

Laporan Modul 4: Laravel Blade Template Engine

Mata Kuliah: Workshop Web Lanjut

Nama: Deva Risny *NIM:* 2024573010060 *Kelas:* TI-2C

Abstrak

- Modul ini membahas penerapan Blade Template Engine dan Controller pada framework Laravel 12 melalui serangkaian praktikum. Praktikum ini dilakukan meliputi pengiriman data dari controller ke view, penggunaan struktur logika dalam Blade, serta pembuatan layout dengan bantuan Bootstrap. Melalui praktikum ini, mempelajari bagaimana mengatur logika aplikasi di controller dan menampilkan data secara dinamis di tampilan view menggunakan Blade, sehingga pengembangan web menjadi lebih terstruktur dan efisien.
-

1. Dasar Teori

1.1 Pengertian Controller Controller bertugas sebagai penghubung antara Model dan View. Controller mendapatkan input dari pengguna melalui HTTP Request, lalu memproses input yang diterima dengan berinteraksi dengan Model. Terakhir mengembalikan hasilnya pada View untuk ditampilkan ke pengguna. Arsitektur MVC memisahkan antara logika, tampilan, serta kontroler membuat mudah dalam pemeliharaan dan pengembangan aplikasi secara menyeluruh. Controller biasanya digunakan untuk mengatur logika bisnis agar kode lebih terstruktur dan mudah dikelola.

1.2 Pengertian Blade Blade adalah mesin templating Laravel yang ringan, tapi tetap kuat. Blade memungkinkan pengembang untuk menggunakan sintaks sederhana untuk membuat tampilan dinamis dan dapat digunakan kembali. Jadi kita tidak perlu menulis kode berulang kali untuk membuat sebuah tampilan yang berulang seperti header, navbar, dan footer.

1.3 Konsep Layout dan Komponen Dalam Laravel, layout digunakan untuk mengatur kerangka dasar halaman sehingga kode yang berulang, seperti header, footer, dan navigasi, tidak perlu ditulis berulang kali di setiap file View. Layout disimpan dalam folder resources/views dan biasanya menggunakan file Blade.

Komponen dalam Laravel Blade adalah potongan kode tampilan view bersifat modular dan dapat digunakan kembali. Komponen biasanya berisi bagian UI yang sama di banyak halaman, seperti kartu, tombol, navigasi, atau footer. Dengan menggunakan komponen, kode HTML menjadi lebih rapi dan mudah dikelola karena setiap elemen yang berulang disimpan dalam satu file terpisah dan bisa dipanggil di berbagai view menggunakan sintaksnya sendiri

Templat parsial adalah salah satu fitur Laravel yang dirancang untuk membantu pengembang menjaga kebersihan kode, meningkatkan efisiensi, dan meningkatkan keterbacaan

2. Langkah-Langkah Praktikum

2.1 Praktikum 1 – Meneruskan Data dari Controller ke Blade View

1. Membuat project laravel baru menggunakan perintah "Laravel new modul-4-blade-view" kemudian masuk ke direktori project dengan perintah "cd modul-4-blade-view".
2. Buat Controller untuk menangani route dan logika, dengan perintah php artisan make:controller DasarBladeController.
3. Tambahkan route pada routes/web.php. Editkan di routes/web.php menjadi seperti pada gambar berikut.

```
<?php

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

Route::get('/admin', [PageController::class, 'admin']);
Route::get('/user', [PageController::class, 'user']);

Route::get('/logic', [LogicController::class, 'show']);
Route::get('/dasar', [DasarBladeController::class, 'showData']);
Route::get('/', function () {
    return view('welcome');
});
```

4. Buat Metode untuk menghandle data pada Controller Buka app/Http/Controllers/DasarBladeController.php dan tambahkan metode berikut:

```
<?php

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

class DasarBladeController extends Controller
{
    public function showData()
    {
        $name = 'Devi';
        $fruits = ['Apple', 'Banana', 'Cherry'];
        $user = [
            'name' => 'Eva',
            'email' => 'eva@ilmudata.id',
            'is_active' => true,
        ];
        $product = (object)[
            'id' => 1,
            'name' => 'Laptop',
            'price' => 12000000
        ];
        return view('dasar', compact('name', 'fruits', 'user', 'product'));
    }
}
```

5. Buat Blade View Buat file baru di resources/views/dasar.blade.php:

```
<!DOCTYPE html>
<html>
<head>
    <title>Data Passing Demo</title>
</head>
<body>
    <h1>Passing Data to Blade View</h1>
    <h2>String</h2>
    <p>Name: {{ $name }}</p>
    <h2>Array</h2>
    <ul>
        @foreach ($fruits as $fruit)
            <li>{{ $fruit }}</li>
        @endforeach
    </ul>
    <h2>Associative Array</h2>
    <p>Name: {{ $user['name'] }}</p>
    <p>Email: {{ $user['email'] }}</p>
    <p>Status: {{ $user['is_active'] ? 'Active' : 'Inactive' }}</p>
    <h2>Object</h2>
    <p>ID: {{ $product->id }}</p>
    <p>Product: {{ $product->name }}</p>
    <p>Price: Rp{{ number_format($product->price, 0, ',', '.') }}</p>
</body>
</html>
```

6. Menjalankan aplikasi dan menampilkan hasil di browser. <http://127.0.0.1:8000/dasar>

Passing Data to Blade View

String

Name: Devi

Array

- Apple
- Banana
- Cherry

Associative Array

Name: Eva

Email: eva@ilmudata..id

Status: Active

Object

ID: 1

Product: Laptop

Price: Rp12.000.000

2.2 Praktikum 2 – Menggunakan Struktur Kontrol Blade

1. Buat Controller baru Di dalam project modul-4-blade-view buatlah sebuah controller baru: php artisan make:controller LogicController Ini membuat app/Http/Controllers/LogicController.php

2. Tambahkan route baru dalam web.php. Editkan di routes/web.php menjadi seperti pada gambar berikut.

```
<?php

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

Route::get('/admin', [PageController::class, 'admin']);
Route::get('/user', [PageController::class, 'user']);

Route::get('/logic', [LogicController::class, 'show']);
Route::get('/dasar', [DasarBladeController::class, 'showData']);
Route::get('/', function () {
    return view('welcome');
});
```

3. Tambahkan Logika di Controller. Sekarang kita akan menambahkan logika ke metode show. Edit app/Http/Controllers/LogicController.php:

```
<?php

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

class LogicController extends Controller
{
    public function show()
    {
        $isLoggedIn = true;
        $users = [
            ['name' => 'Marcel', 'role' => 'admin'],
            ['name' => 'Thariq', 'role' => 'editor'],
            ['name' => 'Ellian', 'role' => 'subscriber'],
        ];
        $products = []; // Simulasi array kosong untuk @forelse
        $profile = [
            'name' => 'Thariq',
            'email' => 'thariq@ilmudata.id'
        ];
        $status = 'active';
        return view('logic', compact('isLoggedIn', 'users', 'products', 'profile', 'status'));
    }
}
```

4. Buat Blade View. Buat file view di resources/views/logic.blade.php:

```
<!DOCTYPE html>
<html>
<head>
    <title>Blade Logic Demo</title>
</head>
<body>
    <h1>Blade Control Structures Demo</h1>

    <h2>1. @@if / @@foreach</h2>
    <ul>
        @foreach ($users as $user)
            <li>{{ $user['name'] }} - Role: {{ $user['role'] }}</li>
        @endforeach
    </ul>

    <h2>3. @@forelse</h2>
    @forelse ($products as $product)
        <p>{{ $product }}</p>
    @empty
        <p>No products found.</p>
    @endforelse

    <h2>4. @@isset</h2>
    @isset($profile['email'])
        <p>User Email: {{ $profile['email'] }}</p>
    @endisset

    <h2>5. @@empty</h2>
    @empty($profile['phone'])
        <p>No phone number available.</p>
    @endempty

    <h2>6. @@switch</h2>
    @switch($status)
        @case('active')
            <p>Status: Active</p>
            @break
        @case('inactive')
            <p>Status: Inactive</p>
            @break
        @default
            <p>Status: Unknown</p>
    @endswitch
</body>
</html>
```

5. Jalankan aplikasi <http://127.0.0.1:8000/logic>

Blade Control Structures Demo

1. @if / @foreach

- Marcel - Role: admin
- Thariq - Role: editor
- Ellian - Role: subscriber

3. @forelse

No products found.

4. @isset

User Email: thariq@ilmudata.id

5. @empty

No phone number available.

6. @switch

Status: Active

2.3 Praktikum 3 – Layout dan Personalisasi di Laravel 12 dengan Bootstrap

1. Buat Controller baru dengan perintah : `php artisan make:controller PageController`. Anda akan menemukan controller baru di `app/Http/Controllers/PageController.php`.

2. Menambahkan Route Buka routes/web.php dan tambahkan rute baru:

```
<?php

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

Route::get('/admin', [PageController::class, 'admin']);
Route::get('/user', [PageController::class, 'user']);

Route::get('/logic', [LogicController::class, 'show']);
Route::get('/dasar', [DasarBladeController::class, 'showData']);
Route::get('/', function () {
    return view('welcome');
});
```

3. Update Controller Di app/Http/Controllers/PageController.php isikan kode berikut:

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class PageController extends Controller
{
    //
    public function admin()
    {
        $role = 'admin';
        $username = 'Yamato Admin';
        return view('admin.dashboard', compact('role', 'username'));
    }

    public function user()
    {
        $role = 'user';
        $username = 'Liu User';
        return view('user.dashboard', compact('role', 'username'));
    }
}
```

4. Buat Layout Dasar dengan Bootstrap Kemudian, buat resources/views/layouts/app.blade.php dan isikan kode berikut:

```
<!DOCTYPE html>
<html>
<head>
    <title>@yield('title') | Layout and Personalization</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark mb-4">
        <div class="container">
            <a class="navbar-brand" href="#">Layout and Personalization</a>
            <div class="collapse navbar-collapse">
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item">
                        <span class="nav-link active">Welcome, {{ $username }}</span>
                    </li>
                </ul>
            </div>
        </div>
    </nav>

    <div class="container">
        @if ($role === 'admin')
            <div class="alert alert-info">Admin Access Granted</div>
        @elseif ($role === 'user')
            <div class="alert alert-success">User Area</div>
        @endif

        @yield('content')
    </div>

    <footer class="bg-light text-center mt-5 p-3 border-top">
        <p class="mb-0">&copy; 2025 Layout and Personalization. All rights reserved.</p>
    </footer>

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

5. Buat view untuk Admin Buat direktori admin di resources/views jika belum ada. Kemudian, buat resources/views/admin/dashboard.blade.php:

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

@section('title', 'Admin Dashboard')

@section('content')
    <h2 class="mb-4">Admin Dashboard</h2>
    <div class="list-group">
        <a href="#" class="list-group-item list-group-item-action">Manage Users</a>
        <a href="#" class="list-group-item list-group-item-action">Site Settings</a>
        <a href="#" class="list-group-item list-group-item-action">System Logs</a>
    </div>
@endsection
```

6. Buat view untuk User Buat direktori user di resources/views jika belum ada. Kemudian, buat resources/views/user/dashboard.blade.php:

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

@section('title', 'User Dashboard')

@section('content')
    <h2 class="mb-4">User Dashboard</h2>
    <p>Welcome to your dashboard, {{ $username }}!</p>
    <div class="list-group">
        <a href="#" class="list-group-item list-group-item-action">View Profile</a>
        <a href="#" class="list-group-item list-group-item-action">Edit Settings</a>
        <a href="#" class="list-group-item list-group-item-action">Logout</a>
    </div>
@endsection
```

7. Jalankan aplikasi <http://127.0.0.1:8000/admin> Output:

Layout and Personalization

Welcome, Yamato Admin

Admin Access Granted

Admin Dashboard

- Manage Users
- Site Settings
- System Logs

© 2025 Layout and Personalization. All rights reserved.



http://127.0.0.1:8000/user Output:

User Area

User Dashboard

Welcome to your dashboard, Liu User!

- [View Profile](#)
- [Edit Settings](#)
- [Logout](#)

© 2025 Layout and Personalization. All rights reserved.



2.4 Praktikum 4 - Partial Views, Blade Components, dan Theme Switching di Laravel 12

1. Buat proyek laravel dengan perintah Laravel new modul-4-blade-ui, kemudian masuk ke proyek dengan perintah cd modul-4-blade-ui.
2. Buat controller untuk menangani semua rute dan logika: php artisan make:controller UIController.
3. Buka routes/web.php dan tambahkan:

```
<?php

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

Route::get('/', [UIController::class, 'home'])->name('home');
Route::get('/about', [UIController::class, 'about'])->name('about');
Route::get('/contact', [UIController::class, 'contact'])->name('contact');
Route::get('/profile', [UIController::class, 'profile'])->name('profile');
Route::get('/switch-theme/{theme}', [UIController::class, 'switchTheme'])->name('switch-theme');
```

4. Update Controller Edit app/Http/Controllers/UIController.php:

```
<?php

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

class UIController extends Controller
{

    public function home(Request $request)
    {

        $theme = session('theme', 'light');
        $alertMessage = 'Selamat datang di laravel UI Integrated Demo!';
        $features = [
            'Partial Views',
            'Blade Component',
            'Theme Switching',
            'Bootstrap 5',
            'responsive Design'
        ];

        return view('home', compact('theme', 'alertMessage', 'features'));
    }

    public function about(Request $request) {
        $theme = session('theme', 'light');
        $alertMessage = 'Halaman ini menggunakan Partial Views!';
        $team = [
            ['name' => 'Deva', 'role' => 'Designer'],
            ['name' => 'Maisha', 'role' => 'Developer'],
            ['name' => 'Bunga', 'role' => 'Project Manager']
        ];
        return view('about', compact('theme', 'alertMessage', 'team'));
    }

    public function contact(Request $request)
    {
        $theme = session('theme', 'light');
        $departments = [
            'Technical Support',
            'Sales',
            'Billing',
            'General Inquiry',
        ];
        return view('contact', compact('theme', 'departments'));
    }

    public function profile(Request $request)
    {
        $theme = session('theme', 'light');
        $user = [
            'name' => 'Deva Risny',
            'email' => 'risnydeva@gmail.com',
            'join-date' => '2024-01-25',
            'preferences' => ['Email Notification', 'Dark mode', 'Newsletter']
        ];
        return view ('profile', compact('theme', 'user'));
    }

    public function switchTheme(Request $request, $theme)
    {
        if (in_array($theme, ['light', 'dark'])) {
            session(['theme' => $theme]);
        }
        return back();
    }
}
```

5. Buat Layout Utama dengan Theme Support Buat direktori layouts di resources/views jika belum ada.

Kemudian buat resources/views/layouts/app.blade.php:

```

<!DOCTYPE html>
<html lang="id" data-bs-theme="{{ $theme }}>
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@yield('title', 'Laravel UI Integrated Demo')</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet">
    <style>
        body{
            padding-top: 4rem;
            transition: all 0.3s ease;
            min-height: 100vh;
        }
        .theme-demo {
            border-radius: 10px;
            padding: 20px;
            margin: 10px 0;
            transition: all 0.3s ease;
        }
        .feature-card {
            transition: transform 0.2s ease;
        }
        .feature-card:hover {
            transform: translateY(-5px);
        }
    </style>
</head>
<body class="{{ $theme === 'dark' ? 'bg-dark text-light' : 'bg-light text-dark' }}>

    @include('partials.navigation')

    <div class="container mt-4">

        @if(isset($alertMessage) && !empty($alertMessage))
            @include('partials.alert', ['message' => $alertMessage, 'type' => 'info'])
        @endif

        @yield('content')
    </div>

    <x-footer :theme="$theme"/>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></script>
    <script>
        //smooth theme transition
        document.addEventListener('DOMContentLoaded', function() {

            const themeLinks = document.querySelectorAll('a[href*="switch-theme"]');
            themeLinks.forEach(link => {
                link.addEventListener('click', function(e) {
                    e.preventDefault();
                    window.location.href = this.href;
                });
            });
        });

    </script>
</body>
</html>

```

6. Buat Partial Views Buat direktori partials di resources/views dan buat file berikut:

- resources/views/partials/navigation.blade.php: 
- Selanjutnya, buatkan sebuah file resources/views/partials/alert.blade.php: 

7. Buat Blade Components php artisan make:component Footer php artisan make:component FeatureCard php artisan make:component TeamMember php artisan make:component ContactForm

- Kemudian, Edit resources/views/components/footer.blade.php:
- Kemudian, Edit resources/views/components/feature-card.blade.php:
- Selanjutnya, Edit resources/views/components/team-member.blade.php:
- Buat Main Views Buat view-view utama:
 - resources/views/home.blade.php
 - resources/views/about.blade.php
 - resources/views/partials/team-stats.blade.php
 - resources/views/contact.blade.php
 - resources/views/components/contact-form.blade.php
 - resources/views/profile.blade.php

8. Jalankan dan Test Aplikasi

- Home: <http://127.0.0.1:8000>

The screenshot shows the home page of the "Laravel UI Integrated Demo" application. At the top, there is a navigation bar with links for Home, About, Contact, and Profile. On the right side of the nav bar, there is a "Theme: Light" dropdown. Below the nav bar, a blue header bar displays the message "Selamat datang di laravel UI Integrated Demo!" with a close button (X) on the right. The main content area features a large title "Laravel UI Integrated Demo" with a rocket ship icon. Below the title is a subtitle: "Demonstrasi lengkap Partial Views, Blade Components, dan Theme Switching dalam satu aplikasi terpadu." Underneath the title, there are three yellow star icons inside separate boxes. In the bottom left corner, there is a sidebar titled "Fitur Utama" containing a bulleted list of features: - Partial Views, - Blade Component, - Theme Switching, - Bootstrap 5, and - responsive Design. In the bottom right corner, there is another sidebar titled "Teknologi yang Digunakan" containing a list of technologies: Laravel 12, Blade Templates, Bootstrap 5, PHP 8.4, and Session Management. At the very bottom of the page, there is a footer bar with the text "Laravel UI Integrated Demo", "Demonstrasi Partial Views, Blade Components, dan Theme Switching", "Current Theme: Light", and "© 2024 Laravel UI Demo. All rights reserved."

- About: <http://127.0.0.1:8000/about>

Laravel UI Demo Home About Contact Profile Theme: Light ▾

Halaman ini menggunakan Partial Views! X

About-Partial Views

Halaman ini mendemonstrasikan penggunaan partial Views dengan `@include` directive.

Tim kami

 Deva Designer Bergabung sejak 2024 dan kontribusi dalam pengembangan	 Maisha Developer Bergabung sejak 2024 dan kontribusi dalam pengembangan	 Bunga Project Manager Bergabung sejak 2024 dan kontribusi dalam pengembangan
--	---	--

Statistik Tim

3 Anggota	12+ Proyek	95% Kepuasan	2+ Tahun
--------------	---------------	-----------------	-------------

Laravel UI Integrated Demo Current Theme: Light
Demonstrasi Partial Views, Blade Components, dan Theme Switching © 2024 Laravel UI Demo. All rights reserved.

- Contact: <http://127.0.0.1:8000/contact>

Laravel UI Demo Home About Contact Profile Theme: Light ▾

Contact - Blade Components

Halaman ini mendemonstrasikan penggunaan Blade Components dengan props dan slots.

Informasi Kontak
Email: info@laraveldemo.com
Telepon: +62 21 1234 5678
Alamat: Jakarta, Indonesia
Department Tersedia:
• Technical Support
• Sales
• Billing
• General Inquiry

Laravel UI Integrated Demo Current Theme: Light
Demonstrasi Partial Views, Blade Components, dan Theme Switching © 2024 Laravel UI Demo. All rights reserved.

- Profile: <http://127.0.0.1:8000/profile>

The screenshot shows the 'Profile' page of the 'Laravel UI Demo'. At the top, there's a navigation bar with 'Laravel UI Demo' and links for Home, About, Contact, and Profile. On the right, it says 'Theme: Light' with a dropdown arrow. The main content area has a title 'Profile - Theme Demo' and a sub-instruction 'Halaman ini menunjukkan implementasi Theme Switching dengan session persistence.' To the left is a user profile card for 'Deva Risny' (risnydeva@gmail.com), showing a placeholder profile picture, the name 'Deva Risny', the email 'risnydeva@gmail.com', and the joining date 'Bergabung: 25 Jan 2024'. To the right is a 'Preferensi Pengguna' section. It shows 'Theme Saat Ini:' with a sun icon and the text 'Mode Light - Cerah dan jelas, cocok untuk siang hari.' Below that is 'Preferensi Lainnya:' with buttons for 'Email Notification', 'Dark mode', and 'Newsletter'. Under 'Change Theme:', there are two buttons: 'Light Mode' (which is highlighted in blue) and 'Dark Mode'. At the bottom of the page, there's a footer with 'Laravel UI Integrated Demo', 'Demonstrasi Partial Views, Blade Components, dan Theme Switching', 'Current Theme: Light' (with a small sun icon), and the copyright notice '© 2024 Laravel UI Demo. All rights reserved.'

3. Hasil dan Pembahasan

Berdasarkan hasil dari seluruh rangkaian praktikum:

Praktikum 1: Data berhasil dikirim dari controller ke view dan ditampilkan menggunakan sintaks {{ }} pada file .blade.php.

Praktikum 2: Struktur kontrol Blade seperti @if, @foreach, dan @isset berhasil diterapkan untuk menampilkan data secara dinamis.

Praktikum 3: Layout berbasis Bootstrap berhasil diterapkan, menghasilkan tampilan yang seragam dan responsif di seluruh halaman.

Praktikum 4: Partial views dan blade components berhasil dibuat untuk meningkatkan modularitas, serta theme switching berjalan dengan baik sesuai preferensi pengguna.

Secara keseluruhan, seluruh endpoint aplikasi (/dasar, /logic, /admin, /user, /about, /contact, /profile) berjalan dengan baik. Hal ini menunjukkan bahwa integrasi antara Controller, View, dan Blade Template Engine berfungsi dengan semestinya.

4. Kesimpulan

Dari hasil praktikum yang telah dilakukan dapat disimpulkan bahwa Controller memiliki peran penting sebagai pengatur logika aplikasi sekaligus penghubung antara Model dan View pada arsitektur MVC. Penggunaan Blade Template Engine membantu proses pembuatan tampilan menjadi lebih dinamis, rapi, dan mudah dikelola.

Selain itu, penerapan layout, partial view, dan component membuat struktur antarmuka web menjadi lebih modular dan efisien dalam pengembangan. Dengan dukungan Bootstrap, tampilan halaman menjadi lebih menarik, responsif, dan konsisten di setiap bagian aplikasi.

Secara keseluruhan, praktikum ini membantu mahasiswa memahami cara kerja dan penerapan konsep MVC pada framework Laravel secara menyeluruh, serta meningkatkan kemampuan dalam mengembangkan aplikasi web yang terstruktur dan mudah dikelola.

5. Referensi

1. Peran Controller dan View <https://www.rumahweb.com/jurnal/belajar-laravel-bagian-3/>
 2. Fungsi Layout <https://buildwithangga.com/tips/belajar-mengenal-views-pada-framework-laravel-11-sebagai-pemula>
 3. komponen <https://laravel.com/docs/12.x/blade#components>
 4. apa itu parsial <https://prateeksha.com/blog/in-laravel-how-to-use-partial-template>
-