LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL XI

LARAVEL 1

Oleh:

(TSAQIF HISYAM SAPUTRA) (2311104024)

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

1.1 Dasar Teori

Laravel adalah framework PHP yang bersifat open-source dan menggunakan arsitektur MVC (Model-View-Controller). Laravel menyediakan berbagai fitur yang memudahkan pengembangan aplikasi web modern.

Routing dalam Laravel adalah mekanisme untuk mengarahkan HTTP request ke controller atau closure yang sesuai. Laravel mendukung berbagai jenis route seperti:

- Route tanpa parameter
- Route dengan parameter wajib
- Route dengan optional parameter

View adalah komponen yang bertanggung jawab untuk menampilkan data kepada pengguna. Laravel menggunakan Blade sebagai template engine default.

Blade Template Engine adalah template engine sederhana namun powerful yang disediakan Laravel. Blade menyediakan fitur seperti:

- Template inheritance
- Sections
- Control structures (@if, @for, @foreach, @while)
- Komponen dan slot

Asset Management adalah pengelolaan file statis seperti CSS, JavaScript, dan gambar. Laravel menyediakan helper `asset()` untuk mengakses file di folder public.

Controller adalah kelas yang mengelompokkan logika penanganan request. Controller memisahkan logika bisnis dari route definition.

1.2 Tujuan

1. Memahami konsep routing pada framework Laravel
2. Mampu membuat route dengan berbagai jenis parameter
3. Memahami pengelolaan asset (CSS, JS, Image) pada Laravel
4. Mampu menggunakan Blade Template Engine untuk membuat view dinamis
5. Memahami penggunaan controller untuk mengelola logika aplikasi

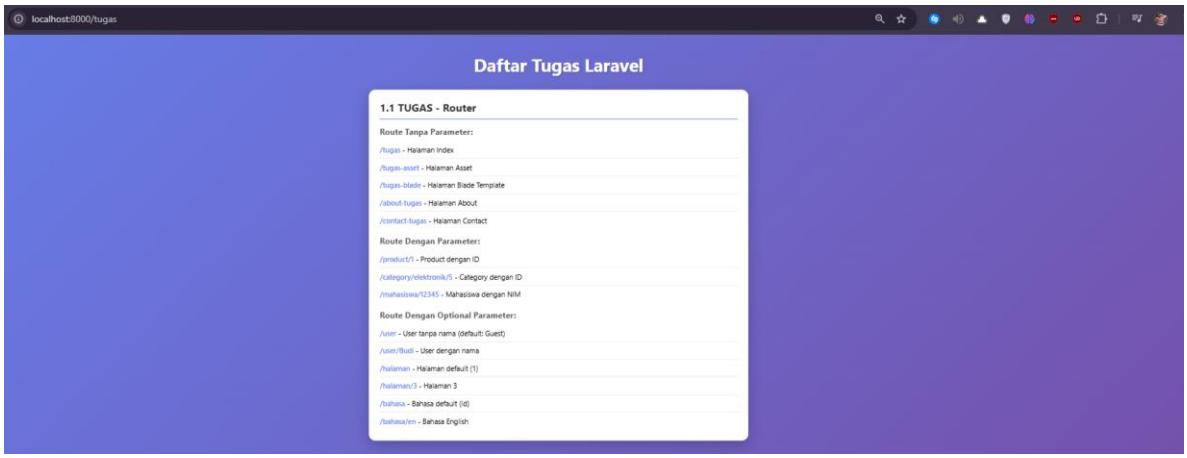
1.3 Manfaat

1. Dapat mengembangkan aplikasi web dengan struktur yang terorganisir
2. Memahami best practice dalam pengembangan aplikasi Laravel
3. Mampu membuat tampilan web yang dinamis menggunakan Blade
4. Dapat mengelola asset dengan efisien
5. Memahami konsep MVC dalam pengembangan web

BAB II

HASIL PRAKTIKUM

2.1 Tugas – Router



2.1.1 Route Tanpa Parameter (5 Route)

```
// 1. Route ke halaman index tugas
Route::get('/tugas', [TugasController::class, 'index'])->name('tugas.index');

// 2. Route ke halaman asset
Route::get('/tugas-asset', [TugasController::class, 'asset'])-
>name('tugas.asset');

// 3. Route ke halaman blade template
Route::get('/tugas-blade', [TugasController::class, 'blade'])-
>name('tugas.blade');

// 4. Route ke halaman about
Route::get('/about-tugas', function () {
    return view('tugas.about');
})->name('tugas.about');

// 5. Route ke halaman contact
Route::get('/contact-tugas', function () {
    return view('tugas.contact');
})->name('tugas.contact');
```

- Route tanpa parameter adalah route yang tidak memerlukan input dari URL
- Dapat menggunakan closure atau controller
- Menggunakan method `name()` untuk memberikan nama route

2.1.2 Route Dengan Parameter (3 Route)

```
// 1. Route dengan parameter id
Route::get('/product/{id}', [TugasController::class, 'showProduct'])-
>name('tugas.product');

// 2. Route dengan 2 parameter (category dan id)
Route::get('/category/{category}/{id}', [TugasController::class,
'showCategory'])->name('tugas.category');

// 3. Route dengan parameter nim
Route::get('/mahasiswa/{nim}', function ($nim) {
    return view('tugas.mahasiswa', compact('nim'));
})->name('tugas.mahasiswa');
```

- Parameter ditulis dalam kurung kurawal `{parameter}`
- Parameter wajib harus diisi, jika tidak akan error 404
- Dapat memiliki lebih dari satu parameter

2.1.3 Route Dengan Optional Parameter (3 Route)

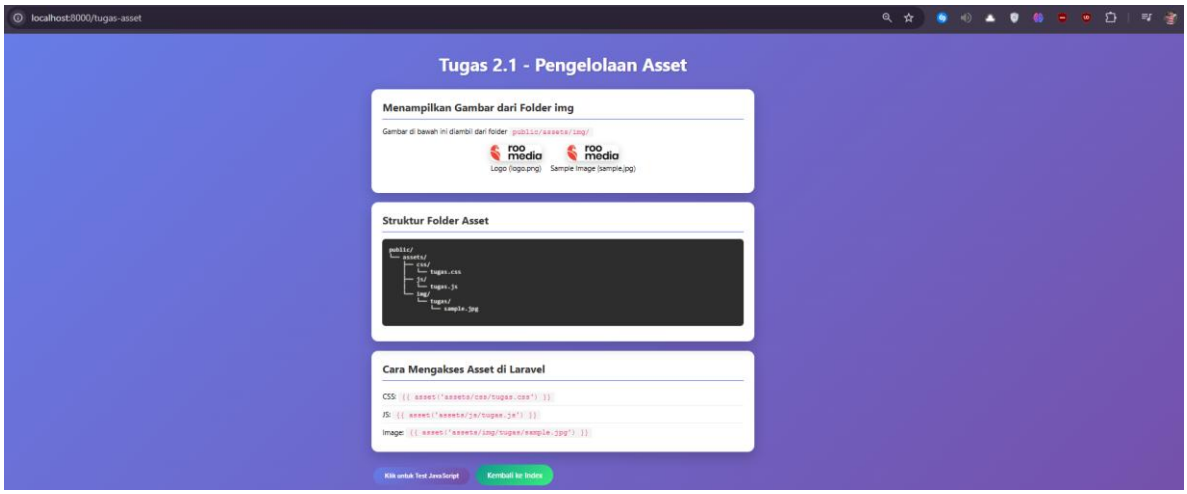
```
// 1. Route dengan optional parameter name
Route::get('/user/{name?}', [TugasController::class, 'showUser'])-
>name('tugas.user');

// 2. Route dengan optional parameter page
Route::get('/halaman/{page?}', [TugasController::class, 'showPage'])-
>name('tugas.page');

// 3. Route dengan optional parameter lang
Route::get('/bahasa/{lang?}', [TugasController::class, 'showLang'])-
>name('tugas.lang');
```

- Optional parameter ditandai dengan tanda tanya `{parameter?}`
- Harus memiliki nilai default di controller
- Jika tidak diisi, akan menggunakan nilai default

2.2 Tugas – View (Pengelolaan Asset)



2.2.1 Struktur Folder Asset

```
public/
└── assets/
    ├── css/
    │   └── tugas.css
    ├── js/
    │   └── tugas.js
    └── img/
        └── tugas/
            └── sample.jpg
```

2.2.2 Mengakses Asset di View

****Mengakses CSS:****

- **html**
  - <link rel="stylesheet" href="{{ asset('assets/css/tugas.css') }}" />

****Mengakses JavaScript:****

- **html**
  - <script src="{{ asset('assets/js/tugas.js') }}"></script>

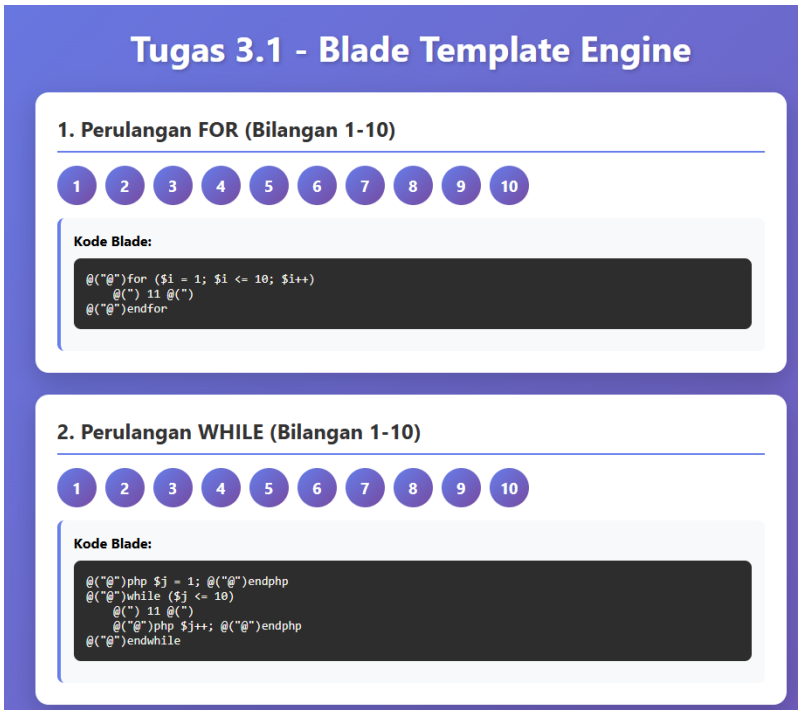
****Mengakses Gambar:****

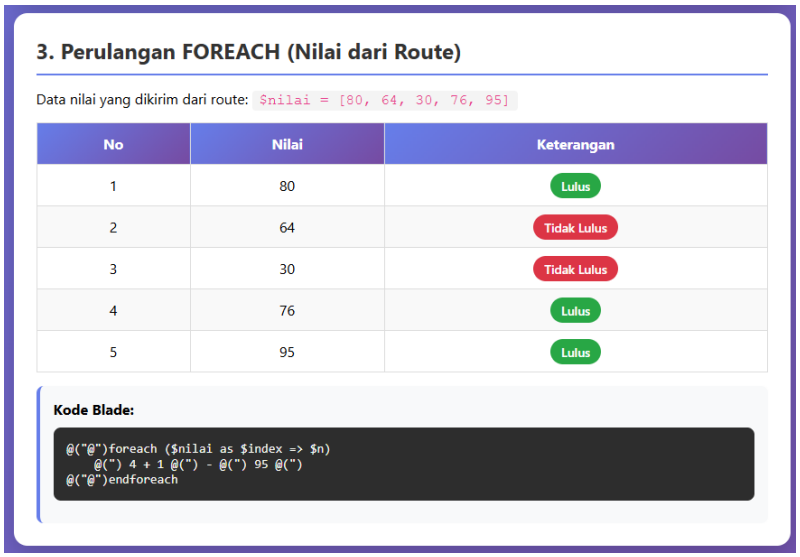
- **html**
  -
  -

- Helper `asset()` menghasilkan URL lengkap ke folder public

- Semua asset statis harus diletakkan di folder public
- Struktur folder dapat disesuaikan dengan kebutuhan project

2.3 Tugas – Blade Template Engine





2.3.1 Perulangan FOR (Bilangan 1-10)

blade

```
@for ($i = 1; $i <= 10; $i++)
    <span class="number-item">{{ $i }}</span>
@endfor
```

2.3.2 Perulangan WHILE (Bilangan 1-10)

blade

```
@php $j = 1; @endphp
@while ($j <= 10)
    <span class="number-item">{{ $j }}</span>
    @php $j++; @endphp
@endwhile
```

2.3.3 Perulangan FOREACH (Nilai dari Route)

Controller:

php

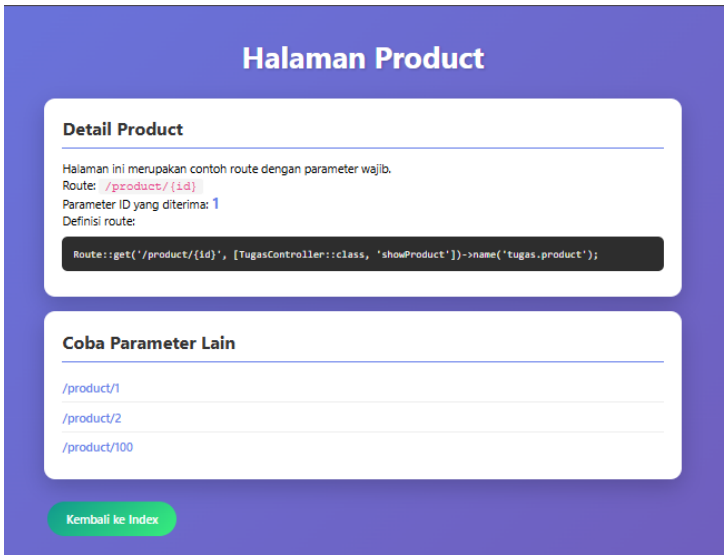
```
public function blade()
{
    $nilai = [80, 64, 30, 76, 95];
    return view('tugas.blade', compact('nilai'));
}
```

View:

blade

```
@foreach ($nilai as $index => $n)
    <tr>
        <td>{{ $index + 1 }}</td>
        <td>{{ $n }}</td>
        <td>
            @if ($n >= 75)
                <span class="badge badge-success">Lulus</span>
            @else
                <span class="badge badge-danger">Tidak
Lulus</span>
            @endif
        </td>
    </tr>
@endforeach
```

2.4 Tugas – Cotroller



2.4.1 Membuat Controller

php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class TugasController extends Controller
{
    public function index()
    {
        return view('tugas.index');
    }

    public function asset()
    {
        return view('tugas.asset');
    }

    public function blade()
    {
        $nilai = [80, 64, 30, 76, 95];
        return view('tugas.blade', compact('nilai'));
    }

    public function showProduct($id)
```

```
    {
        return view('tugas.product', compact('id'));
    }

    public function showUser($name = 'Guest')
    {
        return view('tugas.user', compact('name'));
    }
}
```

2.4.2 Route Mengarah ke Controller Php

```
Route::get('/tugas', [TugasController::class, 'index'])-
>name('tugas.index');
Route::get('/tugas-blade', [TugasController::class, 'blade'])-
>name('tugas.blade');
Route::get('/product/{id}', [TugasController::class, 'showProduct'])-
>name('tugas.product');
```

- **Controller dibuat di folder `app/Http/Controllers`**
- **Route menggunakan array `[ControllerClass::class, 'methodName']`**
- **Data dikirim ke view menggunakan `compact()` atau array**

BAB III

KESIMPULAN & SARAN

3.1 Kesimpulan

Kombinasi Route, View, Blade, dan Controller dalam Laravel membentuk arsitektur MVC yang solid untuk pengembangan aplikasi web, di mana fleksibilitas routing dan pemisahan logika bisnis melalui Controller membuat kode menjadi lebih terstruktur dan mudah dipelihara. Kemudahan ini diperlengkap oleh manajemen aset yang terorganisir serta sintaks Blade Template Engine yang bersih dan efisien dalam menangani struktur kontrol maupun *conditional rendering*, sehingga memudahkan proses *maintenance* jangka panjang.

3.2 Saran

Buat meningkatkan kualitas dan efisiensi pengembangan ke depannya, disarankan agar menggunakan Resource Controller dan Route Groups demi struktur URL yang lebih rapi dan sesuai konvensi, serta menerapkan Form Request Validation untuk keamanan input yang lebih baik. Selain itu, sebaiknya optimalkan sisi tampilan dengan memanfaatkan Blade Components dan Layouts untuk membuat elemen UI yang konsisten dan *reusable*, serta implementasikan Asset Bundling menggunakan Vite atau Laravel Mix guna memaksimalkan performa CSS dan JavaScript di lingkungan produksi.