Pemrograman Web Lanjut

SIB245007 | D-IV Sistem Informasi Bisnis

Pertemuan 3: **Frontend & Templating (Blade, Tailwind, Alpine)**

Ekosistem Frontend, Templating di Server, dan Interaktivitas Ringan

Yang Akan Kamu Pelajari

1. Membandingkan pola **MPA** dan **SPA**, serta menjelaskan posisi Blade dan Alpine.js sebagai pendekatan hibrida
2. Memahami konsep **template engine** dan pendekatan **utility-first CSS**, serta bagaimana keduanya bekerja sama menyusun tampilan halaman
3. Menjelaskan cara **Alpine.js** menambahkan interaktivitas sisi klien tanpa reload halaman, dengan server tetap sebagai sumber kebenaran

> Slide ini membahas konsep. Langkah membangun layout, halaman kasir, dan keranjang belanja dikerjakan di jobsheet praktikum, kali ini secara berkelompok.

Bagian 1

Etalase Kaca vs Pramuniaga yang Sigap

Etalase kaca

- Tidak pernah berubah sampai pemiliknya turun tangan menata ulang
- Pembeli yang ingin tahu sisa stok harus memanggil pramuniaga dan menunggu jawaban

Pramuniaga yang sigap

- Begitu pembeli mengambil satu barang, ia langsung tahu
- Menghitung ulang total di tempat, siap membatalkan pilihan tanpa pembeli mengulang dari awal

> Sebuah halaman yang dibangun murni dengan Blade, tanpa sentuhan JavaScript, berperilaku seperti etalase kaca: tiap kali pengguna menambah satu item ke keranjang, seluruh halaman dimuat ulang. Aplikasi dengan volume interaksi tinggi tidak punya waktu untuk reload puluhan kali dalam satu sesi.

MPA: Muat Ulang Seluruh Halaman

> **Multi-page application (MPA):** setiap kali pengguna berpindah tautan atau mengirim form, browser membuang HTML lama, meminta yang baru, dan merender dari nol.

- Sederhana dipahami dan gampang di-debug: setiap request punya jawaban HTML yang lengkap
- Terasa lambat untuk interaksi kecil yang sering berulang, mis. menambah satu produk ke keranjang

SPA: JavaScript Mengambil Alih

> **Single-page application (SPA):** satu halaman HTML dimuat sekali di awal, lalu seluruh perubahan berikutnya ditangani JavaScript di browser lewat request tersembunyi, memperbarui sebagian DOM saja tanpa reload.

- React, Vue, dan Angular adalah pustaka yang dibangun khusus untuk pola ini
- Cocok untuk interaksi yang sangat sering dan state yang kompleks
- Ongkosnya: kompleksitas build dan state yang tersebar di dua tempat (klien dan server)

Bukan Pilihan Hitam-Putih

MPA murni → **Server-rendered + sedikit JavaScript** → **SPA murni**

- Kebanyakan aplikasi nyata berada di tengah, bukan di salah satu ujung
- Ekosistem punya nama untuk titik tengah ini: **Livewire** (Laravel), **Hotwire** (Rails), **htmx**

> Pada pendekatan ini, server tetap jadi sumber kebenaran; JavaScript hanya bertugas mempercepat bagian yang terasa lambat kalau harus reload penuh.

Perbandingan Tiga Pola

Pola	Sumber Render	Cocok Untuk
MPA murni	Server, HTML penuh tiap navigasi	Halaman dengan interaksi jarang, SEO penting
SPA murni	Klien, JavaScript merender DOM	Aplikasi dengan interaksi sangat sering, state kompleks
Blade + Alpine.js	Server untuk struktur, klien untuk interaksi lokal	Aplikasi dengan halaman umumnya statis, sebagian kecil butuh reaktif

Kapan Blade + Alpine.js Cukup

- Halaman yang jarang berubah dalam satu sesi kerja (mis. daftar, laporan): render server biasa lewat Blade sudah cukup cepat
- Bagian yang butuh interaktivitas berulang tanpa reload (mis. keranjang belanja): Alpine.js mengisi celah itu
- Tidak perlu memaksa seluruh aplikasi pindah arsitektur hanya demi satu bagian kecil ini

> Memindahkan seluruh aplikasi ke SPA hanya demi satu komponen kecil adalah ongkos arsitektur yang tidak sepadan.

Bagian 2

Merender halaman dari server

Konsep Template Engine: Cetakan + Data → HTML

Template engine: alat yang menggabungkan sebuah cetakan (template) berisi placeholder dengan data sesungguhnya, menghasilkan dokumen akhir yang siap ditampilkan.

Template (placeholder) → **+ Data** → **Template Engine** → **HTML Akhir**

- Contoh sederhana: template `<h1>Halo, {{ $nama }}</h1>` + data `$nama = "Rani"` → hasil `<h1>Halo, Rani</h1>`
- Konsep yang sama dipakai lintas bahasa: Blade (Laravel/PHP), Jinja (Python), EJS (JavaScript), Twig (PHP)

Blade adalah salah satu implementasi konsep ini di ekosistem Laravel, bukan konsep itu sendiri.

Blade: Mesin Templating Laravel

> **Blade:** mesin templating bawaan Laravel yang merender HTML di server, memakai sintaks direktif seperti `@if` dan `@foreach` langsung di dalam berkas `.blade.php`.

> **Directive:** instruksi Blade yang diawali `@`, misalnya `@extends`, `@section`, dan `@yield`, dikompilasi Laravel menjadi PHP biasa sebelum dijalankan.

- Blade bukan bahasa baru, hanya cara ringkas menulis PHP di dalam template

Layout: Kerangka Ditulis Sekali

Hampir setiap halaman aplikasi web berbagi kerangka yang sama: judul tab, menu navigasi, dan area konten yang berubah-ubah. Layout Blade menghindari pengulangan kerangka itu di setiap berkas.

```
<!-- resources/views/layouts/app.blade.php -->
<!DOCTYPE html>
<html lang="id">
<head>
    <title>@yield('title', 'Nama Aplikasi')</title>
    @vite(['resources/css/app.css', 'resources/js/app.js'])
</head>
<body>
    <x-nav />
    <main>@yield('content')</main>
</body>
</html>
```

Halaman Mengisi Lubang Layout

```
<!-- resources/views/articles/index.blade.php -->
@extends('layouts.app')

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

@section('content')
    <h1 class="text-lg font-semibold mb-4">Daftar Artikel</h1>
    <div class="grid grid-cols-3 gap-4">
        @foreach ($articles as $article)
            <div class="border rounded-md p-3">
                <p class="font-medium">{{ $article->title }}</p>
            </div>
        @endforeach
    </div>
@endsection
```

- Halaman tidak menulis ulang `<html>`, `<head>`, atau menu navigasi

> Daftar lengkap directive Blade: `laravel.com/docs/blade`

`{{ }}` dan `{!! !!}`: Menampilkan Data ke HTML

`{{ }}`: sintaks Blade untuk menampilkan nilai sebuah variabel atau ekspresi PHP ke dalam HTML, otomatis melakukan escaping karakter HTML.

`{!! !!}`: sintaks Blade yang juga menampilkan nilai sebuah variabel, tapi tanpa escaping sama sekali; isinya dicetak apa adanya sebagai HTML mentah.

- Contoh: `{{ $article->title }}` menampilkan judul artikel, otomatis aman dari karakter seperti `<` yang bisa disalahgunakan untuk menyuntikkan markup asing

Data yang berasal dari input pengguna, termasuk judul artikel yang diketik lewat form, selalu wajib ditampilkan lewat `{{ }}`, bukan `{!! !!}`. Sintaks kedua melewati escaping sama sekali dan membuka celah *cross-site scripting* kalau isinya pernah datang dari input yang tidak dipercaya.

- Pembahasan tuntas soal keamanan input ada di pertemuan validasi & keamanan

Component: Potongan yang Dipakai Ulang

> **Component:** potongan Blade yang bisa dipakai ulang lintas halaman, misalnya kartu produk atau tombol, didaftarkan sebagai berkas terpisah dan dipanggil lewat tag `<x-nama-component>`.

- `<x-nav />` pada layout sebelumnya adalah component
- Kartu artikel dan tombol adalah kandidat component berikutnya

Vite: Build Tool Frontend

> **Vite:** build tool frontend yang dipakai Laravel secara bawaan, menjalankan dev server dengan hot reload lewat `npm run dev` selama pengembangan, dan menggabungkan (*bundle*) serta memperkecil ukuran (*minify*) seluruh CSS & JavaScript jadi berkas produksi lewat `npm run build`.

app.css / app.js → **Vite** → **Tag lewat @vite**

- `@vite([...])` menggantikan tag `<script>` / `<link>` manual

Tiga Berkas di Balik `@vite`

1. `resources/css/app.css`: baris pertama `@import 'tailwindcss';`, tanpa berkas konfigurasi `tailwind.config.js` terpisah
2. `vite.config.js`: mendaftarkan plugin `laravel()` dan `tailwindcss()`
3. `package.json`: mendaftarkan kedua paket itu sebagai `devDependencies`

> Ketiganya sudah ada di proyekmu sejak Pertemuan 1. Pertemuan ini kamu mulai benar-benar memakainya.

Dua Terminal: `artisan serve` + `npm run dev`

- Vite berjalan sebagai dev server terpisah dari `php artisan serve`, di terminal kedua
- Yang diharapkan: `VITE vX.X.X ready` diikuti alamat lokal
- Mengedit `app.css` atau berkas Blade langsung terlihat tanpa refresh (hot reload)

Kalau `npm run dev` dihentikan, direktif `@vite` pada halaman yang dimuat ulang akan gagal menemukan dev server dan melempar `ViteManifestNotFoundException`.

`composer run dev` menjalankan artisan serve, npm run dev, dan proses lain sekaligus dalam satu terminal. Mulailah dengan dua terminal terpisah supaya jelas proses mana yang gagal.

Tailwind: Utility-First CSS

- Class utility ditempel langsung di elemen, bukan aturan CSS terpisah di berkas lain
- Contoh: `class="bg-slate-900 text-white rounded-md px-4 py-2"` untuk tombol gelap dengan sudut membulat dan padding
- Satu class = satu keputusan gaya kecil, mudah ditebak dan dihapus

> Referensi lengkap class utility: `tailwindcss.com/docs` (jangan dihafalkan, cari saat butuh)

Bagian 3

Keranjang belanja tanpa reload halaman

Alpine.js: State di Dalam Markup

> **Alpine.js:** pustaka JavaScript ringan yang menambahkan state dan interaktivitas langsung lewat atribut HTML (`x-data` , `x-model` , `@click`), diinstal lewat npm seperti dependensi JavaScript lain.

- Bukan framework SPA mini, melainkan pelengkap halaman yang sudah dirender Blade
- Elemen apa pun di dalam `x-data` , termasuk elemen anaknya, bisa membaca dan mengubah state lewat atribut Alpine lain

Mekanisme Inti: `x-data`

```
<div x-data="{ cart: [] }">
  <button @click="cart.push({ id: 1, price: 15000 })">
    Tambah
  </button>
  <span x-text="cart.length"></span> item di keranjang
</div>
```

- `cart` dimulai sebagai array kosong, tombol Tambah mendorong satu objek baru setiap kali diklik
- `x-text="cart.length"` otomatis memperbarui angka, tanpa satu baris kode pun yang memerintahkan DOM untuk refresh
- Alpine mengamati perubahan pada `cart` dan menyinkronkan tampilan sendiri

Atribut Alpine yang Akan Kamu Pakai

Atribut	Fungsi
`x-data`	Mendeklarasikan state lokal
`@click`	Menjalankan kode saat elemen diklik
`x-text`	Menampilkan nilai reaktif
`x-for`	Mengulang elemen untuk setiap item
`x-model`	Mengikat input ke state

> Daftar lengkap atribut: `alpinejs.dev`

Instalasi lewat npm, Bukan CDN

```
npm install alpinejs
```

```
// resources/js/app.js
import Alpine from 'alpinejs';

window.Alpine = Alpine;
Alpine.start();
```

- `app.js` sudah dimuat `@vite` di layout, tidak ada tag `<script>` baru yang perlu ditambahkan
- Urutan pemuatan lewat bundler otomatis menghindari masalah yang biasanya diatasi atribut `defer` pada CDN

Pola Interaktivitas: Klik untuk Memperbarui State

Klik item → **addToCart(id, name, price)** → **cart berubah** → **x-for + x-text update**

- `x-data` membungkus daftar item yang relevan, mendefinisikan `cart`, `addToCart`, dan `subtotal()`
- `subtotal()` memakai `reduce` untuk menjumlahkan harga tiap item

> Bukti tidak ada reload: address bar dan favicon tab tidak berkedip seperti biasanya.

Aturan Emas: Server Menghitung Ulang

Subtotal yang dihitung Alpine di sisi klien murni untuk tampilan. Setiap kali form dikirim, server menghitung ulang totalnya dari data di basis data, bukan dari angka yang sempat ditampilkan Alpine di browser. Apa pun yang berasal dari browser tetap bisa dimanipulasi sebelum sampai ke server.

Aturan praktis: logika kosmetik (menampilkan subtotal, menyorot item baru) aman ditaruh di Alpine. Keputusan bisnis (total final, validitas data) wajib dihitung ulang di server.

Versi Lengkap: `Alpine.data`

- Objek `x-data` inline cukup untuk belajar mekanismenya
- Pada studi kasus Simple POS yang kamu bangun di jobsheet, keranjang didaftarkan lewat `Alpine.data('posCart', ...)` di `app.js`, lengkap dengan scan SKU, diskon, dan `sessionStorage`
- Konsep yang sama, skala yang berbeda

> Kode lengkap: `github.com/se-polinema/simple-pos`, branch `chapter-03`

Rangkuman

- MPA memuat ulang seluruh halaman tiap navigasi, SPA merender semuanya di klien, Blade + Alpine.js mengambil jalan tengah: server tetap merender struktur, Alpine hanya menambah reaktivitas pada bagian yang butuh
- Layout Blade lewat `@extends` / `@section` / `@yield` menghindari pengulangan kerangka HTML; `@vite` memuat Tailwind CSS dan JavaScript lewat Vite, dengan `npm run dev` memberi hot reload
- Tailwind bekerja lewat class utility yang ditempel di elemen; `{{ }}` melakukan escaping otomatis, `{!! !!}` tidak dan membuka celah XSS
- Alpine.js diinstal lewat npm; `x-data` mendeklarasikan state di markup, `@click` dan `x-text` membaca serta mengubahnya secara reaktif, dan subtotal yang dihitung klien tetap wajib diverifikasi ulang di server

Referensi & Diskusi

Dokumentasi resmi Laravel (Blade, Vite) · Tailwind CSS (tailwindcss.com) · Alpine.js (alpinejs.dev)

Kode lengkap: `github.com/se-polinema/simple-pos`

Pertemuan berikutnya: Desain Basis Data, Migrasi, dan Seeding