LAPORAN PRAKTIKUM

PERANCANGAN DAN PEMROGRAMAN WEB

MODUL 11

LARAVEL

Oleh:

Aulia Jasifa Br Ginting 2311104060

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

DIREKTORAT KAMPUS PURWOKERTO

UNIVERSITAS TELKOM

2025

BAB I

PENDAHULUAN

A. Dasar Teori

Laravel merupakan sebuah web application framework berbasis PHP yang menerapkan arsitektur Model-View-Controller (MVC) untuk mempermudah pengembangan aplikasi web yang terstruktur, efisien, dan mudah dipelihara. Framework sendiri dapat diartikan sebagai kerangka kerja yang menyediakan kumpulan fungsi, library, dan class siap pakai sehingga pengembang tidak perlu menulis kode dari awal. Dengan menggunakan framework seperti Laravel, proses pengembangan aplikasi menjadi lebih cepat, konsisten, serta memiliki standar penulisan kode yang jelas. Laravel menyediakan berbagai fitur bawaan seperti routing, controller, view dengan Blade Template Engine, ORM (Eloquent), validasi data, manajemen session, dan keamanan aplikasi, yang sangat membantu dalam pengembangan aplikasi web modern.

Konsep MVC pada Laravel memisahkan aplikasi menjadi tiga komponen utama, yaitu Model yang bertugas mengelola dan memanipulasi data ke database, View yang bertanggung jawab menampilkan antarmuka pengguna, serta Controller yang berfungsi sebagai penghubung antara Model dan View dengan mengatur alur logika aplikasi. Pemisahan ini bertujuan agar kode program lebih terorganisir, mudah dikembangkan, dan lebih sederhana dalam proses debugging maupun maintenance. Selain itu, Laravel memiliki alur kerja yang jelas, dimulai dari request yang diterima oleh routing, diproses oleh controller, lalu hasilnya ditampilkan melalui view kepada pengguna. Dengan dokumentasi yang lengkap dan komunitas yang besar, Laravel menjadi salah satu framework PHP yang paling banyak digunakan saat ini dalam pengembangan aplikasi web dinamis

B. Tujuan

Tujuan utama dari praktikum ini adalah agar mahasiswa mampu memahami konsep dasar framework Laravel serta mengimplementasikan pola arsitektur MVC (Model-View-Controller) dalam pengembangan aplikasi web. Melalui praktikum ini, mahasiswa diharapkan dapat mengenal cara kerja Laravel, memahami fungsi routing, controller, dan view, serta mampu membangun aplikasi web sederhana secara terstruktur dan sistematis menggunakan framework Laravel

BAB II

HASIL

A. GUIDED (Praktikum Terbimbing)

ROUTE / ROUTING

a. Membuat route

```php
// Membuat route
Route::get('/beranda', function () {
    return "Halaman Beranda";
});
```

b. Route parameter

```php
// Route parameter
Route::get('/kendaraan/{jenis}', function ($jenis) {
    return "Tampilkan data kendaraan dengan jenis $jenis";
});
```

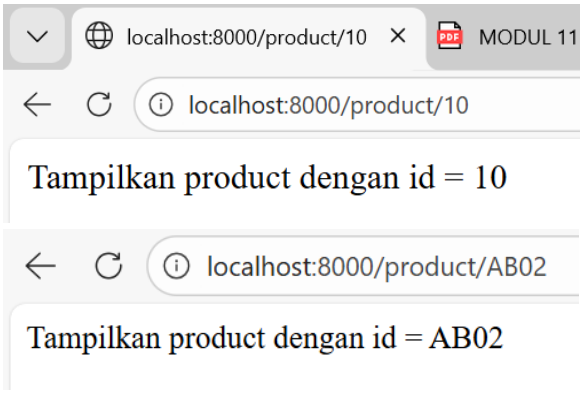
c. Route dengan optional parameter

```php
// Route dengan optional parameter
Route::get('/kendaraan/{jenis?}/{merek?}', function ($a = 'motor',$b = 'honda') {
    return "Cek harga kendaraan $a $b";
});
```

d. Route parameter dengan regular expression

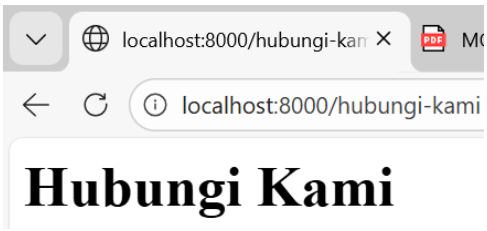
```php
// Route parameter dengan regular expression
Route::get('/product/{id}', function ($id) {
    return "Tampilkan product dengan id = $id";
});
```

```php
Route::get('/product/{id}', function ($id) {
    return "Tampilkan product dengan id = $id";
})->where('id', '[0-9]+');
```



e. Route redirect

```php
// Route redirect
Route::get('/hubungi-kami', function () {
    return '<h1>Hubungi Kami</h1>';
});
Route::redirect('/contact-us', '/hubungi-kami');
```



f. Route group

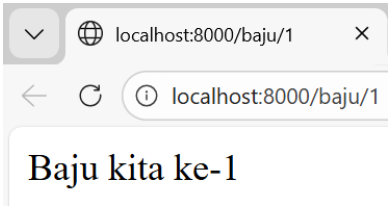
```php
// Route group
Route::prefix('/admin')->group(function() {
    Route::get('/dashboard', function() {
        return 'Tampilkan dashboard aplikasi';
    });
    Route::get('/datapegawai', function() {
        return 'Tampilkan data pegawai';
    });
    Route::get('/datamahasiswa', function() {
        return 'Tampilkan data mahasiswa';
    });
});
```

g. Route fallback

```php
// Route fallback
Route::fallback(function () {
    return "Maaf, alamat tidak ditemukan";
});
```

h. Route priority

```php
// Route priority
Route::get('/baju/1', function () {
    return "Baju ke-1";
});
Route::get('/baju/1', function () {
    return "Baju saya ke-1";
});
Route::get('/baju/1', function () {
    return "Baju kita ke-1";
});
```



Penjelasan:

Kode di atas merupakan contoh penggunaan routing pada Laravel yang diletakkan di dalam file routes/web.php. Baris use Illuminate\Support\Facades\Route;

digunakan untuk memanggil facade Route agar kita dapat mendefinisikan berbagai URL (route) dalam aplikasi web.

Route pertama Route::get('/', ...) berfungsi untuk menampilkan halaman utama aplikasi dan mengembalikan tampilan (view) bernama welcome. Selanjutnya, route /beranda dibuat untuk menampilkan teks sederhana "Halaman Beranda" ketika URL tersebut diakses melalui metode GET.

Route /kendaraan/{jenis} merupakan route parameter, di mana nilai {jenis} diambil dari URL dan dikirim ke fungsi sebagai variabel, lalu ditampilkan dalam bentuk teks. Kemudian, route /kendaraan/{jenis?}/{merek?} menggunakan optional parameter, artinya parameter boleh dikosongkan. Jika tidak diisi, maka Laravel akan menggunakan nilai default yaitu motor dan honda.

Pada route /product/{id}, parameter id digunakan untuk menampilkan data produk berdasarkan ID. Versi kedua dari route ini menambahkan validasi menggunakan regular expression melalui method where(), sehingga hanya angka yang diperbolehkan sebagai nilai id.

Route /hubungi-kami menampilkan halaman sederhana berupa teks HTML, sedangkan Route::redirect('/contact-us', '/hubungi-kami') berfungsi untuk mengalihkan (redirect) pengguna dari URL /contact-us ke /hubungi-kami.

Bagian route group dengan prefix /admin digunakan untuk mengelompokkan beberapa route yang memiliki awalan URL yang sama. Dengan demikian, route seperti /admin/dashboard, /admin/datapegawai, dan /admin/datamahasiswa dapat dikelola dengan lebih rapi dan terstruktur.

Route fallback digunakan sebagai penanganan jika URL tidak ditemukan, sehingga ketika pengguna mengakses alamat yang tidak terdaftar, akan muncul pesan "Maaf, alamat tidak ditemukan".

Pada bagian route priority, terdapat beberapa route dengan URL yang sama (/baju/1). Laravel akan mengeksekusi route yang didefinisikan paling atas, sehingga route pertama yang cocok akan dijalankan. Hal yang sama juga terjadi pada route dengan parameter /baju/{a}, /baju/{b}, dan /baju/{c}, di mana Laravel akan menjalankan route yang pertama kali dideklarasikan.

VIEW

a. View bawaan Laravel

b. Membuat view

```php
// membuat view
Route::get('/mahasiswa', function () {
    return view('mahasiswa');
});
```

Source code didalam file mahasiswa.blade.php

resources > views > mahasiswa.blade.php

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Belajar Laravel</title>
    </head>
    <body>
        <h2> Data Mahasiswa </h2>
        <ol>
            <li>Mahasiswa 1</li>
            <li>Mahasiswa 2</li>
            <li>Mahasiswa 3</li>
            <li>Mahasiswa 4</li>
        </ol>
    </body>
</html>
```

c. Membuat struktur folder view

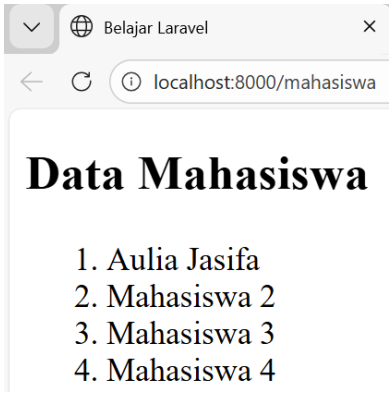
```php
Route::get('/mahasiswa', function () {
    return view('universitas/mahasiswa');
});
```

d. Mengirim data ke view

1. Mengirim Data ke View Sebagai Argumen

```php
Route::get('/mahasiswa', function () {
    return view('universitas.mahasiswa',["mhs1"=>"Aulia Jasifa"]);
});
```

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Belajar Laravel</title>
    </head>
    <body>
        <h2> Data Mahasiswa </h2>
        <ol>
            <li><?php echo $mhs1; ?></li>
            <li>Mahasiswa 2</li>
            <li>Mahasiswa 3</li>
            <li>Mahasiswa 4</li>
        </ol>
    </body>
</html>
```

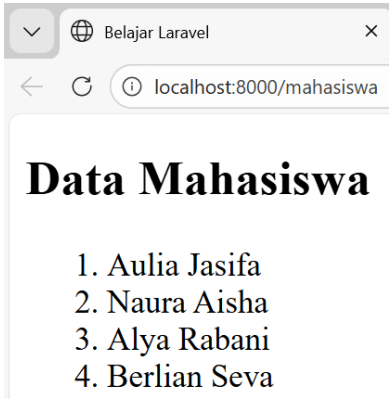


Kemudian apabila ingin mengirim lebih banyak data, tambah element baru ke dalam array yang akan dikirim, ubah route menjadi:

```php
Route::get('/mahasiswa', function () {
    return view('universitas.mahasiswa',
    [
        "mhs1" => "Aulia Jasifa",
        "mhs2" => "Naura Aisha",
        "mhs3" => "Alya Rabani",
        "mhs4" => "Berlian Seva"
    ]);
});
```

Ubah script pada file mahasiswa.blade.php

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Belajar Laravel</title>
    </head>
    <body>
        <h2> Data Mahasiswa </h2>
        <ol>
            <li><?php echo $mhs1; ?></li>
            <li><?php echo $mhs2; ?></li>
            <li><?php echo $mhs3; ?></li>
            <li><?php echo $mhs4; ?></li>
        </ol>
    </body>
</html>
```



Selain cara tersebut, agar lebih rapi, kode program di bagian route bisa ditulis sebagai berikut:

```php
Route::get('/mahasiswa', function () {
    $arrMhs = [
        "mhs1" => "Aulia Jasifa",
        "mhs2" => "Naura Aisha",
        "mhs3" => "Alya Rabani",
        "mhs4" => "Berlian Seva"
    ];
    return view('universitas.mahasiswa', $arrMhs);
});
```

Kirim data dalam bentuk nested array

```php
Route::get('/mahasiswa', function () {
    $arrMhs = ["Aulia Jasifa","Naura Aisha","Alya Rabani","Berlian Seva"];
    return view('universitas.mahasiswa',['mahasiswa' => $arrMhs]);
});
```

Ubah script pada file mahasiswa.blade.php

```php
<!DOCTYPE html>
<html>
    <head>
        <title>Belajar Laravel</title>
    </head>
    <body>
        <h2> Data Mahasiswa </h2>
        <ol>
            <li><?php echo $mahasiswa[0]; ?></li>
            <li><?php echo $mahasiswa[1]; ?></li>
            <li><?php echo $mahasiswa[2]; ?></li>
            <li><?php echo $mahasiswa[3]; ?></li>
        </ol>
    </body>
</html>
```

2. Mengirim Data ke View menggunakan method with
   Cara kedua yang bisa dipakai untuk mengirim data dari route ke view adalah menggunakan method with().

```php
Route::get('/mahasiswa', function () {
    $arrMhs = ["Aulia jasifa","Naura Aisha","Alya Rabani","Berlian Seva"];
    return view('universitas.mahasiswa')->with('mahasiswa',
$arrMhs);
});
```

BLADE TEMPLATE ENGINE

a. Menampilkan Data

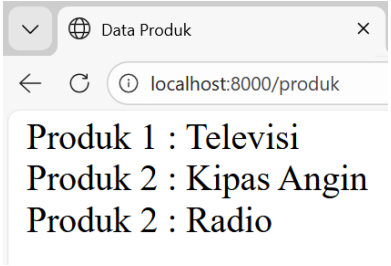
```php
Route::get('/produk', function () {
    $arrProduk = [
        "prod1" => "Televisi",
        "prod2" => "Kipas Angin",
        "prod3" => "Radio"
    ];
    return view('produk', $arrProduk);
});
```

Buat view produk.blade.php

resources > views > produk.blade.php

```blade
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Data Produk</title>
    </head>
    <body>
        Produk 1 : {{ $prod1; }}
        <br>
        Produk 2 : {{ $prod2; }}
        <br>
        Produk 2 : {{ $prod3; }}
    </body>
</html>
```

Tampilan pada browser



b. Merancang Layout

resources > views > master.blade.php

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>@yield('title')</title>
    </head>
    <body>
        @yield('content')
    </body>
</html>
```

produk.blade.php

```
@extends('master')
@section('title','Data Produk'

@section('content')
    Produk 1 : {{ $prod1; }}
    <br>
    Produk 2 : {{ $prod2; }}
    <br>
    Produk 2 : {{ $prod3; }}
@endsection
```

CONTROLLERS

a. Cara Mengakses Controller

```
Route::get('<url>',[App\Http\Controllers\Nama_Controller::class,'nama_method']);
```

Untuk memanggil method index() di controller bernama PageController

```
Route::get('/', [App\Http\Controllers\PageController::class,'index']);
```

Contoh lain

```
Route::get('/mahasiswa',[App\Http\Controllers\PageController::class,'tampil']);
```

app > Http > Controllers > Controller.php

```
<?php
namespace App\Http\Controllers;

use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Routing\Controller as BaseController;

class Controller extends BaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;
}
```

b. Membuat Controller Secara Manual

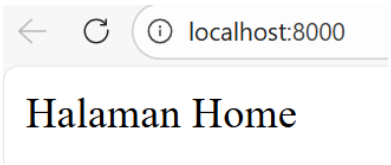
app > Http > Controllers > pageController.php

```php
<?php
namespace App\Http\Controllers;

class PageController extends Controller
{
    public function index() {
        return "Halaman Home";
    }
    public function tampil() {
        return "Data Mahasiswa";
    }
}
```

Kemudian coba akses dengan menambah dua routes berikut ke dalam file routes/web.php:

```php
Route::get('/', [App\Http\Controllers\PageController::class,'index']);
Route::get('/mahasiswa',[App\Http\Controllers\PageController::class,'tampil']);
```



c. Cara Penulisan Route Untuk Controller

routes > web.php

```php
<?php

use Illuminate\Support\Facades\Route;

Route::get('/', [App\Http\Controllers\PageController::class,'index']);
Route::get('/mahasiswa',[App\Http\Controllers\PageController::class,'tampil']);
```

Memindahkan namespace "App\Http\Controllers\PageController" ke bagian atas

routes > web.php

```php
<?php

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

Route::get('/', [App\Http\Controllers\PageController::class,'index']);
Route::get('/mahasiswa',[App\Http\Controllers\PageController::class,'tampil']);
```

d. Mengakses View dari Controller

Modifikasi kembali PageController.php

app > Http > Controllers > pageController.php

```php
<?php
namespace App\Http\Controllers;

class PageController extends Controller
{
    public function index() {
        return "Halaman Home";
    }
    public function tampil() {
        $arrMahasiswa = ["Aulia Jasifa","Naura Aisha","Alya Rabani","Berlian Seva"];
        return view('mahasiswa')->with('mahasiswa', $arrMahasiswa);
    }
}
```

Kemudian buat views dengan nama mahasiswa.php

resources > views > mahasiswa.blade.php

```blade
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Data Mahasiswa</title>
    </head>
    <body>
        <div>
            <h1>Data Mahasiswa</h1>
            <div>
                <div>
                    <ol>
                        @forelse ($mahasiswa as $val)
                        <li>{{$val}}</li>
                        @empty
                        <div>Tidak ada data...</div>
                        @endforelse
                    </ol>
                </div>
            </div>
        </div>
    </body>
</html>
```

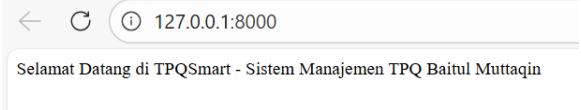
B. UNGUIDED (Tugas Mandiri)

1. TUGAS – Router

1) Buat minimal 5 route tanpa parameter

Halaman Home

```php
// Route tanpa parameter
Route::get('/', function () {
    return "Selamat Datang di TPQSmart - Sistem Manajemen TPQ Baitul Muttaqin";
});
```

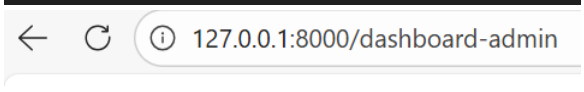


Halaman Login

```
Route::get('/login', function () {
    return "Halaman Login TPQSmart";
});
```

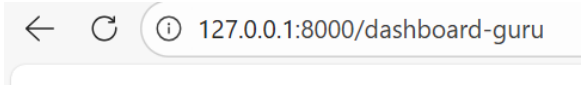
Halaman Dashboard Admin

```
Route::get('/dashboard-admin', function () {
    return "Dashboard Admin TPQ Baitul Muttaqin";
});
```



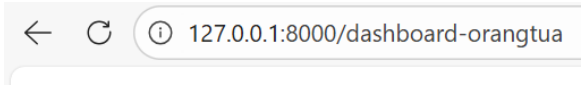
Halaman dashboard-Guru

```
Route::get('/dashboard-guru', function () {
    return "Dashboard Guru TPQ Baitul Muttaqin";
});
```



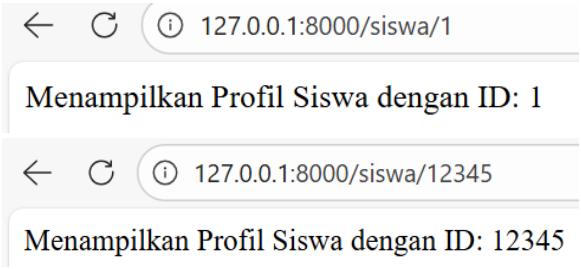
Halaman Dashboard-Orangtua

```
Route::get('/dashboard-orangtua', function () {
    return "Dashboard Orang Tua/Wali Murid";
});
```



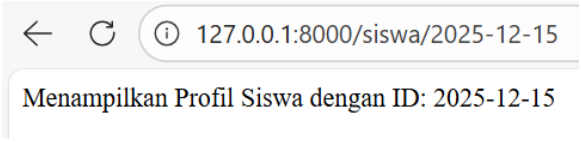
2) Buat minimal 3 route dengan parameter
Lihat Profil Siswa Berdasarkan ID

```
// Route dengan parameter
Route::get('/siswa/{id}', function ($id) {
    return "Menampilkan Profil Siswa dengan ID: $id";
});
```



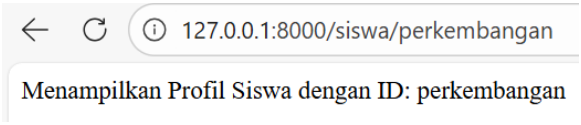
Lihat Presensi Berdasarkan Tanggal

```
Route::get('/presensi/{tanggal}', function ($tanggal) {
    return "Menampilkan Data Presensi Tanggal: $tanggal";
});
```



Detail Perkembangan Siswa

```
Route::get('/perkembangan/{id_siswa}', function ($id_siswa) {
    return "Menampilkan Perkembangan Siswa ID: $id_siswa";
});
```

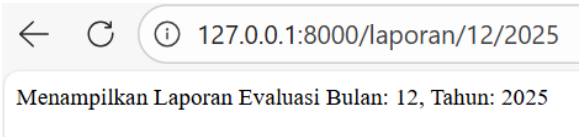


3) Buat minimal 3 route dengan optional parameter

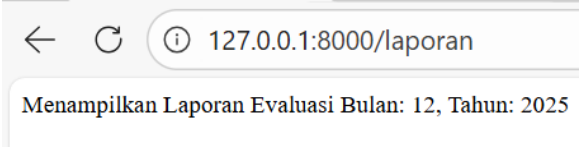
Laporan evaluasi

```
Route::get('/laporan/{bulan?}/{tahun?}', function ($bulan = null, $tahun = null) {
    $bulan = $bulan ?? date('m');
    $tahun = $tahun ?? date('Y');
    return "Menampilkan Laporan Evaluasi Bulan: $bulan, Tahun: $tahun";
});
```

Dengan parameter lengka



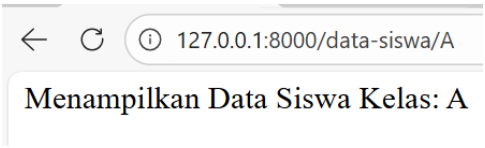
Tanpa parameter lengka



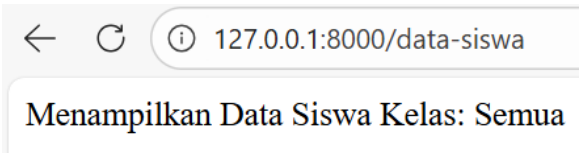
Data sisswa per kelas

```php
Route::get('/data-siswa/{kelas?}', function ($kelas = 'Semua') {
    return "Menampilkan Data Siswa Kelas: $kelas";
});
```

Dengan parameter lengka



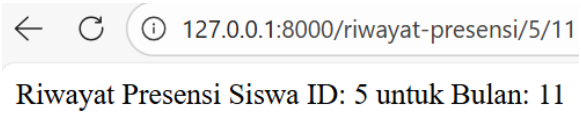
Tanpa parameter lengka



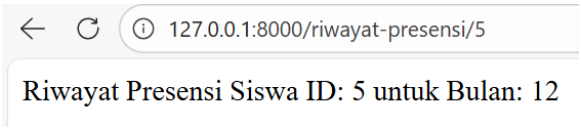
Riwayat presensi

```php
Route::get('/riwayat-presensi/{id_siswa}/{bulan?}', function ($id_siswa, $bulan = null) {
    $bulan = $bulan ?? date('m');
    return "Riwayat Presensi Siswa ID: $id_siswa untuk Bulan: $bulan";
});
```

Dengan parameter lengka



Tanpa parameter lengka



Penjelasan:

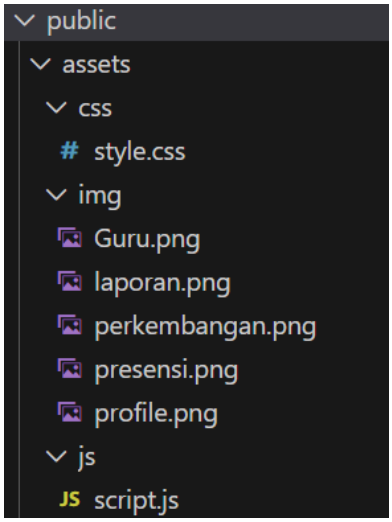
Routing dalam Laravel berfungsi sebagai pengatur lalu lintas aplikasi yang menentukan bagaimana aplikasi merespon permintaan HTTP dari user. Dalam tugas ini, dibuat tiga jenis route yaitu route tanpa parameter, route dengan parameter, dan route dengan optional parameter. Route tanpa parameter seperti /login atau /dashboard-guru digunakan untuk halaman statis yang tidak memerlukan input dinamis, ketika user mengakses URL tersebut, Laravel akan mencocokkan dengan route yang terdaftar di routes/web.php dan menjalankan callback function yang mengembalikan response berupa string atau view. Route dengan parameter seperti /siswa/{id} memungkinkan aplikasi menerima data dinamis melalui URL, dimana nilai parameter akan ditangkap dan dapat diproses lebih lanjut, misalnya untuk menampilkan profil siswa berdasarkan ID tertentu. Sedangkan route dengan optional parameter

seperti /laporan/{bulan?}/{tahun?} memberikan fleksibilitas dimana parameter bersifat opsional dengan memberikan tanda tanya setelah nama parameter, jika user tidak memberikan nilai maka sistem akan menggunakan nilai default yang telah ditentukan dalam callback function. Ketiga jenis route ini bekerja dengan cara yang sama yaitu melalui proses route matching dimana Laravel membandingkan URL yang diakses dengan pola route yang terdaftar, kemudian menjalankan logic yang sesuai dan mengembalikan response ke browser user.

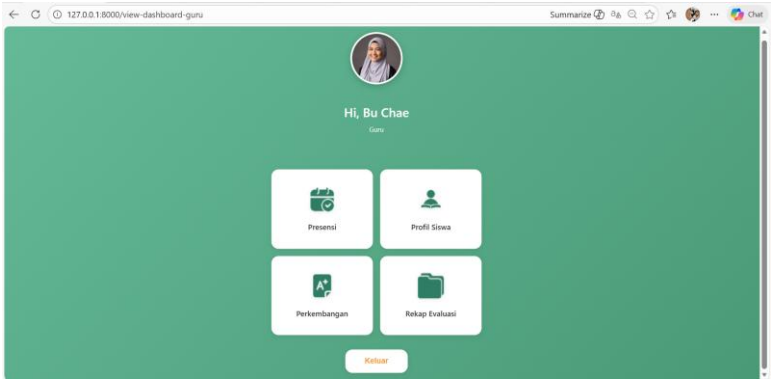
2. TUGAS – View

Pada pembelajaran PHP dasar kita pasti sudah mempelajari pengelolaan asset. Pada tugas kali ini terapkan pengelolaan asset pada framework Laravel.

1) Buat folder asset yang didalamnya birisi folder css, js, img.



2) Buat view yang menampilkan gambar yang terletak di folder img



3) Buat view yang mengakses css dan javascript dari folder css dan js.

```php
// Route untuk menampilkan view dashboard guru
Route::get('/view-dashboard-guru', function () {
    return view('dashboard-guru');
});
```

```php
// Route untuk view presensi siswa
Route::get('/view-presensi', function () {
    return view('presensi-siswa');
});
```

Class public/assets/css/style.css

```css
/* CSS TPQSMART */

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #66BB99 0%, #4A9976 100%);
    min-height: 100vh;
    display: flex;
    align-items: center;
    justify-content: center;
    padding: 20px;
}

.container {
    max-width: 400px;
    width: 100%;
    background: transparent;
    padding: 0;
}

header {
    text-align: center;
    padding: 40px 20px;
    color: white;
}
```

```
header img {
    width: 100px;
    height: 100px;
    border-radius: 50%;
    border: 4px solid white;
    margin-bottom: 20px;
    object-fit: cover;
    box-shadow: 0 4px 15px rgba(0,0,0,0.2);
}

header h1 {
    color: white;
    font-size: 24px;
    margin: 15px 0 10px 0;
    font-weight: 600;
}

header p {
    color: white;
    margin: 5px 0;
    font-size: 14px;
    opacity: 0.95;
}

.menu-grid {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 15px;
    padding: 0 20px;
    margin-top: 30px;
}

.menu-item {
    background: white;
    padding: 30px 20px;
    border-radius: 15px;
    text-align: center;
    cursor: pointer;
    transition: all 0.3s ease;
```

```
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
}

.menu-item:hover {
    transform: translateY(-5px);
    box-shadow: 0 5px 20px rgba(0,0,0,0.15);
}

.menu-item img {
    width: 50px;
    height: 50px;
    margin-bottom: 15px;
    filter: drop-shadow(0 2px 4px rgba(0,0,0,0.1));
}

.menu-item h3 {
    margin: 0;
    font-size: 15px;
    color: #333;
    font-weight: 500;
}

.logout-container {
    text-align: center;
    margin-top: 30px;
    padding: 0 20px;
}

.btn-logout {
    background: white;
    color: #F59E42;
    padding: 15px 40px;
    border: none;
    border-radius: 15px;
    cursor: pointer;
    font-size: 16px;
    font-weight: 600;
    box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    transition: all 0.3s ease;
```

```
    display: inline-flex;
    align-items: center;
    justify-content: center;
    gap: 10px;
}

.btn-logout:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 15px rgba(0,0,0,0.15);
}

.btn-logout::before {
    font-size: 20px;
}

/* Responsive Design untuk Laptop */
@media (min-width: 768px) {
    .container {
        max-width: 450px;
    }
}

@media (min-width: 1024px) {
    .container {
        max-width: 480px;
    }

    header img {
        width: 110px;
        height: 110px;
    }

    header h1 {
        font-size: 26px;
    }

    .menu-item {
        padding: 35px 25px;
    }
```

```
    .menu-item img {
        width: 55px;
        height: 55px;
    }

    .menu-item h3 {
        font-size: 16px;
    }
}
```

Class public/assets/js/sripct.js

```
// JS TPQSMART

console.log('TPQSmart JavaScript Loaded Successfully!');

// Fungsi untuk menampilkan pesan selamat datang
function tampilkanPesan() {
    const waktu = new Date().getHours();
    let salam;

    if (waktu < 11) {
        salam = "Selamat Pagi";
    } else if (waktu < 15) {
        salam = "Selamat Siang";
    } else if (waktu < 19) {
        salam = "Selamat Sore";
    } else {
        salam = "Selamat Malam";
    }

    return salam;
}

// Jalankan saat halaman dimuat
document.addEventListener('DOMContentLoaded', function() {
    console.log('Halaman TPQSmart siap digunakan!');
    console.log(tampilkanPesan());
```

```
    // Tambahkan interaksi pada menu
    const menuItems = document.querySelectorAll('.menu-item');
    menuItems.forEach(function(item) {
        item.addEventListener('click', function() {
            const menuName = this.querySelector('h3').textContent;
            alert('Anda mengklik menu: ' + menuName);
        });
    });
});

// Fungsi untuk logout
function handleLogout() {
    if (confirm('Apakah Anda yakin ingin keluar?')) {
        alert('Anda telah logout dari sistem TPQSmart');
        window.location.href = '/login';
    }
}
```

Class view/dashboard-guru.blade.php

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Dashboard Guru - TPQSmart</title>
    <link rel="stylesheet" href="{{ asset('assets/css/style.css') }}">
</head>
<body>
    <div class="container">
        <header>
            <!-- Menampilkan gambar dari folder img -->
            <img src="{{ asset('assets/img/Guru.png') }}" alt="Guru TPQ Baitul
Muttaqin">
            <h1>Hi, Bu Chae</h1>
            <p>Guru</p>
        </header>
```

```
        <div class="menu-grid">
            <!-- Menu 1: Presensi -->
            <div class="menu-item">
                <img src="{{ asset('assets/img/presensi.png') }}" alt="Icon
Presensi">
                <h3>Presensi</h3>
            </div>

            <!-- Menu 2: Profil Siswa -->
            <div class="menu-item">
                <img src="{{ asset('assets/img/profile.png') }}" alt="Icon Profil">
                <h3>Profil Siswa</h3>
            </div>

            <!-- Menu 3: Perkembangan -->
            <div class="menu-item">
                <img src="{{ asset('assets/img/perkembangan.png') }}" alt="Icon
Perkembangan">
                <h3>Perkembangan</h3>
            </div>

            <!-- Menu 4: Rekap Evaluasi -->
            <div class="menu-item">
                <img src="{{ asset('assets/img/laporan.png') }}" alt="Icon
Laporan">
                <h3>Rekap Evaluasi</h3>
            </div>
        </div>

        <div class="logout-container">
            <button class="btn-logout"
onclick="handleLogout()">Keluar</button>
        </div>
    </div>

    <!-- Mengakses JavaScript dari folder js -->
    <script src="{{ asset('assets/js/script.js') }}"></script>
</body>
```

```
</html>
```

Class presense-siswa.blade.php

```
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Presensi Siswa - TPQSmart</title>
    <link rel="stylesheet" href="{{ asset('assets/css/style.css') }}">
    <style>
        .table-container {
            margin-top: 30px;
        }

        table {
            width: 100%;
            border-collapse: collapse;
        }

        table th, table td {
            padding: 12px;
            text-align: left;
            border-bottom: 1px solid #ddd;
        }

        table th {
            background-color: #83D1A7;
            color: white;
        }

        table tr:hover {
            background-color: #f5f5f5;
        }

        .btn-simpan {
            background: #83D1A7;
```

```
            color: white;
            padding: 12px 30px;
            border: none;
            border-radius: 5px;
            margin-top: 20px;
            cursor: pointer;
            font-size: 16px;
        }

        .btn-simpan:hover {
            background: #83D1A7;
        }
    </style>
</head>
<body>
    <div class="container">
        <header>
            <img src="{{ asset('assets/img/TPQSmart Logo.png') }}" alt="Logo
TPQ">
            <h1>Form Presensi Siswa</h1>
            <p>Kelas A - Tanggal: <span id="tanggal-hari-ini"></span></p>
        </header>

        <div class="table-container">
            <table id="tabel-presensi">
                <thead>
                    <tr>
                        <th>No</th>
                        <th>Nama Siswa</th>
                        <th>Status</th>
                    </tr>
                </thead>
                <tbody>
                    <tr>
                        <td>1</td>
                        <td>Ahmad Fauzi</td>
                        <td>
                            <select>
                                <option value="hadir">Hadir</option>
```

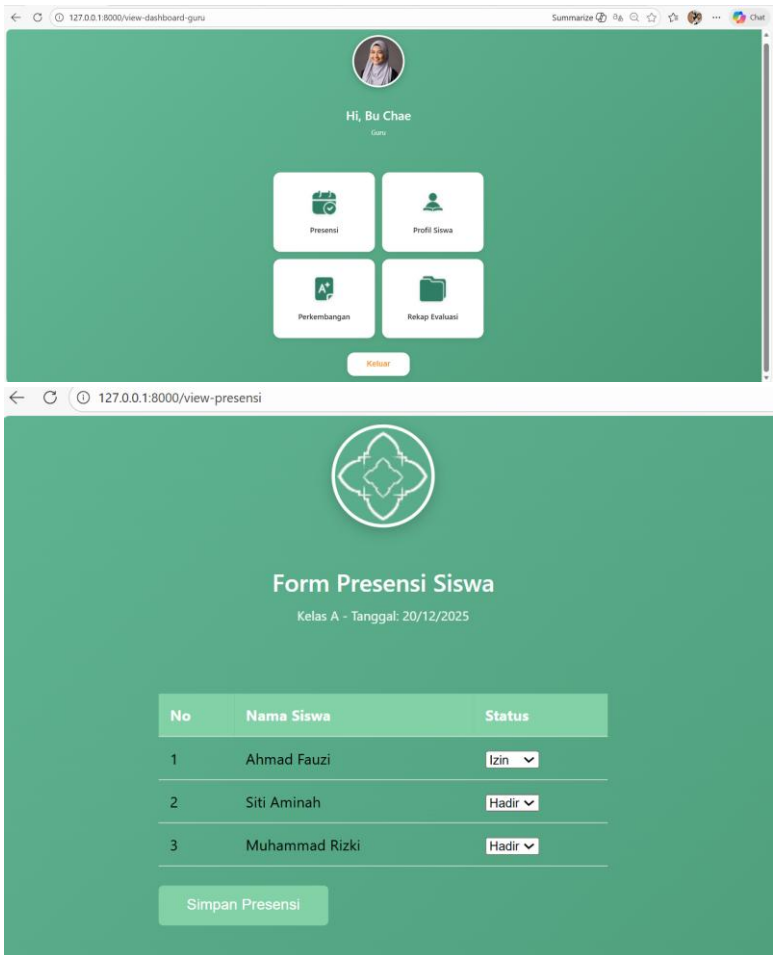
```
                    <option value="izin">Izin</option>
                    <option value="sakit">Sakit</option>
                    <option value="alpa">Alpa</option>
                  </select>
                </td>
              </tr>
              <tr>
                <td>2</td>
                <td>Siti Aminah</td>
                <td>
                  <select>
                    <option value="hadir">Hadir</option>
                    <option value="izin">Izin</option>
                    <option value="sakit">Sakit</option>
                    <option value="alpa">Alpa</option>
                  </select>
                </td>
              </tr>
              <tr>
                <td>3</td>
                <td>Muhammad Rizki</td>
                <td>
                  <select>
                    <option value="hadir">Hadir</option>
                    <option value="izin">Izin</option>
                    <option value="sakit">Sakit</option>
                    <option value="alpa">Alpa</option>
                  </select>
                </td>
              </tr>
            </tbody>
          </table>

          <button class="btn-simpan" onclick="simpanPresensi()">Simpan
Presensi</button>
        </div>
      </div>

      <script src="{{ asset('assets/js/script.js') }}"></script>
```

```
  <script>
    // Tampilkan tanggal hari ini
    document.getElementById('tanggal-hari-ini').textContent = new
Date().toLocaleDateString('id-ID');

    // Fungsi simpan presensi
    function simpanPresensi() {
      alert('Data presensi berhasil disimpan!\n\nNotifikasi WhatsApp akan
dikirim ke orang tua/wali.');
      console.log('Presensi disimpan pada:', new Date());
    }
  </script>
</body>
</html>
```



Penjelasan

Kode ini merupakan halaman web untuk sistem presensi siswa TPQ (Taman Pendidikan Al-Quran) yang diberi nama TPQSmart. Fungsi utamanya adalah menyediakan formulir digital bagi guru untuk mencatat kehadiran siswa di Kelas A. Halaman ini menampilkan header dengan logo TPQ, judul form, dan tanggal otomatis yang diambil dari sistem menggunakan JavaScript. Bagian utama berisi tabel presensi yang memuat daftar nama siswa beserta dropdown menu untuk memilih status kehadiran (Hadir, Izin, Sakit, atau Alpa). Setiap baris tabel mewakili satu siswa dengan nomor urut, nama, dan pilihan status kehadiran.

Cara kerja sistem ini cukup sederhana: ketika halaman dibuka, JavaScript secara otomatis menampilkan tanggal hari ini dalam format Indonesia di bagian header. Guru dapat memilih status kehadiran untuk setiap siswa melalui dropdown yang tersedia. Setelah semua status dipilih, guru menekan tombol "Simpan Presensi" yang akan memicu fungsi JavaScript simpanPresensi(). Fungsi ini menampilkan alert konfirmasi bahwa data berhasil disimpan dan memberitahu bahwa notifikasi WhatsApp akan dikirim ke orang tua/wali siswa. Sistem juga mencatat waktu penyimpanan data di console browser untuk keperluan logging. Styling halaman menggunakan kombinasi CSS eksternal dan inline CSS untuk tampilan tabel yang rapi dengan efek hover dan warna tema hijau tosca yang konsisten dengan identitas TPQSmart.

3. TUGAS - BLADE TEMPLATE ENGINE

Kerjakan tugas berikut menggunakan blade

1) Buat perulangan for untuk menampilkan bilangan 1 s.d 10

```
// Route untuk perulangan FOR
Route::get('/blade-for', function () {
    return view('blade-for');
});
```

```blade
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <title>Blade For Loop </title>
</head>
<body>
    <div class="container">
        <header>
            <h1>Latihan Blade: Perulangan FOR</h1>
            <p>Menampilkan bilangan 1 sampai 10</p>
        </header>

        <div style="background: #f5f5f5; padding: 20px; border-radius: 10px;">
            <h3>Hasil Perulangan FOR:</h3>
            <ul style="font-size: 18px; line-height: 2;">
                @for ($i = 1; $i <= 10; $i++)
                    <li>Bilangan ke-{{ $i }}</li>
                @endfor
            </ul>
        </div>
    </div>
</body>
</html>
```

2) Buat perulangan while untuk menampilkan bilangan 1 s.d 10

```php
// Route untuk perulangan WHILE
Route::get('/blade-while', function () {
    return view('blade-while');
});
```

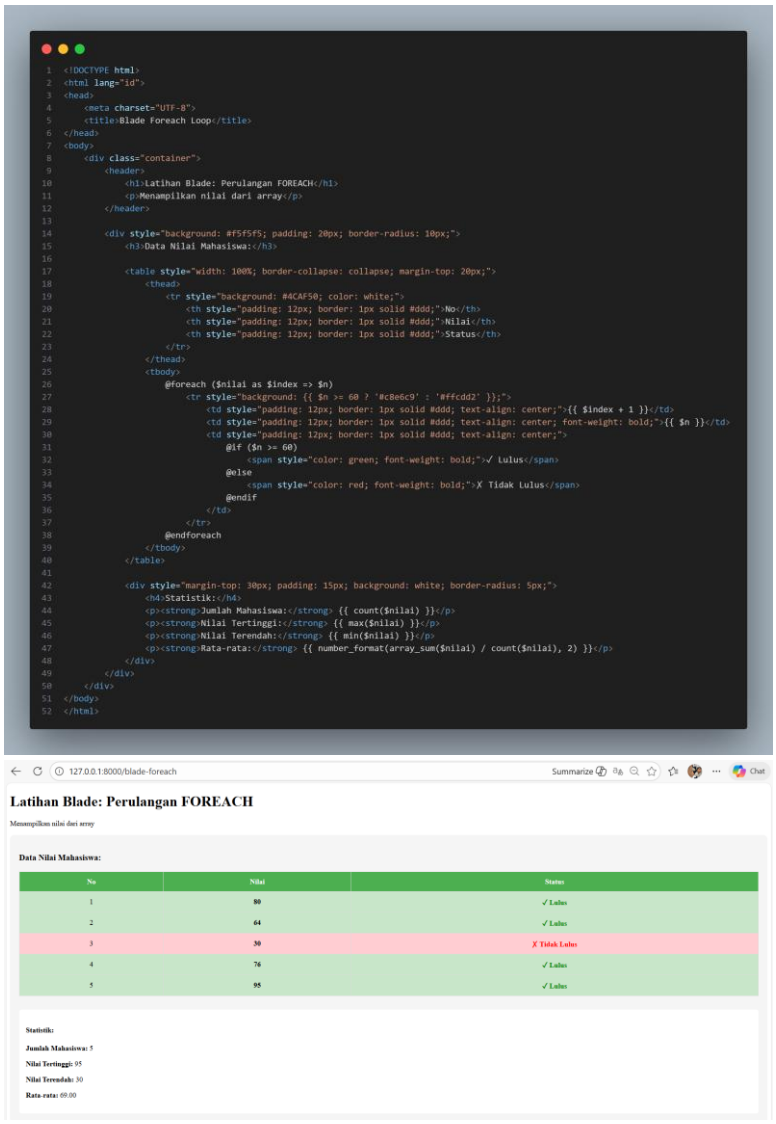
```blade
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <title>Blade While Loop</title>
</head>
<body>
    <div class="container">
        <header>
            <h1>Latihan Blade: Perulangan WHILE</h1>
            <p>Menampilkan bilangan 1 sampai 10</p>
        </header>

        <div style="background: #f5f5f5; padding: 20px; border-radius: 10px;">
            <h3>Hasil Perulangan WHILE:</h3>
            <ul style="font-size: 18px; line-height: 2;">
                @php
                    $angka = 1;
                @endphp

                @while ($angka <= 10)
                    <li>Bilangan ke-{{ $angka }}</li>
                    @php
                        $angka++;
                    @endphp
                @endwhile
            </ul>
        </div>
    </div>
</body>
</html>
```

3) Buat perulangan foreach untuk menampilkan nilai pada route berikut

```php
// Route untuk perulangan FOREACH
Route::get('/blade-foreach', function () {
    $nilai = [80, 64, 30, 76, 95];
    return view('blade-foreach', ['nilai' => $nilai]);
});
```



```blade
<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8">
    <title>Blade Foreach Loop</title>
</head>
<body>
    <div class="container">
        <header>
            <h1>Latihan Blade: Perulangan FOREACH</h1>
            <p>Menampilkan nilai dari array</p>
        </header>

        <div style="background: #f5f5f5; padding: 20px; border-radius: 10px;">
            <h3>Data Nilai Mahasiswa:</h3>

            <table style="width: 100%; border-collapse: collapse; margin-top: 20px;">
                <thead>
                    <tr style="background: #4CAF50; color: white;">
                        <th style="padding: 12px; border: 1px solid #ddd;">No</th>
                        <th style="padding: 12px; border: 1px solid #ddd;">Nilai</th>
                        <th style="padding: 12px; border: 1px solid #ddd;">Status</th>
                    </tr>
                </thead>
                <tbody>
                    @foreach ($nilai as $index => $n)
                        <tr style="background: {{ $n >= 60 ? '#c8e6c9' : '#ffcdd2' }};">
                            <td style="padding: 12px; border: 1px solid #ddd; text-align: center;">{{ $index + 1 }}</td>
                            <td style="padding: 12px; border: 1px solid #ddd; text-align: center; font-weight: bold;">{{ $n }}</td>
                            <td style="padding: 12px; border: 1px solid #ddd; text-align: center;">
                                @if ($n >= 60)
                                    <span style="color: green; font-weight: bold;">✓ Lulus</span>
                                @else
                                    <span style="color: red; font-weight: bold;">✗ Tidak Lulus</span>
                                @endif
                            </td>
                        </tr>
                    @endforeach
                </tbody>
            </table>

            <div style="margin-top: 30px; padding: 15px; background: white; border-radius: 5px;">
                <h4>Statistik:</h4>
                <p><strong>Jumlah Mahasiswa:</strong> {{ count($nilai) }}</p>
                <p><strong>Nilai Tertinggi:</strong> {{ max($nilai) }}</p>
                <p><strong>Nilai Terendah:</strong> {{ min($nilai) }}</p>
                <p><strong>Rata-rata:</strong> {{ number_format(array_sum($nilai) / count($nilai), 2) }}</p>
            </div>
        </div>
    </div>
</body>
</html>
```


Penjelasan:

Blade template engine adalah mesin template bawaan Laravel yang menyediakan sintaks yang lebih bersih dan mudah dibaca untuk operasi-operasi umum seperti perulangan dan kondisional, menggantikan sintaks PHP standar yang verbose dengan directive yang lebih elegan. Dalam tugas ini, dibuat tiga view yang mendemonstrasikan penggunaan directive Blade untuk perulangan: blade-for.blade.php menggunakan directive @for($i = 1; $i <= 10; $i++) untuk menampilkan bilangan 1 sampai 10 dengan mekanisme counter yang eksplisit, blade-while.blade.php menggunakan directive @while($angka <= 10) dengan inisialisasi variabel menggunakan @php dan increment manual di dalam loop untuk mendemonstrasikan perulangan

berbasis kondisi, dan blade-foreach.blade.php menggunakan directive @foreach($nilai as $index => $n) untuk iterasi melalui array nilai mahasiswa yang dikirim dari route, lengkap dengan conditional @if untuk menentukan status kelulusan dan perhitungan statistik. Cara kerja Blade adalah dengan mengkompilasi directive-directive tersebut menjadi kode PHP native saat pertama kali view di-load, kemudian menyimpan hasil kompilasi di folder storage/framework/views untuk meningkatkan performa pada akses selanjutnya. Keuntungan utama menggunakan Blade adalah kode menjadi lebih readable karena tidak perlu menulis tag pembuka dan penutup PHP berulang-ulang, otomatis melakukan escaping untuk mencegah XSS attack saat menampilkan data menggunakan {{ }}, dan menyediakan control structures yang lebih intuitif seperti @foreach, @if, @for, dan @while yang membuat template lebih mudah dipahami bahkan oleh developer yang baru belajar Laravel.

4. TUGAS – Controller

1) Buat root yang mengarah ke controller

routes > web.php

```php
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\PageController;
use App\Http\Controllers\GuruController;
```

2) Tampilkan template pada view menggunakan controller
Class App\Http\Controllers\PageController;

app > Http > Controllers > PageController.php

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PageController extends Controller
{
    public function index(){
        return view('welcome-tpq');
    }

    public function login(){
        return view('login');
    }

    public function about(){
        $data = [
            'nama_tpq' => 'TPQ Baitul Muttaqin',
            'alamat' => 'Jl. Merdeka No. 123, Jakarta',
            'tahun_berdiri' => '2010',
            'jumlah_santri' => 150,
            'jumlah_guru' => 8
        ];

        return view('about', $data);
    }
}
```

Class App\Http\Controllers\GuruController;

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class GuruController extends Controller
{
   public function dashboard(){
      $data = [
         'nama_guru' => 'Bu Chae',
         'kelas' => 'Kelas A',
         'tahun_ajaran' => '2025-2026',
         'jumlah_siswa' => 25
      ];

      return view('dashboard-guru', $data);
   }

   public function profilSiswa($id){
      $siswa = [
         'id_siswa' => $id,
         'nama' => 'Ahmad Fauzi',
         'jenis_kelamin' => 'L',
         'tanggal_lahir' => '2010-05-15',
         'umur' => 15,
         'alamat' => 'Jl. Merdeka No. 10, Jakarta',
         'nama_wali' => 'Bapak Fauzi',
         'no_hp_wali' => '081234567890',
         'kelas' => 'A'
      ];

      return view('profil-siswa', $siswa);
   }

   public function dataSiswa($kelas = 'Semua')
   {
```

```
        // Simulasi data siswa
        $allSiswa = [
            ['id' => 1, 'nama' => 'Ahmad Fauzi', 'kelas' => 'A', 'umur' => 15],
            ['id' => 2, 'nama' => 'Siti Aminah', 'kelas' => 'A', 'umur' => 14],
            ['id' => 3, 'nama' => 'Muhammad Rizki', 'kelas' => 'B', 'umur' => 13],
            ['id' => 4, 'nama' => 'Fatimah Zahra', 'kelas' => 'B', 'umur' => 14],
            ['id' => 5, 'nama' => 'Ali Hasan', 'kelas' => 'C', 'umur' => 15],
        ];

        // Filter berdasarkan kelas jika bukan 'Semua'
        if ($kelas != 'Semua') {
            $siswa = array_filter($allSiswa, function($s) use ($kelas) {
                return $s['kelas'] == $kelas;
            });
        } else {
            $siswa = $allSiswa;
        }

        return view('data-siswa', [
            'siswa' => $siswa,
            'kelas_filter' => $kelas
        ]);
    }
}
```

Penjelasan:

Controller dalam Laravel mengimplementasikan konsep MVC dengan memisahkan logika bisnis dari route dan view, dimana route hanya bertugas memetakan URL sedangkan controller menangani pemrosesan data dan business logic. Dalam tugas ini, dibuat empat controller menggunakan perintah artisan php artisan make:controller NamaController yang menghasilkan file di app/Http/Controllers: PageController menangani halaman-halaman umum seperti beranda, login, dan about dengan method yang mengembalikan view dengan atau tanpa data menggunakan return view('nama-view', $data), GuruController menangani fitur-fitur guru seperti dashboard, profil siswa, data siswa, presensi, dan perkembangan dengan method yang menerima parameter dari URL seperti profilSiswa($id) atau dataSiswa($kelas = 'Semua') untuk menghasilkan output dinamis, AdminController mengelola fungsi administrator seperti dashboard

admin, data pengguna, laporan evaluasi dengan optional parameter, dan pengelolaan presensi sistem secara menyeluruh, serta OrangTuaController yang menyediakan fitur untuk orang tua seperti dashboard, riwayat presensi anak, melihat perkembangan, dan profil anak dengan scope data yang terbatas hanya pada anak yang diasuh. Cara kerja controller adalah ketika route dipanggil menggunakan sintaks Route::get('/url', [NamaController::class, 'namaMethod']), Laravel akan melakukan routing ke controller yang ditentukan, menjalankan method yang sesuai, method tersebut kemudian memproses data (saat ini masih berupa data dummy, nantinya akan query database melalui Model), dan mengembalikan view dengan data yang telah diproses untuk ditampilkan ke user. Struktur ini mengikuti prinsip separation of concerns dimana setiap controller fokus pada satu role atau aspek aplikasi, setiap method fokus pada satu fungsi spesifik, membuat aplikasi TPQSmart lebih modular, mudah di-maintain, scalable untuk pengembangan fitur baru, dan memudahkan implementasi authorization berbasis role untuk memastikan setiap user hanya mengakses fitur sesuai dengan perannya dalam sistem.

BAB III

PENUTUP

A. Kesimpulan

Dari praktikum ini, dipelajari konsep fundamental Laravel dalam pengembangan aplikasi web berbasis MVC, termasuk routing yang mengatur request HTTP dengan URL statis, parameter dinamis, dan opsional untuk fleksibilitas; pengelolaan asset melalui struktur folder terorganisir dan helper asset() untuk akses resource; Blade template engine yang menyederhanakan view dengan directive seperti @for dan @if; serta controller yang memisahkan logika bisnis dan mengirim data efisien, semuanya terintegrasi dalam aplikasi TPQSmart yang terstruktur dan mudah dikembangkan. Kendala seperti kebingungan route parameter diatasi dengan eksperimen URL, error path asset diperbaiki dengan penempatan di folder public dan helper asset(), sintaks Blade dipahami via dokumentasi dan contoh sederhana, struktur controller diselesaikan dengan mengikuti sintaks Laravel 8+, serta masalah data tidak tampil di-debug menggunakan dd() untuk memastikan konsistensi variabel.