas'])-
>findOrFail(session('user')['id']); // Menampilkan data siswa dengan relasi kelas
    return view('siswa.dashboard', compact('siswa')); // Mengirim data siswa ke view
}

```

```

class SiswaController extends Controller
{
    public function store(Request $request): RedirectResponse
    {
        $request->validate([
            'nama_lengkap' => 'required|string|max:100',
            'nisn' => 'required|unique:siswa,nisn|max:20',
            'jenis_kelamin' => 'required|in:Laki-Laki,Perempuan',
            'tanggal_lahir' => 'required|date',
            'alamat' => 'required|string',
            'kelas_id' => 'required|exists:kelas,id_kelas', // Pastikan kelas_id ada di tabel kelas
        ]);

        \App\Models\Siswa::create(attributes: $request->all()); // Menyimpan data siswa baru
        return redirect()->route('siswa.index')->with(key: 'success', value: 'Siswa berhasil ditambahkan!');
    }

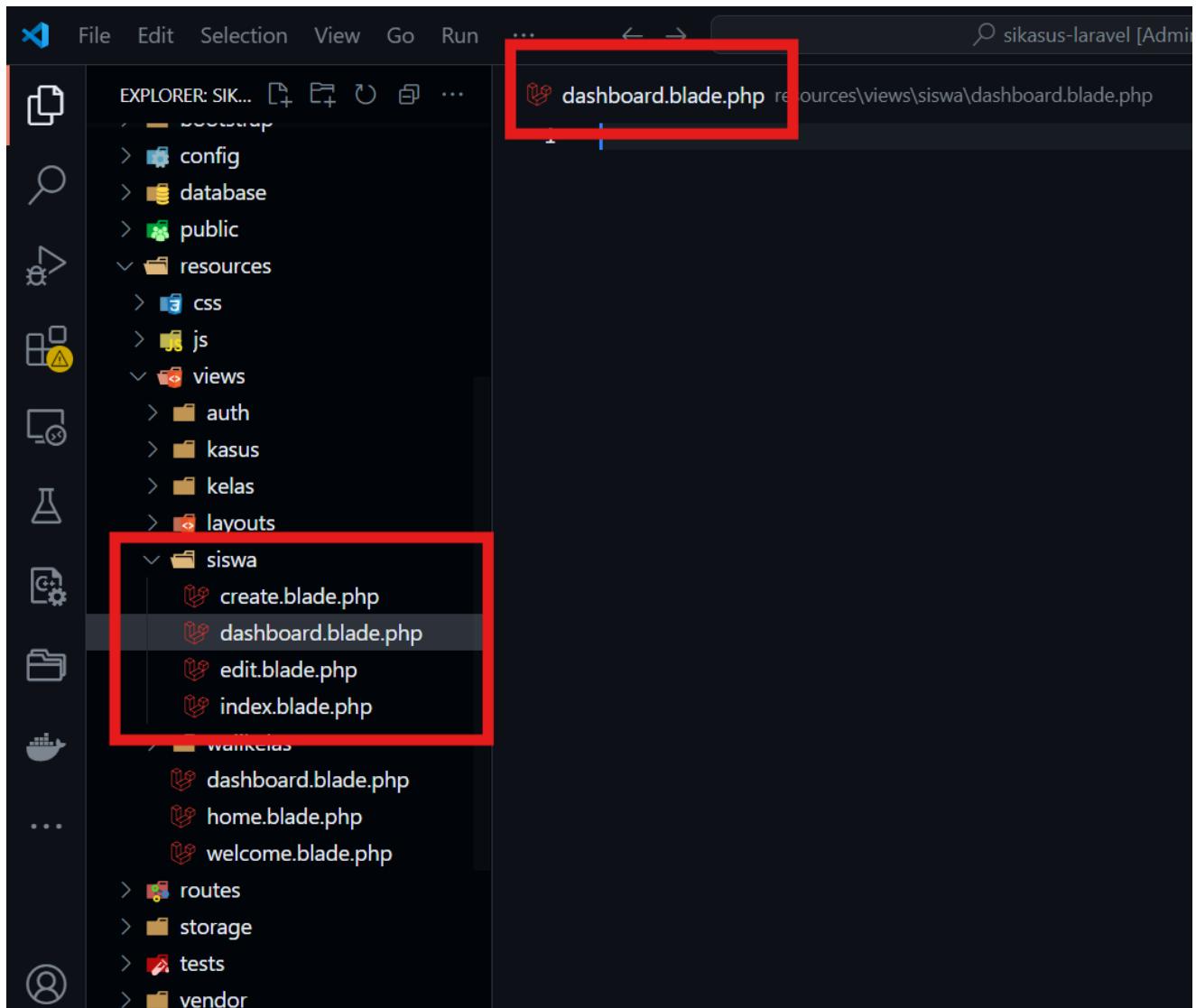
    public function destroy($id): RedirectResponse
    {
        $siswa = \App\Models\Siswa::findOrFail(id: $id); // Menemukan siswa berdasarkan ID
        $siswa->delete(); // Menghapus data siswa
        return redirect()->route('siswa.index')->with(key: 'success', value: 'Siswa berhasil dihapus!');
    }

    public function dashboard(): View
    {
        $siswa = \App\Models\Siswa::with(relations: ['kasus', 'kelas'])->findOrFail(id: session(key: 'user')['id']); // Mengirim data siswa ke view
    }
}

```

Membuat View untuk Menampilkan Kasus Siswa (View dashboard)

Buat view `dashboard.blade.php` di dalam folder `resources/views/siswa` untuk menampilkan riwayat kasus siswa.



Kemudian, buat kode untuk halaman dashboard siswa :

```
@extends('layouts.app')
@section('title', "Riwayat Kasus - $siswa->nama_lengkap")

@section('content')
    <div class="container mt-4">
        <div class="card">
            <div class="card-header bg-primary text-white">
                <div class="d-flex align-items-center">
                    <div class="fs-2 me-3">👤</div>
                    <div>
                        <h1 class="card-title mb-1">{{ $siswa->nama_lengkap }}</h1>
                        <p class="card-text">NISN: {{ $siswa->nisn }}</p>
                    </div>
                </div>
            </div>

            <div class="card-body">
                <div class="row mb-3">
```

```

<div class="col-md-6">
    <strong>Kelas:</strong>
    {{ $siswa->kelas ? $siswa->kelas->nama_kelas :
'Belum ditentukan' }}
</div>
<div class="col-md-6">
    <strong>Total Kasus:</strong>
    <span class="badge bg-danger">{{ $siswa->kasus-
>count() }}</span>
</div>
</div>

<h2 class="h4 mb-3">Riwayat Kasus</h2>

@if ($siswa->kasus->count() > 0)
<div class="list-group">
    @foreach ($siswa->kasus as $kasusItem)
        <div class="list-group-item list-group-item-
action">
            <div class="d-flex w-100 justify-content-
between">
                <h5 class="mb-1">
                    {{ \Carbon\Carbon::parse($kasusItem-
>tanggal_kasus)->translatedFormat('d F Y') }}
                </h5>
                </div>
                <p class="mb-1">{{ $kasusItem-
>deskripsi_kasus }}</p>
            </div>
        @endforeach
    </div>
    @else
        <div class="alert alert-info text-center" role="alert">
            Tidak Ada Riwayat Kasus
        </div>
    @endif
</div>
</div>
</div>
@endsection

```

```
resources\views\siswa\dashboard.blade.php

1 @extends('Layouts.app')
2 @section('title', "Riwayat Kasus - $siswa->nama_lengkap ")
3
4 @section('content')
5     <div class="container mt-4">
6         <div class="card">
7             <div class="card-header bg-primary text-white">
8                 <div class="d-flex align-items-center">
9                     <div class="fs-2 me-3">👤</div>
10                    <div>
11                        <h1 class="card-title mb-1">{{ $siswa->nama_lengkap }}</h1>
12                        <p class="card-text">NISN: {{ $siswa->nisn }}</p>
13                    </div>
14                </div>
15            </div>
16        </div>
17
18        <div class="card-body">
19            <div class="row mb-3">
20                <div class="col-md-6">
21                    <strong>Kelas:</strong>
22                    {{ $siswa->kelas ? $siswa->kelas->nama_kelas : 'Belum ditentukan' }}
23                </div>
24                <div class="col-md-6">
25                    <strong>Total Kasus:</strong>
26                    <span class="badge bg-danger">{{ $siswa->kasus->count() }}</span>
27                </div>
28            </div>
29
30            <h2 class="h4 mb-3">Riwayat Kasus</h2>
31
32            @if ($siswa->kasus->count() > 0)
33                <div class="list-group">
34                    @foreach ($siswa->kasus as $kasusItem)
35                        <div class="list-group-item list-group-item-action">
36                            <div class="d-flex w-100 justify-content-between">
37                                <h5 class="mb-1">
38                                    {{ \Carbon\Carbon::parse($kasusItem->tanggal_kasus)->translatedFormat('d F Y') }}
```

Penjelasan :

- **Routing:** Kita menambahkan route dengan menggunakan middleware `role:siswa`, yang memastikan hanya siswa yang dapat mengakses halaman dashboard mereka.
 - **Controller SiswaController :** Method `dashboard` mengambil data siswa yang sedang login melalui `session('user')['id']`. Data tersebut juga mencakup informasi kasus yang terkait dengan siswa tersebut.
 - **View `dashboard.blade.php` :** Di sini kita menampilkan informasi siswa, seperti nama, NISN, kelas, serta riwayat kasus siswa yang terdaftar. Jika tidak ada kasus, pesan pemberitahuan "Tidak Ada Riwayat Kasus" akan ditampilkan.

Topik	Sub Topik	Status
Menginstal Laravel 11		<input checked="" type="checkbox"/>
Konfigurasi Proyek Laravel		<input checked="" type="checkbox"/>
Membuat Model dan Migration		<input checked="" type="checkbox"/>
Menambahkan Kolom di Dalam Migration		<input checked="" type="checkbox"/>
	1. Migration Kasus	<input checked="" type="checkbox"/>
	2. Migration Siswa	<input checked="" type="checkbox"/>
	3. Migration Kelas	<input checked="" type="checkbox"/>
	4. Migration Walikelas	<input checked="" type="checkbox"/>
Menambahkan Mass Assignment		<input checked="" type="checkbox"/>
	1. Model Kasus	<input checked="" type="checkbox"/>

Topik	Sub Topik	Status
	2. Model Siswa	✓
	3. Model Kelas	✓
	4. Model Walikelas	✓
Menjalankan Proses Migrate		✓
Menerapkan Autentikasi		✓
Membuat Controller Auth		✓
Menambahkan Fungsi di AuthController		✓
	Menampilkan Halaman Jenis Login	✓
	Menampilkan Form Login Berdasarkan Jenis	✓
	Proses Login	✓
	Proses Logout	✓
	Membuat Routing untuk Autentikasi	✓
Membuat View untuk Menampilkan Halaman Login		✓
	Halaman Pilihan Login	✓
	Membuat Form Login	✓
Membuat Middleware		✓
	Membuat Middleware untuk Autentikasi	✓
	Membuat Middleware untuk Role	✓
	Registrasi Middleware	✓
Membuat Blade Layout		✓
Membuat Tampilan Homepage		✓
Membuat Tampilan Dashboard Setelah Login		✓
Membuat Fitur Kasus		✓
	Menambahkan Routing untuk Kasus	✓
	Membuat Controller Kasus	✓
	Menampilkan Daftar Kasus (Method index)	✓
	Membuat View untuk Menampilkan Kasus (View index)	✓
	Menambahkan Kasus Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Kasus (View create)	✓
	Edit dan Update Kasus ke Database	✓

Topik	Sub Topik	Status
	Membuat View Form Edit Kasus (View edit)	✓
	Menambahkan Method Destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Siswa		✓
	Menambahkan Routing untuk Siswa	✓
	Membuat Controller Siswa	✓
	Menampilkan Daftar Siswa (Method index)	✓
	Membuat View untuk Menampilkan Siswa (View index)	✓
	Menambahkan Siswa Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Siswa (View create)	✓
	Menambahkan Method destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Walikelas		✓
	Menambahkan Routing untuk Walikelas	✓
	Membuat Controller Walikelas	✓
	Menampilkan Daftar Walikelas (Method index)	✓
	Membuat View untuk Menampilkan Walikelas (View index)	✓
	Menambahkan Walikelas Baru ke Dalam Database (Method create dan store)	✓
	Membuat Form untuk Menambah Walikelas (View create)	✓
	Edit dan Update Walikelas ke Database (Method edit dan update)	✓
	Membuat View Form Edit Walikelas (View edit)	✓
	Menambahkan Method destroy di Controller	✓
	Menambahkan Tombol Hapus di View	✓
Membuat Fitur Kelas		✓
	Menambahkan Routing untuk Kelas	✓
	Membuat Controller Kelas	✓
	Menampilkan Daftar Kelas (Method index)	✓

Topik	Sub Topik	Status
	Membuat View untuk Menampilkan Kelas (View index)	<input checked="" type="checkbox"/>
	Menambahkan Kelas Baru ke Dalam Database (Method create dan store)	<input checked="" type="checkbox"/>
	Membuat Form untuk Menambah Kelas (View create)	<input checked="" type="checkbox"/>
	Edit dan Update Kelas ke Database	<input checked="" type="checkbox"/>
	Membuat View Form Edit Kelas (View edit)	<input checked="" type="checkbox"/>
	Menambahkan Method destroy di Controller	<input checked="" type="checkbox"/>
	Menambahkan Tombol Hapus di View	<input checked="" type="checkbox"/>
Membuat Dashboard untuk Siswa		<input checked="" type="checkbox"/>
	Menambahkan Routing untuk 'Siswa'	<input checked="" type="checkbox"/>
	Membuat Controller 'Siswa'	<input checked="" type="checkbox"/>