## BUKU BAHAN AJAR SISWA

# PEMROGRAMAN WEB LANJUTAN

**Program Keahlian Teknik Jaringan Komputer dan Telekomunikasi (TJKT)**

*PHP OOP • Laravel • MVC • Eloquent ORM • Autentikasi • Git • Vibecoding*

*Disusun Berdasarkan Capaian Pembelajaran Kurikulum Merdeka SMK*

**Untuk Jenjang Kelas XI SMK**

**Yusuf Burhani, S.Kom., S.Pd**

# Kata Pengantar

Buku ini disusun sebagai bahan ajar bagi peserta didik mata pelajaran Pemrograman Web Lanjutan pada Program Keahlian Teknik Jaringan Komputer dan Telekomunikasi (TJKT) di Sekolah Menengah Kejuruan kelas XI. Buku ini merupakan kelanjutan langsung dari Pemrograman Web Dasar kelas X, yang telah membekali peserta didik dengan logika pemrograman, HTML, CSS, JavaScript, serta praktik dasar PHP dan MySQL melalui satu form login sederhana.

Materi dalam buku ini mengajak peserta didik melangkah dari PHP native menuju framework Laravel — salah satu framework PHP paling populer dan paling banyak dicari di dunia kerja. Peserta didik akan mempelajari pemrograman berorientasi objek (OOP) sebagai prasyarat, pola arsitektur MVC (Model-View-Controller), pembangunan fitur CRUD (Create, Read, Update, Delete) secara penuh dan terstruktur, autentikasi, relasi antar tabel database, styling menggunakan Bootstrap, hingga version control menggunakan Git dan GitHub.

Buku ini juga melanjutkan pembahasan vibecoding yang telah diperkenalkan pada kelas X, kini pada tingkat yang lebih matang — bagaimana menggunakan AI secara efektif untuk mempercepat penulisan boilerplate code, membantu debugging, dan melakukan code review, sambil tetap menjunjung pemahaman mendalam dan verifikasi kritis terhadap setiap kode yang dihasilkan.

Setiap bab disusun dengan pendekatan yang konsisten: konsep dijelaskan dengan mengaitkan kembali pada kompetensi yang telah dikuasai di kelas X, dilengkapi contoh kode nyata, dan bermuara pada satu proyek aplikasi CRUD terintegrasi yang dibangun secara bertahap sepanjang buku, memuncak pada proyek akhir semester yang mengintegrasikan seluruh kompetensi menjadi sebuah aplikasi web yang utuh dan siap dipresentasikan.

Semoga buku ini membekali peserta didik dengan fondasi pengembangan web yang solid, relevan dengan kebutuhan dunia kerja, dan menjadi batu loncatan menuju kompetensi yang lebih matang pada jenjang kelas XII.

Penyusun

# Daftar Isi

## Peta Kompetensi Buku

Tabel berikut memetakan setiap bab terhadap fokus kompetensi dan elemen pola MVC yang dibangun dalam buku ini.

| Bab | Judul | Fokus Utama |
|---|---|---|
| 1 | Pendahuluan: Dari PHP Native Menuju Framework | Alasan penggunaan framework, konsep MVC |
| 2 | PHP OOP Dasar | Class, object, constructor, visibility, inheritance |
| 3 | Instalasi dan Struktur Proyek Laravel | Composer, struktur folder, Artisan CLI |
| 4 | Routing dan Controller | Definisi rute, route parameter, response |
| 5 | Blade Templating | Sintaks tampilan, layout, komponen |
| 6 | Eloquent ORM dan Migration | Struktur database sebagai kode, query tanpa SQL manual |
| 7 | CRUD Penuh dengan Laravel | Resource routing, integrasi penuh MVC |
| 8 | Validasi dan Form Request | Validasi server-side, pesan error |
| 9 | Autentikasi Laravel dengan Breeze | Login, register, middleware auth |
| 10 | Relasi Antar Tabel | One-to-Many, Many-to-Many, eager loading |
| 11 | Styling dengan Bootstrap | Komponen UI siap pakai, responsif |
| 12 | Git dan GitHub untuk Kolaborasi | Version control, branch, kolaborasi |
| 13 | Vibecoding dengan Laravel | AI untuk boilerplate, debugging, code review |
| 14 | Proyek Akhir Semester | Integrasi penuh seluruh kompetensi |
| 15 | Rangkuman dan Evaluasi Akhir | Konsolidasi |

# BAB 1 — Pendahuluan: Dari PHP Native Menuju Framework

Pemrograman Web Lanjutan merupakan mata pelajaran kelas XI pada Program Keahlian Teknik Jaringan Komputer dan Telekomunikasi (TJKT) yang melanjutkan kompetensi Pemrograman Web Dasar kelas X. Jika pada kelas X peserta didik telah mempelajari logika pemrograman, HTML, CSS, JavaScript, serta praktik dasar PHP dan MySQL melalui satu form login sederhana, buku ini mengajak peserta didik melangkah lebih jauh: membangun aplikasi web yang terstruktur, dapat dipelihara (maintainable), dan mendekati standar yang digunakan di industri sesungguhnya, menggunakan framework Laravel.

Peralihan dari PHP native menuju framework bukanlah mengganti seluruh pengetahuan yang telah dipelajari sebelumnya, melainkan mengorganisasikannya menjadi struktur yang lebih rapi. Setiap baris kode PHP native yang telah ditulis pada Bab 12 buku kelas X — koneksi database, query SQL, validasi form — pada dasarnya tetap terjadi di balik layar sebuah aplikasi Laravel, hanya saja dibungkus dalam pola yang lebih terorganisir sehingga aplikasi besar tetap mudah dipahami dan dikembangkan oleh banyak orang sekaligus.

### *Tujuan Pembelajaran*

- Peserta didik memahami alasan dan manfaat penggunaan framework dibanding PHP native murni.
- Peserta didik memahami kedudukan mata pelajaran ini sebagai kelanjutan langsung dari Pemrograman Web Dasar kelas X.
- Peserta didik mengenal Laravel sebagai framework PHP yang akan digunakan sepanjang buku ini beserta lingkungan kerjanya.

## 1.1 Mengapa Framework? Mengingat Kembali Keterbatasan PHP Native

Pada Bab 12 buku Pemrograman Web Dasar, form login dibangun dengan menulis koneksi database, query SQL, dan validasi secara manual dalam satu file PHP. Pendekatan ini berjalan baik untuk proyek kecil dengan satu-dua fitur, namun mulai terasa menyulitkan ketika aplikasi bertumbuh: kode database tercampur dengan kode tampilan, setiap halaman baru berarti mengulang pola koneksi yang sama, dan tidak ada aturan baku tentang di mana logika bisnis seharusnya diletakkan — sehingga aplikasi berskala menengah ke atas menjadi sulit dipelihara, apalagi dikerjakan oleh tim yang terdiri dari banyak orang.

Framework hadir untuk mengatasi masalah ini dengan menyediakan **struktur baku** dan **alat bantu siap pakai** untuk kebutuhan yang hampir selalu muncul di setiap aplikasi web: routing, koneksi database, validasi, autentikasi, dan keamanan dasar. Dengan struktur yang konsisten, seorang programmer baru yang bergabung ke sebuah tim dapat lebih cepat memahami kode yang sudah ada, karena mengikuti pola yang telah disepakati secara luas oleh komunitas.

## 1.2 Mengapa Laravel?

Laravel adalah salah satu framework PHP paling populer di dunia, dikenal karena sintaksnya yang ekspresif (mudah dibaca layaknya bahasa Inggris biasa), dokumentasi yang sangat lengkap, serta ekosistem paket pendukung yang luas untuk hampir setiap kebutuhan (autentikasi, pengiriman email, antrian tugas, dan lain-lain). Di Indonesia khususnya, Laravel menjadi salah satu framework PHP yang paling banyak digunakan pada lowongan kerja bidang web developer, menjadikannya pilihan yang relevan secara langsung dengan kesiapan kerja lulusan SMK Program Keahlian TJKT.

| Aspek | PHP Native (Kelas X) | Laravel (Kelas XI) |
|---|---|---|
| Struktur kode | Bebas, sering tercampur dalam satu file | Terstruktur mengikuti pola MVC (Model-View-Controller) |
| Koneksi database | Ditulis manual dengan mysqli setiap kali | Menggunakan Eloquent ORM, query menjadi kode PHP biasa |
| Routing URL | Satu file PHP = satu URL | Diatur terpusat pada file routes, lebih fleksibel |
| Keamanan dasar | Ditulis manual (prepared statement, dsb.) | Sebagian besar sudah tersedia bawaan (CSRF protection, dsb.) |
| Kurva belajar awal | Lebih sederhana untuk pemula | Perlu memahami konsep OOP dan struktur MVC terlebih dahulu |

## 1.3 Konsep MVC (Model-View-Controller)

MVC adalah pola arsitektur perangkat lunak yang memisahkan sebuah aplikasi menjadi tiga bagian dengan tanggung jawab berbeda, dan menjadi konsep paling fundamental yang harus dipahami sebelum mempelajari Laravel lebih jauh.

| Komponen | Tanggung Jawab | Analogi Restoran |
|---|---|---|
| Model | Mengelola data dan interaksi dengan database | Dapur — tempat bahan mentah (data) diolah |
| View | Menampilkan tampilan/antarmuka kepada pengguna | Meja makan — tempat hidangan disajikan ke pelanggan |
| Controller | Menjembatani permintaan pengguna dengan Model dan View | Pelayan — menerima pesanan, meneruskan ke dapur, menyajikan hasilnya |

Ketika seorang pengguna mengakses sebuah halaman, alurnya secara sederhana adalah: permintaan (request) diterima oleh **Controller**, Controller meminta data yang diperlukan dari **Model**, Model mengambil data tersebut dari database, kemudian Controller mengirimkan data itu ke **View** untuk ditampilkan kepada pengguna dalam bentuk halaman HTML yang sudah jadi.

## 1.4 Menyiapkan Lingkungan Kerja Laravel

| Kebutuhan | Keterangan |
|---|---|
| PHP versi terbaru (8.1 ke atas) | Laravel versi terbaru mensyaratkan versi PHP modern |
| Composer | Pengelola paket (dependency manager) PHP, digunakan untuk menginstal Laravel dan seluruh library pendukungnya |
| XAMPP/Laragon | Menyediakan PHP, MySQL, dan web server, sama seperti yang digunakan pada kelas X |
| Node.js dan NPM | Diperlukan untuk mengelola aset frontend (CSS/JS) pada proyek Laravel modern |
| Visual Studio Code + ekstensi Laravel | Editor kode yang sama seperti kelas X, dilengkapi ekstensi seperti Laravel Blade Snippets |

Composer menjadi alat baru yang belum digunakan pada buku kelas X — perannya mirip seperti manajer paket pada sistem operasi Linux (apt pada Ubuntu, misalnya) yang telah dipelajari pada mata pelajaran Keamanan Jaringan, namun khusus untuk mengelola pustaka (library) kode PHP.

### *Rangkuman*

Framework seperti Laravel hadir mengatasi keterbatasan PHP native pada aplikasi berskala menengah ke atas, menyediakan struktur baku melalui pola MVC (Model-View-Controller) yang memisahkan pengelolaan data, tampilan, dan logika penghubung keduanya. Laravel dipilih karena popularitas, dokumentasi lengkap, dan relevansinya dengan kebutuhan dunia kerja. Persiapan lingkungan kerja meliputi PHP versi terbaru, Composer, XAMPP/Laragon, dan Node.js sebagai fondasi sebelum instalasi Laravel pada bab berikutnya.

# BAB 2 — PHP OOP Dasar: Prasyarat Wajib Sebelum Laravel

### *Tujuan Pembelajaran*

- Peserta didik mampu menjelaskan konsep dasar pemrograman berorientasi objek (OOP).
- Peserta didik mampu membuat dan menggunakan class, property, dan method dalam PHP.
- Peserta didik mampu menerapkan konsep constructor, inheritance, dan visibility pada PHP OOP.

## 2.1 Dari Prosedural Menuju Berorientasi Objek

Seluruh kode PHP yang ditulis pada buku kelas X bersifat **prosedural** — serangkaian instruksi yang dijalankan dari atas ke bawah menggunakan variabel dan fungsi lepas. Pemrograman Berorientasi Objek (Object-Oriented Programming/OOP) menawarkan cara berpikir berbeda: membungkus data dan fungsi-fungsi terkait ke dalam satu kesatuan yang disebut **objek**, dicetak dari sebuah cetakan yang disebut **class**. Laravel dibangun sepenuhnya di atas konsep OOP, sehingga penguasaan bab ini adalah prasyarat mutlak sebelum melanjutkan ke Bab 3.

Analogi paling umum untuk memahami class dan objek adalah cetakan kue dan kue itu sendiri: class 'Siswa' adalah cetakan yang mendefinisikan bahwa setiap siswa memiliki nama, kelas, dan nilai, sementara setiap objek siswa (Budi, Siti, Andi) adalah 'kue' hasil cetakan tersebut — memiliki struktur yang sama, namun dengan isi data yang berbeda-beda.

## 2.2 Membuat Class, Property, dan Method

```
<?php
class Siswa {
    // Property - menyimpan data
    public $nama;
    public $kelas;
    public $nilai;

    // Method - fungsi milik class ini
    public function tampilkanBiodata() {
        return $this->nama . " - Kelas " . $this->kelas . " - Nilai: " . $this-
>nilai;
    }
}

// Membuat objek (instance) dari class Siswa
$siswa1 = new Siswa();
$siswa1->nama = "Budi Santoso";
$siswa1->kelas = "XI TJKT 1";
$siswa1->nilai = 88;

echo $siswa1->tampilkanBiodata();
// Output: Budi Santoso - Kelas XI TJKT 1 - Nilai: 88
```

Kata kunci `$this` merujuk pada objek yang sedang aktif, memungkinkan method mengakses property miliknya sendiri. Setiap objek yang dibuat dari class yang sama memiliki property yang independen satu sama lain — mengubah nilai pada $siswa1 tidak akan memengaruhi objek $siswa2 yang dibuat terpisah.

## 2.3 Constructor: Inisialisasi Otomatis Saat Objek Dibuat

Constructor adalah method khusus bernama `__construct()` yang otomatis dijalankan setiap kali sebuah objek baru dibuat, umumnya digunakan untuk langsung mengisi nilai property saat objek diciptakan, tanpa perlu menuliskan baris terpisah untuk setiap property seperti contoh sebelumnya.

```
<?php
class Siswa {
    public $nama;
    public $kelas;

    public function __construct($nama, $kelas) {
        $this->nama = $nama;
        $this->kelas = $kelas;
    }

    public function tampilkanBiodata() {
        return $this->nama . " - " . $this->kelas;
    }
}

// Kini nama dan kelas langsung diisi saat objek dibuat
$siswa1 = new Siswa("Budi Santoso", "XI TJKT 1");
echo $siswa1->tampilkanBiodata();
```

## 2.4 Visibility: Public, Private, dan Protected

Visibility mengatur dari mana sebuah property atau method dapat diakses, sebuah konsep keamanan dasar dalam OOP yang disebut **encapsulation** — menyembunyikan detail internal sebuah class agar tidak dapat diubah sembarangan dari luar.

| Visibility | Dapat Diakses Dari |
|---|---|
| public | Dari mana saja, termasuk dari luar class |
| private | Hanya dari dalam class itu sendiri |
| protected | Dari dalam class itu sendiri dan class turunannya (inheritance) |

```
<?php
class RekeningBank {
    private $saldo = 0; // tidak bisa diakses langsung dari luar class

    public function setor($jumlah) {
```

```
        $this->saldo += $jumlah;
    }

    public function cekSaldo() {
        return $this->saldo;
    }
}

$rekening = new RekeningBank();
$rekening->setor(50000);
echo $rekening->cekSaldo(); // 50000

// $rekening->saldo = 999999999; // ERROR - tidak diizinkan karena private
```

Property `$saldo` yang bersifat private hanya dapat diubah melalui method `setor()` yang telah ditentukan aturannya, mencegah kode dari luar mengubah saldo secara sembarangan (misalnya langsung menetapkan angka besar tanpa melalui proses setor yang sah) — sebuah praktik yang menjaga integritas data, relevan langsung dengan prinsip keamanan aplikasi yang akan diperdalam pada buku Kelas XII.

## 2.5 Inheritance: Pewarisan Antar Class

Inheritance memungkinkan sebuah class (disebut child/subclass) mewarisi property dan method dari class lain (disebut parent/superclass), menghindari penulisan kode yang sama berulang kali pada class-class yang memiliki kemiripan karakteristik.

```
<?php
class Pengguna {
    public $nama;

    public function __construct($nama) {
        $this->nama = $nama;
    }

    public function login() {
        return $this->nama . " berhasil login";
    }
}

class Admin extends Pengguna {
    public function hapusData() {
        return $this->nama . " menghapus data (khusus admin)";
    }
}

$admin1 = new Admin("Budi");
echo $admin1->login();      // diwarisi dari class Pengguna
echo $admin1->hapusData();  // method khusus milik Admin
```

Konsep inheritance ini akan sangat sering ditemui saat mempelajari Laravel — misalnya, setiap Controller yang dibuat akan mewarisi (extends) class Controller bawaan Laravel, dan setiap Model akan mewarisi class Model bawaan yang menyediakan banyak fungsi siap pakai untuk berinteraksi dengan database.

## Aktivitas Pembelajaran

1. Peserta didik membuat class 'Buku' dengan property judul, penulis, dan tahun terbit, beserta method untuk menampilkan informasi buku, kemudian membuat tiga objek buku berbeda dan menampilkan informasinya.
2. Peserta didik memodifikasi class RekeningBank pada bab ini dengan menambahkan method tarik() yang tidak mengizinkan penarikan melebihi saldo yang tersedia.
3. Peserta didik membuat class 'Kendaraan' sebagai parent class, kemudian membuat class 'Motor' dan 'Mobil' yang mewarisi 'Kendaraan' dengan tambahan method khusus masing-masing.

## Rangkuman

Pemrograman Berorientasi Objek membungkus data dan fungsi terkait ke dalam class yang dicetak menjadi objek, dengan constructor untuk inisialisasi otomatis, visibility (public/private/protected) untuk mengatur hak akses, dan inheritance untuk mewariskan karakteristik antar class. Seluruh konsep ini menjadi fondasi wajib yang digunakan secara masif oleh Laravel, dan akan langsung dipraktikkan sejak instalasi Laravel pada bab berikutnya.

## Soal Latihan / Refleksi

4. Jelaskan perbedaan antara class dan objek menggunakan analogi selain cetakan kue.
5. Apa fungsi constructor (__construct) pada sebuah class PHP?
6. Jelaskan perbedaan visibility public dan private beserta alasan mengapa private penting untuk keamanan data.
7. Tuliskan contoh sederhana penggunaan inheritance (extends) pada dua class PHP buatan sendiri.

# BAB 3 — Instalasi dan Struktur Proyek Laravel

### *Tujuan Pembelajaran*

- Peserta didik mampu menginstal Laravel menggunakan Composer.
- Peserta didik mampu menjelaskan fungsi setiap folder utama dalam struktur proyek Laravel.
- Peserta didik mampu menjalankan proyek Laravel menggunakan Artisan CLI.

## 3.1 Menginstal Laravel dengan Composer

```
# Menginstal Laravel installer secara global (sekali saja)
composer global require laravel/installer

# Membuat proyek Laravel baru
laravel new nama-proyek

# Alternatif menggunakan composer secara langsung
composer create-project laravel/laravel nama-proyek

# Masuk ke folder proyek dan menjalankan server pengembangan
cd nama-proyek
php artisan serve
```

Perintah `php artisan serve` menjalankan server pengembangan bawaan Laravel pada alamat http://127.0.0.1:8000, memungkinkan proyek langsung diakses melalui browser tanpa perlu konfigurasi Apache/virtual host terlebih dahulu — sangat memudahkan tahap pengembangan sebelum proyek benar-benar di-deploy ke server sungguhan seperti yang telah dipelajari pada Bab 14 buku kelas X.

## 3.2 Struktur Folder Proyek Laravel

| Folder/File | Fungsi |
|---|---|
| app/Models/ | Berisi seluruh class Model — representasi tabel database (bagian 'M' pada MVC) |
| app/Http/Controllers/ | Berisi seluruh class Controller — logika penghubung request dan response (bagian 'C' pada MVC) |
| resources/views/ | Berisi seluruh file Blade template — tampilan yang dilihat pengguna (bagian 'V' pada MVC) |
| routes/web.php | Mendaftarkan seluruh URL/rute aplikasi beserta controller yang menanganinya |
| database/migrations/ | Berisi definisi struktur tabel database dalam bentuk kode PHP (dibahas Bab 6) |
| public/ | Folder yang dapat diakses langsung dari browser — berisi file index.php, gambar, CSS, JS terkompilasi |
| .env | File konfigurasi environment — kredensial database, mode aplikasi, dan pengaturan sensitif lainnya |
| composer.json | Mendaftarkan seluruh dependency/paket PHP yang digunakan proyek |

Struktur folder yang konsisten ini adalah keunggulan utama framework dibanding PHP native — seorang developer yang telah terbiasa dengan Laravel dapat langsung memahami di mana harus mencari kode tertentu pada proyek Laravel siapa pun, karena strukturnya selalu sama.

## 3.3 Mengenal Artisan CLI

Artisan adalah command-line interface bawaan Laravel yang menyediakan berbagai perintah untuk mempercepat pekerjaan pengembangan berulang, seperti membuat file Controller, Model, atau Migration secara otomatis tanpa perlu mengetik struktur dasarnya dari nol.

| Perintah Artisan | Fungsi |
|---|---|
| php artisan serve | Menjalankan server pengembangan lokal |
| php artisan make:controller NamaController | Membuat file Controller baru secara otomatis |
| php artisan make:model NamaModel | Membuat file Model baru secara otomatis |
| php artisan make:migration nama_migration | Membuat file Migration baru |
| php artisan migrate | Menjalankan seluruh migration, membentuk tabel pada database (dibahas Bab 6) |
| php artisan route:list | Menampilkan seluruh rute yang telah didaftarkan pada aplikasi |
| php artisan --version | Menampilkan versi Laravel yang terinstal |

## 3.4 Konfigurasi Awal Melalui File .env

```
APP_NAME="Sistem Data Siswa"
APP_ENV=local
APP_DEBUG=true
APP_URL=http://localhost:8000

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

File `.env` menyimpan seluruh konfigurasi yang berbeda-beda antara lingkungan pengembangan (lokal) dan lingkungan produksi (server sungguhan) — prinsip yang sejalan dengan praktik pemisahan kredensial database yang telah disinggung pada Bab 14 buku kelas X. File ini **tidak boleh diunggah ke repositori Git publik** karena berisi informasi sensitif seperti password database, sebuah aturan keamanan dasar yang akan diperdalam pada Bab 12 mengenai Git dan GitHub.

## Aktivitas Pembelajaran

8. Peserta didik menginstal Composer dan Laravel pada komputer masing-masing, membuat proyek Laravel pertama, dan menjalankannya menggunakan php artisan serve hingga muncul halaman selamat datang Laravel di browser.
9. Peserta didik mengeksplorasi struktur folder proyek Laravel yang baru dibuat, mencatat isi masing-masing folder utama (app, resources, routes, database) dalam bentuk tabel ringkasan.
10. Peserta didik mengonfigurasi file .env untuk tersambung ke database MySQL lokal, membuat database kosong melalui phpMyAdmin, dan memverifikasi koneksi berhasil.

## Rangkuman

Laravel diinstal menggunakan Composer dan dijalankan melalui perintah php artisan serve untuk pengembangan lokal. Struktur folder proyek Laravel terorganisir sesuai pola MVC (Models, Controllers, Views) dengan routes dan database/migrations sebagai pelengkap. Artisan CLI mempercepat pekerjaan berulang seperti pembuatan Controller dan Model, sementara file .env menyimpan konfigurasi environment termasuk kredensial database yang harus dijaga kerahasiaannya.

## Soal Latihan / Refleksi

11. Sebutkan perintah untuk membuat proyek Laravel baru menggunakan Composer.
12. Jelaskan fungsi folder app/Models/ dan app/Http/Controllers/ dalam struktur proyek Laravel.
13. Apa fungsi file .env, dan mengapa file ini tidak boleh diunggah ke repositori Git publik?
14. Sebutkan tiga perintah Artisan beserta fungsinya masing-masing.

# BAB 4 — Routing dan Controller

### *Tujuan Pembelajaran*

- Peserta didik mampu mendefinisikan rute (route) pada aplikasi Laravel.
- Peserta didik mampu membuat dan menggunakan Controller untuk menangani logika aplikasi.
- Peserta didik mampu menerapkan route parameter dan mengembalikan response dalam berbagai bentuk.

## 4.1 Routing: Peta Jalan Aplikasi

Routing adalah proses mendefinisikan bagaimana aplikasi merespons permintaan terhadap URL tertentu. Pada PHP native kelas X, setiap halaman umumnya direpresentasikan oleh satu file terpisah (misalnya login.php, proses_login.php). Pada Laravel, seluruh rute didaftarkan secara terpusat pada file routes/web.php, memberikan gambaran menyeluruh mengenai seluruh URL yang tersedia pada aplikasi hanya dengan membuka satu file.

```
<?php
// routes/web.php
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return "Selamat datang di aplikasi!";
});

Route::get('/tentang', function () {
    return view('tentang'); // menampilkan file resources/views/tentang.blade.php
});
```

## 4.2 Membuat dan Menggunakan Controller

Meskipun rute dapat langsung berisi kode logika (seperti contoh function di atas), praktik yang baik adalah memindahkan logika tersebut ke dalam Controller, menjaga file routes/web.php tetap ringkas dan mudah dibaca — sejalan dengan prinsip pemisahan tanggung jawab pada pola MVC yang dipelajari di Bab 1.

```
# Membuat controller baru menggunakan Artisan
php artisan make:controller SiswaController

<?php
// app/Http/Controllers/SiswaController.php
namespace App\Http\Controllers;

class SiswaController extends Controller
{
    public function index()
    {
        return "Ini adalah daftar seluruh siswa";
    }

    public function show($id)
```

```
    {
        return "Menampilkan detail siswa dengan ID: " . $id;
    }
}

<?php
// routes/web.php
use App\Http\Controllers\SiswaController;

Route::get('/siswa', [SiswaController::class, 'index']);
Route::get('/siswa/{id}', [SiswaController::class, 'show']);
```

Perhatikan bahwa `SiswaController extends Controller` — inheritance yang telah dipelajari pada Bab 2 — mewarisi berbagai fungsi bawaan dari class Controller milik Laravel, tanpa perlu peserta didik menulisnya sendiri dari nol.

## 4.3 Route Parameter

Route parameter memungkinkan sebagian dari URL diperlakukan sebagai variabel yang dapat berubah-ubah nilainya, seperti `{id}` pada contoh sebelumnya. Ketika pengguna mengakses /siswa/5, nilai 5 tersebut otomatis diteruskan sebagai argumen ke method show() pada Controller.

| URL yang Diakses | Method yang Dipanggil | Nilai $id |
|---|---|---|
| /siswa/1 | show(1) | 1 |
| /siswa/25 | show(25) | 25 |
| /siswa/abc | show('abc') | "abc" (kecuali dibatasi hanya angka) |

```
// Membatasi parameter hanya menerima angka menggunakan regex
Route::get('/siswa/{id}', [SiswaController::class, 'show'])->whereNumber('id');
```

## 4.4 Mengembalikan Berbagai Jenis Response

| Jenis Response | Contoh Kode | Kegunaan |
|---|---|---|
| Teks/String | return "Halo"; | Response sederhana, jarang dipakai pada aplikasi nyata |
| View (Blade) | return view('siswa.index'); | Menampilkan halaman HTML yang dirender dari template Blade (Bab 5) |
| JSON | return response()->json($data); | Mengembalikan data terstruktur, umum digunakan pada API (buku Kelas XII) |
| Redirect | return redirect('/siswa'); | Mengalihkan pengguna ke URL lain, misalnya setelah form berhasil disimpan |

## 4.5 HTTP Verb: GET vs POST

Sebagaimana atribut method pada tag form HTML yang telah dipelajari pada Bab 8 buku kelas X, Laravel membedakan rute berdasarkan HTTP verb (metode permintaan): `Route::get()` untuk menampilkan halaman/data, dan `Route::post()` untuk mengirim/menyimpan data — sebuah pola yang akan langsung dipraktikkan secara penuh saat membangun fitur CRUD pada Bab 7.

```
Route::get('/siswa/tambah', [SiswaController::class, 'create']);   // menampilkan
form
Route::post('/siswa', [SiswaController::class, 'store']);          // memproses data
form yang dikirim
```

### *Aktivitas Pembelajaran*

15. Peserta didik membuat SiswaController menggunakan Artisan, menambahkan method index() dan show($id), serta mendaftarkan rute yang sesuai pada routes/web.php, kemudian mengujinya melalui browser.
16. Peserta didik membuat route parameter tambahan untuk menampilkan detail berdasarkan dua parameter sekaligus, misalnya /nilai/{kelas}/{siswa}, dan menampilkan kedua nilai tersebut pada response.
17. Peserta didik membuat sebuah rute yang mengembalikan response dalam bentuk JSON berisi data siswa sederhana (array asosiatif), menguji hasilnya menggunakan browser atau Postman.

### *Rangkuman*

Routing pada Laravel didaftarkan secara terpusat pada routes/web.php, idealnya diarahkan ke Controller untuk memisahkan logika dari definisi rute. Route parameter memungkinkan bagian URL menjadi variabel dinamis, sementara Laravel mendukung berbagai jenis response (teks, view, JSON, redirect) serta membedakan rute berdasarkan HTTP verb (GET untuk menampilkan, POST untuk mengirim data). Kompetensi ini menjadi fondasi sebelum mempelajari Blade templating pada bab berikutnya.

### *Soal Latihan / Refleksi*

18. Jelaskan perbedaan antara mendefinisikan logika langsung pada routes/web.php dengan memindahkannya ke Controller.
19. Tuliskan kode route dan method Controller untuk menampilkan detail produk berdasarkan parameter id.
20. Jelaskan perbedaan penggunaan Route::get() dan Route::post() beserta contoh kasusnya masing-masing.
21. Sebutkan tiga jenis response yang dapat dikembalikan oleh sebuah method Controller pada Laravel.

# BAB 5 — Blade Templating

### *Tujuan Pembelajaran*

- Peserta didik mampu menggunakan sintaks dasar Blade untuk menampilkan data pada view.
- Peserta didik mampu menerapkan struktur kontrol (if, foreach) dalam Blade.
- Peserta didik mampu membuat layout utama dan komponen yang dapat digunakan berulang.

## 5.1 Apa Itu Blade?

Blade adalah engine templating bawaan Laravel yang memperluas kemampuan HTML biasa dengan sintaks tambahan yang ringkas untuk menampilkan data dinamis, menjalankan struktur kontrol, dan menyusun ulang komponen tampilan — menggantikan pola pencampuran `<?php echo $variabel; ?>` yang telah dipelajari pada buku kelas X dengan sintaks yang jauh lebih ringkas dan mudah dibaca.

```
{{-- resources/views/tentang.blade.php --}}
<h1>Halo, {{ $nama }}!</h1>
<p>Selamat datang di aplikasi sekolah kami.</p>

<?php
// Pada Controller, data dikirim ke view menggunakan parameter kedua
public function tentang() {
    return view('tentang', ['nama' => 'Budi Santoso']);
}
```

## 5.2 Menampilkan Data dengan Sintaks {{ }}

Sintaks kurung kurawal ganda `{{ $variabel }}` menampilkan nilai variabel sekaligus secara otomatis melindungi dari serangan Cross-Site Scripting (XSS) — konsep keamanan yang akan diperdalam secara khusus pada Bab 5 buku Kelas XII — dengan mengubah karakter HTML berbahaya menjadi bentuk aman (escaping) sebelum ditampilkan.

```
<h1>{{ $judulHalaman }}</h1>
<p>Total siswa terdaftar: {{ $totalSiswa }}</p>
<p>Harga: Rp {{ number_format($harga, 0, ',', '.') }}</p>
```

## 5.3 Struktur Kontrol pada Blade

Blade menyediakan sintaks ringkas untuk struktur kontrol yang telah dipelajari secara konsep pada Bab 4 dan 5 buku kelas X (percabangan dan perulangan), kini diterapkan langsung untuk mengatur tampilan berdasarkan data dari database.

```
{{-- Percabangan --}}
@if ($totalSiswa > 30)
    <p class="text-danger">Kelas penuh!</p>
@elseif ($totalSiswa > 20)
    <p class="text-warning">Kelas hampir penuh.</p>
@else
```

```
    <p class="text-success">Kelas masih tersedia.</p>
@endif

{{-- Perulangan menampilkan daftar siswa --}}
<ul>
@foreach ($daftarSiswa as $siswa)
    <li>{{ $siswa->nama }} - {{ $siswa->kelas }}</li>
@endforeach
</ul>

{{-- Menangani kondisi daftar kosong --}}
@forelse ($daftarSiswa as $siswa)
    <li>{{ $siswa->nama }}</li>
@empty
    <li>Belum ada data siswa.</li>
@endforelse
```

Perhatikan bagaimana `@if`, `@foreach`, dan struktur kontrol lain pada Blade merupakan penerapan langsung dari konsep percabangan dan perulangan yang telah dikuasai sejak buku kelas X, hanya dengan sintaks yang disesuaikan agar lebih ringkas ketika ditulis di antara markup HTML.

## 5.4 Layout Utama dan Pewarisan Template

Hampir setiap halaman pada sebuah aplikasi web memiliki bagian yang sama (header, navigasi, footer) dan bagian yang berbeda-beda (konten utama). Blade menyediakan konsep layout utama yang dapat 'diwarisi' oleh halaman-halaman lain — konsep yang mengingatkan kembali pada inheritance OOP di Bab 2, kini diterapkan pada level template.

```
{{-- resources/views/layouts/app.blade.php --}}
<!DOCTYPE html>
<html lang="id">
<head>
    <title>@yield('title', 'Sistem Data Siswa')</title>
</head>
<body>
    <nav>Menu Navigasi Sekolah</nav>
    <main>
        @yield('content')
    </main>
    <footer>&copy; 2026 SMK TJKT</footer>
</body>
</html>

{{-- resources/views/siswa/index.blade.php --}}
@extends('layouts.app')

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

@section('content')
    <h1>Daftar Seluruh Siswa</h1>
    @foreach ($daftarSiswa as $siswa)
        <p>{{ $siswa->nama }}</p>
```

```
    @endforeach
@endsection
```

Directive `@extends` menyatakan bahwa halaman ini menggunakan layout app.blade.php sebagai kerangka dasarnya, `@section` mengisi bagian-bagian yang telah ditandai dengan `@yield` pada layout tersebut. Pendekatan ini menghindari penulisan ulang header, navigasi, dan footer pada setiap halaman — cukup ditulis sekali pada layout utama.

## 5.5 Komponen Blade yang Dapat Digunakan Berulang

Untuk bagian tampilan kecil yang digunakan berulang di banyak tempat (misalnya kartu produk, tombol, atau alert pesan), Blade menyediakan sistem komponen yang lebih modular dibanding sekadar layout.

```
{{-- resources/views/components/alert.blade.php --}}
<div class="alert alert-{{ $jenis }}">
    {{ $slot }}
</div>

{{-- Digunakan pada halaman lain --}}
<x-alert jenis="success">
    Data siswa berhasil disimpan!
</x-alert>
```

### *Aktivitas Pembelajaran*

22. Peserta didik membuat layout utama (layouts/app.blade.php) lengkap dengan navigasi sederhana, kemudian membuat dua halaman berbeda (Beranda dan Tentang) yang mewarisi layout tersebut menggunakan @extends dan @section.
23. Peserta didik membuat sebuah view yang menampilkan daftar data (array asosiatif sederhana dari Controller) menggunakan @foreach, lengkap dengan kondisi @forelse untuk menangani data kosong.
24. Peserta didik membuat sebuah komponen Blade kustom (misalnya kartu profil atau badge status) dan menggunakannya pada minimal dua halaman berbeda.

### *Rangkuman*

Blade adalah engine templating Laravel yang menyediakan sintaks {{ }} untuk menampilkan data dengan perlindungan XSS otomatis, directive @if/@foreach untuk struktur kontrol, serta sistem layout (@extends, @section, @yield) dan komponen untuk menghindari duplikasi kode tampilan. Kompetensi ini melengkapi pemahaman routing dan Controller dari bab sebelumnya, membentuk pemahaman MVC yang utuh sebelum mempelajari koneksi database melalui Eloquent pada bab berikutnya.

### *Soal Latihan / Refleksi*

25. Jelaskan manfaat sintaks {{ }} pada Blade dibanding menuliskan <?php echo ?> secara manual.
26. Tuliskan kode Blade untuk menampilkan pesan berbeda berdasarkan status siswa (Lulus/Tidak Lulus) menggunakan @if.
27. Jelaskan fungsi directive @extends, @section, dan @yield dalam sistem layout Blade.

28. Apa perbedaan antara @foreach dan @forelse pada Blade?

# BAB 6 — Eloquent ORM dan Migration

### *Tujuan Pembelajaran*

- Peserta didik mampu membuat struktur tabel database menggunakan Migration.
- Peserta didik mampu membuat Model Eloquent dan menghubungkannya dengan tabel database.
- Peserta didik mampu melakukan operasi dasar database (query) menggunakan Eloquent tanpa menulis SQL manual.

## 6.1 Dari Query SQL Manual Menuju Eloquent ORM

Pada Bab 12 buku kelas X, interaksi dengan database dilakukan dengan menulis query SQL secara manual menggunakan mysqli dan prepared statement. Object-Relational Mapping (ORM) adalah teknik yang memungkinkan interaksi dengan database dilakukan menggunakan kode PHP berorientasi objek, tanpa perlu menulis SQL secara eksplisit untuk operasi-operasi umum. **Eloquent** adalah ORM bawaan Laravel yang merepresentasikan setiap tabel database sebagai sebuah Model (class PHP), dan setiap baris data sebagai objek dari Model tersebut.

| Pendekatan | Contoh Kode |
|---|---|
| SQL Manual (Kelas X) | $stmt = $koneksi->prepare("SELECT * FROM siswa WHERE id = ?"); |
| Eloquent (Kelas XI) | $siswa = Siswa::find($id); |

## 6.2 Migration: Struktur Database sebagai Kode

Migration adalah cara mendefinisikan struktur tabel database (nama kolom, tipe data, relasi) menggunakan kode PHP alih-alih perintah SQL CREATE TABLE secara langsung. Keuntungan utamanya adalah struktur database dapat disimpan dan dibagikan melalui Git bersama kode aplikasi, sehingga setiap anggota tim (atau server produksi) dapat membentuk struktur database yang identik hanya dengan menjalankan satu perintah.

```
# Membuat file migration baru
php artisan make:migration create_siswa_table

<?php
// database/migrations/xxxx_create_siswa_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    public function up()
    {
```

```
        Schema::create('siswa', function (Blueprint $table) {
            $table->id();                          // kolom id, auto-increment,
primary key
            $table->string('nama');
            $table->string('kelas');
            $table->integer('nilai')->default(0);
            $table->timestamps();                  // kolom created_at dan
updated_at otomatis
        });
    }

    public function down()
    {
        Schema::dropIfExists('siswa');
    }
};

# Menjalankan seluruh migration, membentuk tabel pada database
php artisan migrate
```

Method `up()` mendefinisikan apa yang terjadi saat migration dijalankan (membuat tabel), sedangkan method `down()` mendefinisikan cara membatalkannya (menghapus tabel) — berguna ketika sebuah perubahan struktur database perlu dibatalkan menggunakan `php artisan migrate:rollback`.

## 6.3 Membuat Model Eloquent

```
# Membuat model sekaligus migration dalam satu perintah
php artisan make:model Siswa -m

<?php
// app/Models/Siswa.php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;

class Siswa extends Model
{
    protected $fillable = ['nama', 'kelas', 'nilai'];
}
```

Secara default, Eloquent mengasumsikan nama tabel adalah bentuk jamak (plural) dari nama Model dalam huruf kecil — Model `Siswa` otomatis terhubung ke tabel `siswas` kecuali ditentukan lain, sehingga sedikit penyesuaian nama tabel pada migration (mis. `siswa` bukan `siswas`) mungkin perlu ditambahkan properti `protected $table = 'siswa';` pada Model. Property `$fillable` mendaftarkan kolom mana saja yang boleh diisi secara massal (mass assignment) — sebuah lapisan keamanan dasar yang akan dibahas lebih mendalam pada Bab 5 buku Kelas XII.

## 6.4 Operasi Dasar Query dengan Eloquent

| Operasi | Kode Eloquent | Setara SQL |
|---|---|---|
| Ambil semua data | Siswa::all(); | SELECT * FROM siswa |
| Ambil satu data by ID | Siswa::find(1); | SELECT * FROM siswa WHERE id = 1 |
| Filter data | Siswa::where('kelas', 'XI TJKT 1')->get(); | SELECT * FROM siswa WHERE kelas = 'XI TJKT 1' |
| Urutkan data | Siswa::orderBy('nilai', 'desc')->get(); | SELECT * FROM siswa ORDER BY nilai DESC |
| Tambah data baru | Siswa::create([...]); | INSERT INTO siswa (...) VALUES (...) |
| Perbarui data | $siswa->update([...]); | UPDATE siswa SET ... WHERE id = ... |
| Hapus data | $siswa->delete(); | DELETE FROM siswa WHERE id = ... |

```php
<?php
// Contoh penggunaan pada Controller
public function index()
{
    $daftarSiswa = Siswa::orderBy('nama')->get();
    return view('siswa.index', ['daftarSiswa' => $daftarSiswa]);
}
```

Perhatikan bagaimana kode Eloquent jauh lebih ringkas dan mudah dibaca dibanding menuliskan query SQL secara manual dengan mysqli, tanpa mengorbankan keamanan — Eloquent secara otomatis menerapkan prepared statement di balik layar, sehingga tetap terlindungi dari SQL Injection sebagaimana yang dipelajari secara manual pada Bab 12 buku kelas X.

## 6.5 Soft Delete: Menghapus Tanpa Benar-Benar Menghilangkan Data

Pada beberapa kasus, menghapus data secara permanen (seperti method delete() yang telah dipelajari) berisiko — misalnya data siswa yang dihapus keliru tidak dapat dikembalikan. Laravel menyediakan fitur **Soft Delete**, yaitu data tidak benar-benar dihapus dari database, melainkan hanya ditandai dengan kolom timestamp khusus (deleted_at), sehingga masih dapat dipulihkan bila diperlukan.

```php
<?php
// Menambahkan kolom deleted_at pada migration
$table->softDeletes();

// Pada Model, tambahkan trait SoftDeletes
use Illuminate\Database\Eloquent\SoftDeletes;

class Siswa extends Model
{
    use SoftDeletes;
```

```
    protected $fillable = ['nama', 'kelas_id', 'nilai'];
}

<?php
$siswa->delete();                    // kini hanya menandai deleted_at, bukan hapus
permanen
Siswa::withTrashed()->get();         // mengambil data termasuk yang sudah "dihapus"
Siswa::onlyTrashed()->get();         // hanya menampilkan data yang sudah dihapus
$siswa->restore();                   // memulihkan data yang sudah dihapus
```

Fitur ini sangat relevan untuk data penting seperti data akademik siswa, di mana kesalahan penghapusan permanen dapat berakibat serius, sebuah pertimbangan yang sejalan dengan prinsip kehati-hatian dalam pengelolaan data yang telah ditekankan pada mata pelajaran Keamanan Jaringan.

## *Aktivitas Pembelajaran*

29. Peserta didik membuat migration dan model untuk tabel 'siswa' sesuai contoh pada bab ini, menjalankan php artisan migrate, dan memverifikasi tabel terbentuk melalui phpMyAdmin.
30. Peserta didik menggunakan Tinker (php artisan tinker) untuk mencoba operasi create, all, where, dan delete secara langsung dari command line tanpa membangun halaman web terlebih dahulu.
31. Peserta didik membuat migration dan model kedua untuk tabel 'mata_pelajaran', kemudian menampilkan datanya pada sebuah halaman menggunakan kombinasi Controller, Model, dan Blade view.

## *Rangkuman*

Migration mendefinisikan struktur tabel database menggunakan kode PHP yang dapat dibagikan melalui version control, dijalankan menggunakan php artisan migrate. Model Eloquent merepresentasikan tabel database sebagai class PHP, memungkinkan operasi query (all, find, where, create, update, delete) dilakukan tanpa menulis SQL manual, sambil tetap aman dari SQL Injection. Kompetensi ini menjadi fondasi wajib sebelum membangun fitur CRUD penuh pada bab berikutnya.

## *Soal Latihan / Refleksi*

32. Jelaskan keuntungan menggunakan migration dibanding membuat tabel secara manual melalui phpMyAdmin.
33. Tuliskan kode Eloquent untuk mengambil seluruh data siswa dari kelas 'XI TJKT 2' saja.
34. Apa fungsi property $fillable pada sebuah Model Eloquent?
35. Jelaskan mengapa Eloquent tetap aman dari SQL Injection meskipun tidak menulis prepared statement secara manual.

# BAB 7 — CRUD Penuh dengan Laravel

### *Tujuan Pembelajaran*

- Peserta didik mampu membangun fitur Create, Read, Update, dan Delete secara lengkap menggunakan Laravel.
- Peserta didik mampu menghubungkan routing, Controller, Model, dan View menjadi satu alur CRUD yang utuh.
- Peserta didik mampu menerapkan resource routing untuk menyederhanakan pendaftaran rute CRUD.

## 7.1 Studi Kasus: Sistem Data Siswa

Bab ini menyatukan seluruh kompetensi dari Bab 3 hingga 6 (instalasi, routing, Controller, Blade, Eloquent) menjadi satu fitur utuh: sistem CRUD data siswa. Berbeda dengan Bab 12 buku kelas X yang hanya membangun satu form login, bab ini membangun siklus pengelolaan data yang lengkap — menambah, melihat, mengubah, dan menghapus data — pola yang menjadi fondasi hampir seluruh aplikasi bisnis nyata (sistem akademik, inventaris, hingga aplikasi e-commerce).

## 7.2 Resource Routing: Menyederhanakan Pendaftaran Rute CRUD

Alih-alih mendaftarkan setiap rute CRUD satu per satu (index, create, store, edit, update, destroy), Laravel menyediakan `Route::resource()` yang secara otomatis mendaftarkan ketujuh rute standar CRUD hanya dengan satu baris kode.

```
<?php
// routes/web.php
use App\Http\Controllers\SiswaController;

Route::resource('siswa', SiswaController::class);
```

| HTTP Verb | URL | Method Controller | Fungsi |
|---|---|---|---|
| GET | /siswa | index | Menampilkan seluruh data siswa |
| GET | /siswa/create | create | Menampilkan form tambah data baru |
| POST | /siswa | store | Menyimpan data baru dari form |
| GET | /siswa/{id}/edit | edit | Menampilkan form edit data |
| PUT/PATCH | /siswa/{id} | update | Memperbarui data yang sudah ada |
| DELETE | /siswa/{id} | destroy | Menghapus data |

## 7.3 Membangun Controller CRUD Lengkap

```
# Membuat controller sekaligus dengan seluruh method resource
php artisan make:controller SiswaController --resource

<?php
namespace App\Http\Controllers;

use App\Models\Siswa;
use Illuminate\Http\Request;

class SiswaController extends Controller
{
    public function index() {
        $daftarSiswa = Siswa::orderBy('nama')->get();
        return view('siswa.index', compact('daftarSiswa'));
    }

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

    public function store(Request $request) {
        Siswa::create($request->only(['nama', 'kelas', 'nilai']));
        return redirect('/siswa')->with('sukses', 'Data berhasil ditambahkan!');
    }

    public function edit($id) {
        $siswa = Siswa::findOrFail($id);
        return view('siswa.edit', compact('siswa'));
    }

    public function update(Request $request, $id) {
        $siswa = Siswa::findOrFail($id);
        $siswa->update($request->only(['nama', 'kelas', 'nilai']));
        return redirect('/siswa')->with('sukses', 'Data berhasil diperbarui!');
    }

    public function destroy($id) {
        Siswa::destroy($id);
        return redirect('/siswa')->with('sukses', 'Data berhasil dihapus!');
    }
}
```

Fungsi `compact('daftarSiswa')` adalah cara ringkas mengirim variabel ke view — setara dengan menuliskan `['daftarSiswa' => $daftarSiswa]` seperti pada Bab 5. Method `findOrFail()` otomatis menampilkan halaman error 404 apabila data dengan id tersebut tidak ditemukan, mencegah aplikasi menampilkan error yang membingungkan pengguna.

## 7.4 Membangun View untuk Setiap Aksi CRUD

```
{{-- resources/views/siswa/index.blade.php --}}
@extends('layouts.app')
@section('content')
<h1>Daftar Siswa</h1>
<a href="{{ route('siswa.create') }}">+ Tambah Siswa</a>
```

```
@if (session('sukses'))
    <div class="alert alert-success">{{ session('sukses') }}</div>
@endif

<table border="1">
    <tr><th>Nama</th><th>Kelas</th><th>Nilai</th><th>Aksi</th></tr>
    @foreach ($daftarSiswa as $siswa)
    <tr>
        <td>{{ $siswa->nama }}</td>
        <td>{{ $siswa->kelas }}</td>
        <td>{{ $siswa->nilai }}</td>
        <td>
            <a href="{{ route('siswa.edit', $siswa->id) }}">Edit</a>
            <form action="{{ route('siswa.destroy', $siswa->id) }}" method="POST"
style="display:inline">
                @csrf
                @method('DELETE')
                <button type="submit" onclick="return confirm('Yakin
hapus?')">Hapus</button>
            </form>
        </td>
    </tr>
    @endforeach
</table>
@endsection
```

Directive `@csrf` menyisipkan token keamanan otomatis yang mencegah serangan Cross-Site Request Forgery (dibahas mendalam pada Bab 5 buku Kelas XII), dan `@method('DELETE')` diperlukan karena HTML form native hanya mendukung GET dan POST, sehingga Laravel menggunakan trik field tersembunyi untuk mensimulasikan method PUT/PATCH/DELETE yang sesungguhnya dibutuhkan resource routing.

## 7.5 Alur Lengkap CRUD: Menghubungkan Semua Bagian

Untuk memastikan pemahaman menyeluruh, perhatikan alur lengkap ketika seorang pengguna menghapus data siswa: (1) pengguna mengklik tombol 'Hapus' pada halaman index, (2) form dengan `@method('DELETE')` mengirim request ke rute `/siswa/{id}` dengan verb DELETE, (3) Laravel mencocokkan request tersebut dengan resource routing pada Bab 7.2, meneruskannya ke method `destroy($id)` pada SiswaController, (4) method tersebut memanggil `Siswa::destroy($id)` yang berinteraksi dengan Eloquent (Bab 6) untuk menjalankan query DELETE ke database, (5) pengguna dialihkan kembali ke halaman index dengan pesan sukses yang ditampilkan menggunakan Blade (Bab 5).

### *Aktivitas Pembelajaran*

36. Peserta didik membangun sistem CRUD data siswa lengkap mengikuti seluruh contoh pada bab ini: migration, model, resource routing, controller, dan keempat view (index, create, edit, serta partial form yang dapat digunakan bersama).
37. Peserta didik menambahkan fitur pencarian sederhana pada halaman index menggunakan Eloquent where() dan input pencarian pada form.

38. Peserta didik mengembangkan CRUD serupa untuk entitas berbeda (misalnya 'Buku Perpustakaan' atau 'Barang Inventaris') secara mandiri, menerapkan seluruh pola yang telah dipelajari tanpa mencontoh langsung.

## *Rangkuman*

CRUD penuh pada Laravel dibangun dengan menghubungkan resource routing (menyederhanakan pendaftaran tujuh rute standar), Controller resource (index, create, store, edit, update, destroy), Model Eloquent, dan Blade view untuk setiap aksi. Directive @csrf dan @method melengkapi form HTML agar sesuai dengan konvensi RESTful yang digunakan Laravel. Proyek CRUD ini menjadi bukti nyata integrasi seluruh kompetensi Bab 3–6, dan menjadi pola dasar yang akan terus digunakan pada bab-bab berikutnya.

## *Soal Latihan / Refleksi*

39. Jelaskan keuntungan menggunakan Route::resource() dibanding mendaftarkan ketujuh rute CRUD satu per satu.
40. Jelaskan fungsi directive @csrf pada form Laravel dan risiko yang dicegahnya.
41. Mengapa method destroy dan update pada form HTML memerlukan @method('DELETE') atau @method('PUT')?
42. Jelaskan alur lengkap dari pengguna mengisi form tambah data hingga data tersebut tersimpan di database dan pengguna melihat pesan sukses.

# BAB 8 — Validasi dan Form Request

### *Tujuan Pembelajaran*

- Peserta didik mampu menerapkan validasi data pada Controller menggunakan Laravel.
- Peserta didik mampu menampilkan pesan error validasi pada Blade view.
- Peserta didik mampu menggunakan Form Request untuk memisahkan logika validasi dari Controller.

## 8.1 Mengapa Validasi Tetap Diperlukan Meski Sudah Ada Atribut HTML5

Pada Bab 8 buku kelas X, atribut seperti `required` telah digunakan untuk validasi dasar di sisi browser (client-side). Namun sebagaimana ditekankan pada Bab 12 buku kelas X, validasi client-side dapat dilewati oleh pengguna yang mengubah kode HTML secara manual atau mengirim request langsung tanpa melalui form. Oleh karena itu, validasi di sisi server (server-side) pada Controller tetap wajib diterapkan sebagai lapisan pertahanan yang sesungguhnya, mengulang prinsip yang telah dipelajari namun kini diterapkan dengan cara Laravel yang jauh lebih ringkas.

## 8.2 Validasi Dasar pada Controller

```
<?php
public function store(Request $request)
{
    $validated = $request->validate([
        'nama'  => 'required|string|max:100',
        'kelas' => 'required|string',
        'nilai' => 'required|integer|min:0|max:100',
        'email' => 'required|email|unique:siswa,email',
    ]);

    Siswa::create($validated);
    return redirect('/siswa')->with('sukses', 'Data berhasil ditambahkan!');
}
```

| Aturan Validasi | Fungsi |
|---|---|
| required | Kolom wajib diisi, tidak boleh kosong |
| string / integer | Tipe data harus sesuai (teks / bilangan bulat) |
| max:100 / min:0 | Membatasi panjang teks atau nilai maksimal/minimal angka |
| email | Format harus berupa alamat email yang valid |
| unique:siswa,email | Nilai harus unik, belum pernah ada pada tabel siswa kolom email |

Apabila validasi gagal, Laravel secara otomatis mengalihkan pengguna kembali ke halaman form disertai seluruh pesan error — peserta didik tidak perlu menulis logika if-else manual seperti pada Bab 4 buku kelas X untuk memeriksa setiap kondisi validasi satu per satu.

## 8.3 Menampilkan Pesan Error pada Blade

```
{{-- resources/views/siswa/create.blade.php --}}
@extends('layouts.app')
@section('content')
<form action="{{ route('siswa.store') }}" method="POST">
    @csrf
    <label>Nama:</label>
    <input type="text" name="nama" value="{{ old('nama') }}">
    @error('nama')
        <span class="text-danger">{{ $message }}</span>
    @enderror

    <label>Nilai:</label>
    <input type="number" name="nilai" value="{{ old('nilai') }}">
    @error('nilai')
        <span class="text-danger">{{ $message }}</span>
    @enderror

    <button type="submit">Simpan</button>
</form>
@endsection
```

Directive `@error('nama')` menampilkan pesan kesalahan spesifik untuk kolom tersebut apabila validasi gagal, sementara fungsi `old('nama')` mengisi kembali nilai yang sebelumnya diketik pengguna sebelum form dikirim ulang — sebuah pengalaman pengguna yang jauh lebih baik dibanding mengharuskan pengguna mengetik ulang seluruh form dari awal hanya karena satu kolom yang salah.

## 8.4 Form Request: Memisahkan Validasi dari Controller

Ketika aturan validasi menjadi cukup kompleks, sebaiknya logika validasi dipisahkan ke dalam class tersendiri yang disebut Form Request, menjaga Controller tetap ringkas — sejalan dengan filosofi pemisahan tanggung jawab yang telah dipelajari sejak Bab 1.

```
# Membuat Form Request baru
php artisan make:request StoreSiswaRequest

<?php
// app/Http/Requests/StoreSiswaRequest.php
namespace App\Http\Requests;

use Illuminate\Foundation\Http\FormRequest;

class StoreSiswaRequest extends FormRequest
{
    public function authorize() {
        return true; // izinkan siapa saja mengirim request ini (bisa diubah untuk
```

```
otorisasi)
    }

    public function rules() {
        return [
            'nama'  => 'required|string|max:100',
            'kelas' => 'required|string',
            'nilai' => 'required|integer|min:0|max:100',
        ];
    }

    public function messages() {
        return [
            'nama.required' => 'Nama siswa wajib diisi.',
            'nilai.max' => 'Nilai tidak boleh lebih dari 100.',
        ];
    }
}

<?php
// Controller kini jauh lebih ringkas
use App\Http\Requests\StoreSiswaRequest;

public function store(StoreSiswaRequest $request)
{
    Siswa::create($request->validated());
    return redirect('/siswa')->with('sukses', 'Data berhasil ditambahkan!');
}
```

Method `authorize()` pada Form Request menjadi titik awal penerapan otorisasi — menentukan siapa yang boleh menjalankan aksi ini — sebuah konsep yang akan diperdalam signifikan melalui middleware dan role-based access control pada buku Kelas XII.

## *Aktivitas Pembelajaran*

43. Peserta didik menambahkan validasi lengkap pada method store() dan update() sistem CRUD siswa yang telah dibangun pada Bab 7, mencakup minimal empat aturan validasi berbeda.
44. Peserta didik memodifikasi view create dan edit untuk menampilkan pesan error menggunakan @error dan mengisi ulang nilai form menggunakan old().
45. Peserta didik membuat Form Request terpisah (StoreSiswaRequest dan UpdateSiswaRequest) untuk memindahkan validasi dari Controller, lengkap dengan pesan error kustom menggunakan method messages().

## *Rangkuman*

Validasi server-side pada Laravel dilakukan melalui method validate() pada Controller dengan aturan ringkas (required, string, integer, email, unique), jauh lebih sederhana dibanding menulis if-else manual seperti pada PHP native. Directive @error dan fungsi old() pada Blade menampilkan pesan kesalahan dan mempertahankan input pengguna. Form Request memisahkan logika validasi kompleks dari Controller, sekaligus menjadi titik awal penerapan otorisasi yang akan diperluas pada buku Kelas XII.

### *Soal Latihan / Refleksi*

46. Mengapa validasi server-side tetap diperlukan meskipun form HTML sudah menggunakan atribut required?
47. Tuliskan aturan validasi Laravel untuk kolom email yang wajib diisi, harus format email valid, dan harus unik pada tabel pengguna.
48. Jelaskan fungsi directive @error dan fungsi old() pada Blade view.
49. Apa keuntungan memindahkan validasi ke dalam Form Request dibanding menuliskannya langsung pada Controller?

# BAB 9 — Autentikasi Laravel dengan Breeze

### *Tujuan Pembelajaran*

- Peserta didik mampu menginstal dan mengonfigurasi Laravel Breeze untuk fitur autentikasi.
- Peserta didik mampu menjelaskan alur kerja registrasi, login, dan logout menggunakan Breeze.
- Peserta didik mampu melindungi rute tertentu agar hanya dapat diakses oleh pengguna yang telah login.

## 9.1 Dari Form Login Manual Menuju Sistem Autentikasi Siap Pakai

Bab 12 buku kelas X membangun form login secara manual: membuat tabel pengguna, menghash password menggunakan password_hash(), memverifikasi menggunakan password_verify(), dan memeriksa sesi login secara manual. Seluruh langkah tersebut benar dan penting untuk dipahami — namun pada praktiknya, hampir setiap aplikasi Laravel menggunakan paket **Laravel Breeze** yang menyediakan seluruh fitur autentikasi (registrasi, login, logout, reset password, verifikasi email) secara siap pakai dan telah mengikuti praktik keamanan terbaik, sehingga developer tidak perlu membangunnya dari nol setiap kali membuat proyek baru.

Memahami cara kerja login manual pada kelas X tetap penting — pemahaman tersebut menjadi dasar untuk mengerti apa yang sesungguhnya terjadi 'di balik layar' Breeze, sehingga peserta didik tidak sekadar menggunakan fitur ini sebagai kotak hitam (black box), melainkan memahami mengapa fitur tersebut bekerja.

## 9.2 Menginstal Laravel Breeze

```
# Menginstal paket Breeze melalui Composer
composer require laravel/breeze --dev

# Menjalankan installer Breeze (pilih stack Blade untuk konsistensi dengan bab
sebelumnya)
php artisan breeze:install blade

# Menginstal dependency frontend dan mengompilasi aset
npm install
npm run build

# Menjalankan migration - Breeze menambahkan tabel users secara otomatis
php artisan migrate
```

Setelah instalasi selesai, Breeze secara otomatis membuat seluruh rute, Controller, dan view yang diperlukan untuk registrasi, login, logout, lupa password, dan verifikasi email, lengkap dengan validasi dan hashing password yang telah mengikuti standar keamanan Laravel.

## 9.3 Struktur yang Dibuat Otomatis oleh Breeze

| Komponen | Lokasi | Fungsi |
|---|---|---|
| Model User | app/Models/User.php | Merepresentasikan tabel users, sudah dilengkapi hashing password otomatis |
| AuthenticatedSessionController | app/Http/Controllers/Auth/ | Menangani proses login dan logout |
| RegisteredUserController | app/Http/Controllers/Auth/ | Menangani proses registrasi pengguna baru |
| View login/register | resources/views/auth/ | Halaman form login dan registrasi siap pakai |
| Middleware auth | routes/web.php | Middleware untuk melindungi rute tertentu (dibahas 9.4) |

Perhatikan bahwa struktur ini mengikuti pola MVC yang telah dipelajari sejak Bab 1 — Model User untuk data, Controller Auth untuk logika, dan view auth untuk tampilan — namun kali ini seluruhnya sudah dibuatkan secara otomatis oleh Breeze, sehingga peserta didik dapat mempelajari kode tersebut sebagai contoh nyata implementasi autentikasi yang baik.

## 9.4 Melindungi Rute dengan Middleware Auth

Middleware adalah lapisan yang memeriksa atau memodifikasi request sebelum mencapai Controller. Middleware `auth` bawaan Laravel memeriksa apakah pengguna telah login sebelum mengizinkan akses ke sebuah rute — apabila belum, pengguna otomatis dialihkan ke halaman login.

```
<?php
// routes/web.php
use App\Http\Controllers\SiswaController;

// Rute publik - dapat diakses tanpa login
Route::get('/', function () {
    return view('welcome');
});

// Rute yang dilindungi - hanya untuk pengguna yang sudah login
Route::middleware(['auth'])->group(function () {
    Route::resource('siswa', SiswaController::class);
});
```

Dengan mengelompokkan rute CRUD data siswa (yang telah dibangun pada Bab 7) ke dalam `Route::middleware(['auth'])->group()`, seluruh fitur tersebut kini hanya dapat diakses oleh pengguna yang telah login — mengintegrasikan langsung kompetensi autentikasi bab ini dengan kompetensi CRUD yang telah dikuasai sebelumnya.

## 9.5 Mengakses Data Pengguna yang Sedang Login

```
<?php
// Pada Controller manapun, mengakses data pengguna yang sedang login
public function dashboard()
{
    $namaPengguna = auth()->user()->name;
    return view('dashboard', compact('namaPengguna'));
}

{{-- Pada Blade view, langsung menggunakan helper auth() --}}
<p>Selamat datang, {{ auth()->user()->name }}!</p>
<a href="{{ route('logout') }}"
   onclick="event.preventDefault(); document.getElementById('logout-
form').submit();">
    Logout
</a>
<form id="logout-form" action="{{ route('logout') }}" method="POST" class="d-none">
    @csrf
</form>
```

Helper `auth()->user()` mengembalikan objek Model User milik pengguna yang sedang login, memungkinkan akses ke seluruh property-nya (name, email, dan kolom lain) langsung dari Controller maupun Blade view, tanpa perlu menyimpan dan memeriksa session secara manual seperti pada Bab 12 buku kelas X.

## 9.6 Fitur Reset Password Bawaan Breeze

Selain login dan registrasi, Breeze turut menyediakan alur reset password yang lengkap: pengguna yang lupa password dapat meminta tautan reset melalui email, mengikuti tautan tersebut, dan menetapkan password baru — seluruhnya tanpa perlu peserta didik membangun sistem pengiriman email dan token keamanan dari nol.

| Langkah Alur Reset Password | Komponen yang Menangani |
|---|---|
| Pengguna mengklik 'Lupa Password?' pada halaman login | View resources/views/auth/forgot-password.blade.php |
| Sistem mengirim email berisi tautan reset dengan token unik | PasswordResetLinkController bawaan Breeze |
| Pengguna mengklik tautan dan memasukkan password baru | View resources/views/auth/reset-password.blade.php |
| Sistem memverifikasi token dan memperbarui password (di-hash otomatis) | NewPasswordController bawaan Breeze |

Pada tahap pengembangan lokal, email yang dikirim dapat diarahkan ke layanan seperti Mailtrap atau ditampilkan langsung pada log Laravel (`storage/logs/laravel.log`) alih-alih benar-benar terkirim, memudahkan pengujian alur reset password tanpa memerlukan server email sungguhan terlebih dahulu.

### *Aktivitas Pembelajaran*

50. Peserta didik menginstal Laravel Breeze pada proyek yang telah dibangun sepanjang bab-bab sebelumnya, kemudian mencoba melakukan registrasi akun baru dan login menggunakan akun tersebut.
51. Peserta didik melindungi seluruh rute CRUD data siswa (Bab 7) menggunakan middleware auth, kemudian memverifikasi bahwa pengguna yang belum login otomatis dialihkan ke halaman login saat mencoba mengakses /siswa.
52. Peserta didik memodifikasi halaman dashboard bawaan Breeze untuk menampilkan sapaan personal dan tautan menuju halaman data siswa, menggunakan auth()->user()->name.

## *Rangkuman*

Laravel Breeze menyediakan sistem autentikasi siap pakai (registrasi, login, logout) yang mengikuti praktik keamanan terbaik, melengkapi pemahaman login manual yang telah dipelajari pada buku kelas X. Middleware auth melindungi rute tertentu agar hanya dapat diakses pengguna yang telah login, dan helper auth()->user() memudahkan akses data pengguna yang sedang aktif. Kompetensi ini menjadi dasar bagi sistem role-based access control yang akan diperdalam pada buku Kelas XII.

## *Soal Latihan / Refleksi*

53. Jelaskan manfaat menggunakan Laravel Breeze dibanding membangun sistem autentikasi secara manual seperti pada buku kelas X.
54. Sebutkan tiga komponen utama yang dibuat otomatis oleh Breeze beserta fungsinya.
55. Jelaskan fungsi middleware auth pada sebuah rute Laravel.
56. Tuliskan kode untuk mengambil nama pengguna yang sedang login pada sebuah Controller.

# BAB 10 — Relasi Antar Tabel

### *Tujuan Pembelajaran*

- Peserta didik mampu menjelaskan konsep relasi database One-to-Many dan Many-to-Many.
- Peserta didik mampu mendefinisikan relasi pada Model Eloquent.
- Peserta didik mampu menampilkan data dari tabel yang saling berelasi pada Blade view.

## 10.1 Mengapa Data Perlu Direlasikan?

Aplikasi nyata jarang hanya memiliki satu tabel yang berdiri sendiri. Studi kasus data siswa yang dibangun sejak Bab 7 pada praktiknya akan berkembang: setiap siswa perlu terhubung dengan kelasnya, dan setiap kelas memiliki banyak mata pelajaran yang diampu oleh guru berbeda. Merelasikan tabel-tabel ini menghindari duplikasi data yang tidak perlu (misalnya menuliskan ulang nama kelas berkali-kali pada setiap baris siswa) dan menjaga konsistensi data.

## 10.2 Relasi One-to-Many (Satu-ke-Banyak)

Relasi One-to-Many terjadi ketika satu baris pada sebuah tabel dapat berhubungan dengan banyak baris pada tabel lain — misalnya satu Kelas memiliki banyak Siswa, namun satu Siswa hanya dimiliki oleh satu Kelas.

```
# Membuat migration untuk tabel kelas, dan menambahkan foreign key pada tabel siswa
php artisan make:migration create_kelas_table
php artisan make:migration add_kelas_id_to_siswa_table

<?php
// Migration tabel kelas
Schema::create('kelas', function (Blueprint $table) {
    $table->id();
    $table->string('nama_kelas'); // contoh: "XI TJKT 1"
    $table->timestamps();
});

// Migration menambahkan foreign key pada tabel siswa
Schema::table('siswa', function (Blueprint $table) {
    $table->foreignId('kelas_id')->constrained('kelas');
});

<?php
// app/Models/Kelas.php
class Kelas extends Model
{
    protected $fillable = ['nama_kelas'];

    public function siswa() {
        return $this->hasMany(Siswa::class); // satu Kelas punya banyak Siswa
    }
}
```

```
// app/Models/Siswa.php
class Siswa extends Model
{
    protected $fillable = ['nama', 'kelas_id', 'nilai'];

    public function kelas() {
        return $this->belongsTo(Kelas::class); // satu Siswa dimiliki satu Kelas
    }
}
```

## 10.3 Menggunakan Relasi pada Query

```
<?php
// Mengambil seluruh siswa pada sebuah kelas tertentu
$kelas = Kelas::find(1);
$daftarSiswaKelas1 = $kelas->siswa; // otomatis mengambil seluruh siswa berelasi

// Mengambil data kelas dari sisi siswa
$siswa = Siswa::find(5);
echo $siswa->kelas->nama_kelas; // menampilkan nama kelas milik siswa ini

// Eager loading - lebih efisien, mengambil siswa beserta data kelasnya sekaligus
$daftarSiswa = Siswa::with('kelas')->get();

{{-- Menampilkan data siswa beserta nama kelasnya pada Blade --}}
@foreach ($daftarSiswa as $siswa)
    <tr>
        <td>{{ $siswa->nama }}</td>
        <td>{{ $siswa->kelas->nama_kelas }}</td>
    </tr>
@endforeach
```

Penggunaan `with('kelas')` disebut **eager loading**, sebuah teknik optimasi yang mengambil data relasi dalam jumlah query yang jauh lebih sedikit dibanding memanggil `$siswa->kelas` satu per satu di dalam perulangan (yang disebut masalah N+1 query) — sebuah pertimbangan performa penting ketika aplikasi menangani data dalam jumlah besar.

## 10.4 Relasi Many-to-Many (Banyak-ke-Banyak)

Relasi Many-to-Many terjadi ketika banyak baris pada sebuah tabel dapat berhubungan dengan banyak baris pada tabel lain — misalnya satu Siswa dapat mengikuti banyak Ekstrakurikuler, dan satu Ekstrakurikuler diikuti oleh banyak Siswa. Relasi jenis ini memerlukan tabel perantara (pivot table) yang menyimpan pasangan id dari kedua tabel.

```
<?php
// Migration tabel pivot, nama tabel mengikuti konvensi Laravel (urutan alfabetis,
tunggal)
Schema::create('ekstrakurikuler_siswa', function (Blueprint $table) {
    $table->foreignId('siswa_id')->constrained('siswa');
    $table->foreignId('ekstrakurikuler_id')->constrained('ekstrakurikuler');
});
```

```php
<?php
// app/Models/Siswa.php
public function ekstrakurikuler() {
    return $this->belongsToMany(Ekstrakurikuler::class);
}

// app/Models/Ekstrakurikuler.php
public function siswa() {
    return $this->belongsToMany(Siswa::class);
}

<?php
// Menambahkan relasi many-to-many
$siswa = Siswa::find(1);
$siswa->ekstrakurikuler()->attach(3); // siswa mengikuti ekstrakurikuler id 3

// Menampilkan seluruh ekstrakurikuler yang diikuti seorang siswa
foreach ($siswa->ekstrakurikuler as $ekskul) {
    echo $ekskul->nama_ekskul;
}
```

## 10.5 Menghitung Data Relasi dengan withCount

Selain menampilkan daftar data relasi, kebutuhan yang sama umumnya adalah menghitung jumlahnya saja — misalnya menampilkan berapa banyak siswa pada setiap kelas tanpa perlu memuat seluruh data siswa tersebut satu per satu. Eloquent menyediakan method `withCount()` untuk kebutuhan ini secara efisien.

```php
<?php
$daftarKelas = Kelas::withCount('siswa')->get();

foreach ($daftarKelas as $kelas) {
    echo $kelas->nama_kelas . ": " . $kelas->siswa_count . " siswa";
}
```

Method `withCount('siswa')` menghasilkan kolom tambahan `siswa_count` pada setiap objek Kelas, dihitung langsung melalui satu query database yang efisien, tanpa perlu memuat seluruh data siswa terlebih dahulu kemudian menghitungnya secara manual menggunakan PHP — sebuah contoh nyata bagaimana Eloquent menyediakan solusi ringkas untuk kebutuhan yang umum ditemui pada aplikasi data akademik.

### *Aktivitas Pembelajaran*

57. Peserta didik membuat tabel 'kelas' dan menghubungkannya dengan tabel 'siswa' menggunakan relasi One-to-Many, kemudian memodifikasi halaman index siswa (Bab 7) untuk menampilkan nama kelas menggunakan relasi Eloquent.
58. Peserta didik menerapkan eager loading (with()) pada query yang telah dibuat, membandingkan jumlah query yang dijalankan dengan dan tanpa eager loading menggunakan Laravel Debugbar (opsional) atau log query.

59. Peserta didik membuat contoh relasi Many-to-Many sederhana antara 'siswa' dan 'ekstrakurikuler', lengkap dengan tabel pivot, dan menampilkan daftar ekstrakurikuler yang diikuti setiap siswa pada sebuah halaman.

## *Rangkuman*

Relasi antar tabel menghindari duplikasi data dan menjaga konsistensi, dengan dua jenis utama: One-to-Many (menggunakan hasMany dan belongsTo, seperti Kelas dan Siswa) dan Many-to-Many (menggunakan belongsToMany dengan tabel pivot, seperti Siswa dan Ekstrakurikuler). Eager loading dengan with() mengoptimalkan performa query saat mengambil data berelasi dalam jumlah besar. Kompetensi ini memperkaya sistem CRUD yang telah dibangun sejak Bab 7 menjadi lebih merepresentasikan struktur data dunia nyata yang saling terhubung.

## *Soal Latihan / Refleksi*

60. Jelaskan perbedaan antara relasi One-to-Many dan Many-to-Many beserta contoh masing-masing.
61. Tuliskan kode Model Eloquent untuk relasi hasMany dan belongsTo antara tabel Guru dan MataPelajaran (satu guru mengampu banyak mata pelajaran).
62. Apa yang dimaksud dengan masalah N+1 query, dan bagaimana eager loading mengatasinya?
63. Jelaskan fungsi tabel pivot pada relasi Many-to-Many.

# BAB 11 — Styling dengan Bootstrap di dalam Blade

### *Tujuan Pembelajaran*

- Peserta didik mampu mengintegrasikan Bootstrap ke dalam proyek Laravel.
- Peserta didik mampu menerapkan komponen Bootstrap (navbar, card, form, table) pada Blade view.
- Peserta didik mampu menyusun tampilan CRUD yang telah dibangun agar lebih rapi dan responsif menggunakan Bootstrap.

## 11.1 Menghubungkan Kembali dengan Kompetensi CSS Kelas X

Pada Bab 9 buku kelas X, CSS ditulis secara manual — selector, properti warna, Flexbox untuk layout. Bootstrap adalah CSS framework yang menyediakan komponen siap pakai (tombol, kartu, navigasi, form) yang telah dirancang secara konsisten dan responsif, mempercepat proses styling secara signifikan tanpa perlu menulis CSS dari nol untuk kebutuhan umum. Pemahaman CSS dasar tetap penting — Bootstrap pada dasarnya adalah kumpulan class CSS yang telah ditulis sebelumnya, dan peserta didik yang memahami CSS akan lebih mudah menyesuaikan atau memodifikasi tampilan Bootstrap sesuai kebutuhan.

## 11.2 Menghubungkan Bootstrap ke Proyek Laravel

```
{{-- Cara termudah: menggunakan CDN pada layout utama --}}
{{-- resources/views/layouts/app.blade.php --}}
<head>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
    ...
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"><
/script>
</body>
```

Menggunakan CDN (Content Delivery Network) adalah cara tercepat untuk mulai menggunakan Bootstrap pada tahap belajar, cukup menyalin tautan <link> dan <script> ke dalam layout utama — konsisten dengan pola pemisahan HTML/CSS/JS yang telah dipelajari sejak Bab 9 buku kelas X, hanya kini file CSS dan JS tersebut disediakan oleh pihak Bootstrap, bukan ditulis sendiri.

## 11.3 Navbar dan Layout Responsif

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container">
    <a class="navbar-brand" href="/">Sistem Data Siswa</a>
    <div class="navbar-nav">
      <a class="nav-link" href="{{ route('siswa.index') }}">Data Siswa</a>
      <a class="nav-link" href="{{ route('dashboard') }}">Dashboard</a>
    </div>
```

```
  </div>
</nav>

<div class="container mt-4">
    @yield('content')
</div>
```

Class `navbar-expand-lg` membuat navigasi otomatis berubah menjadi menu hamburger pada layar kecil (ponsel), menerapkan konsep responsive design secara otomatis tanpa perlu menulis media query CSS secara manual — jauh lebih efisien dibanding pendekatan Flexbox manual pada Bab 9 buku kelas X untuk kebutuhan layout yang kompleks.

## 11.4 Mempercantik Tabel dan Form CRUD

```
{{-- Tabel data siswa dengan styling Bootstrap --}}
<table class="table table-striped table-hover">
    <thead class="table-dark">
        <tr><th>Nama</th><th>Kelas</th><th>Nilai</th><th>Aksi</th></tr>
    </thead>
    <tbody>
    @foreach ($daftarSiswa as $siswa)
        <tr>
            <td>{{ $siswa->nama }}</td>
            <td>{{ $siswa->kelas->nama_kelas }}</td>
            <td>{{ $siswa->nilai }}</td>
            <td>
                <a href="{{ route('siswa.edit', $siswa->id) }}" class="btn btn-sm
btn-warning">Edit</a>
                <button type="submit" form="hapus-{{ $siswa->id }}" class="btn btn-
sm btn-danger">Hapus</button>
            </td>
        </tr>
    @endforeach
    </tbody>
</table>

{{-- Form dengan styling Bootstrap --}}
<div class="mb-3">
    <label class="form-label">Nama Siswa</label>
    <input type="text" name="nama" class="form-control @error('nama') is-invalid
@enderror" value="{{ old('nama') }}">
    @error('nama')
        <div class="invalid-feedback">{{ $message }}</div>
    @enderror
</div>
<button type="submit" class="btn btn-primary">Simpan</button>
```

Perhatikan bagaimana directive `@error('nama')` yang telah dipelajari pada Bab 8 kini dikombinasikan dengan class Bootstrap `is-invalid` dan `invalid-feedback`, menghasilkan tampilan validasi form yang jauh lebih profesional dan mudah dipahami pengguna dibanding sekadar teks merah polos.

## 11.5 Komponen Card dan Alert untuk Dashboard

Selain tabel dan form, komponen **card** dan **alert** Bootstrap sangat berguna untuk menyusun halaman dashboard yang menampilkan ringkasan informasi, seperti jumlah total siswa atau jumlah kelas yang telah dibuat, tanpa perlu menyajikannya dalam bentuk tabel.

```
{{-- Ringkasan statistik pada dashboard menggunakan card --}}
<div class="row">
    <div class="col-md-4">
        <div class="card text-white bg-primary mb-3">
            <div class="card-body">
                <h5 class="card-title">Total Siswa</h5>
                <p class="card-text fs-3">{{ $totalSiswa }}</p>
            </div>
        </div>
    </div>
    <div class="col-md-4">
        <div class="card text-white bg-success mb-3">
            <div class="card-body">
                <h5 class="card-title">Total Kelas</h5>
                <p class="card-text fs-3">{{ $totalKelas }}</p>
            </div>
        </div>
    </div>
</div>
```

Class `row` dan `col-md-4` merupakan bagian dari sistem grid Bootstrap yang membagi lebar halaman menjadi 12 kolom — `col-md-4` berarti sebuah elemen mengambil 4 dari 12 kolom (sepertiga lebar halaman) pada layar berukuran medium ke atas, sebuah pendekatan layout yang melengkapi Flexbox yang telah dipelajari pada Bab 9 buku kelas X dengan sistem yang lebih terstruktur untuk tata letak berbasis kolom.

## *Aktivitas Pembelajaran*

64. Peserta didik menghubungkan Bootstrap ke proyek Laravel yang telah dibangun sepanjang bab-bab sebelumnya menggunakan CDN, kemudian memperbarui layout utama dengan navbar Bootstrap.
65. Peserta didik mempercantik seluruh halaman CRUD data siswa (index, create, edit) menggunakan komponen table, form, dan button Bootstrap.
66. Peserta didik menguji tampilan aplikasi pada ukuran layar berbeda (menggunakan Developer Tools mode responsif pada browser) untuk memverifikasi navbar dan layout beradaptasi dengan baik pada layar kecil.

## *Rangkuman*

Bootstrap mempercepat proses styling aplikasi Laravel melalui komponen siap pakai (navbar, table, form, button) yang responsif secara otomatis, melengkapi pemahaman CSS manual yang telah dikuasai pada buku kelas X. Integrasi Bootstrap paling mudah dilakukan melalui CDN pada layout utama Blade, dan dapat dikombinasikan langsung dengan directive Blade seperti @error untuk menghasilkan tampilan validasi form yang profesional. Aplikasi CRUD yang telah dibangun sejak Bab 7 kini memiliki tampilan yang jauh lebih siap digunakan secara nyata.

### *Soal Latihan / Refleksi*

67. Jelaskan keuntungan menggunakan Bootstrap dibanding menulis seluruh CSS secara manual untuk aplikasi berskala menengah.
68. Sebutkan cara paling sederhana menghubungkan Bootstrap ke sebuah proyek Laravel.
69. Jelaskan bagaimana class navbar-expand-lg pada Bootstrap membantu penerapan responsive design.
70. Bagaimana directive @error pada Blade dapat dikombinasikan dengan class Bootstrap untuk menampilkan validasi form yang lebih baik?

# BAB 12 — Git dan GitHub untuk Kolaborasi

### *Tujuan Pembelajaran*

- Peserta didik mampu menjelaskan konsep version control dan manfaatnya dalam pengembangan perangkat lunak.
- Peserta didik mampu menggunakan perintah dasar Git untuk melacak perubahan kode.
- Peserta didik mampu mengunggah proyek Laravel ke GitHub dan berkolaborasi menggunakan branch.

## 12.1 Mengapa Version Control Diperlukan?

Sepanjang buku ini, proyek Laravel yang dibangun terus bertambah kompleks — dari instalasi dasar hingga CRUD dengan relasi dan autentikasi. Tanpa sistem pelacakan perubahan yang baik, seorang programmer sulit mengingat perubahan apa saja yang telah dilakukan, sulit mengembalikan kode ke versi sebelumnya apabila terjadi kesalahan, dan hampir mustahil berkolaborasi dengan programmer lain tanpa saling menimpa (overwrite) pekerjaan satu sama lain. **Git** adalah sistem version control terdistribusi yang menjadi standar industri untuk mengatasi seluruh masalah tersebut, dan **GitHub** adalah platform daring yang menghosting repositori Git sehingga dapat diakses dan dikolaborasikan dari mana saja.

## 12.2 Perintah Dasar Git

```
# Inisialisasi repositori Git baru pada proyek Laravel
git init

# Memeriksa status perubahan file
git status

# Menambahkan file ke staging area (siap untuk di-commit)
git add .

# Menyimpan perubahan sebagai sebuah commit dengan pesan deskriptif
git commit -m "Menambahkan fitur CRUD data siswa"

# Melihat riwayat seluruh commit yang pernah dibuat
git log --oneline
```

| Istilah | Penjelasan |
|---|---|
| Repository (repo) | Folder proyek yang dilacak perubahannya oleh Git |
| Commit | Sebuah 'titik simpan' yang mencatat perubahan tertentu beserta pesan penjelasnya |
| Staging Area | Area sementara berisi perubahan yang siap untuk di-commit |
| Branch | Jalur pengembangan terpisah, memungkinkan fitur baru dikembangkan tanpa mengganggu kode utama |

## 12.3 File .gitignore pada Proyek Laravel

Tidak seluruh file pada proyek Laravel perlu diunggah ke Git — file `.env` (berisi kredensial database, sebagaimana diperingatkan pada Bab 3) dan folder `vendor/` (berisi paket Composer yang dapat diinstal ulang kapan saja) sebaiknya dikecualikan dari pelacakan Git. Laravel secara otomatis menyediakan file `.gitignore` yang telah dikonfigurasi dengan pengecualian yang tepat sejak awal instalasi.

```
# Cuplikan isi .gitignore bawaan Laravel
/vendor
/node_modules
.env
.env.backup
/public/hot
/public/storage
/storage/*.key
```

Mengabaikan file `.env` bukan sekadar kerapian, melainkan **praktik keamanan penting** — apabila file ini tidak sengaja terunggah ke repositori publik di GitHub, siapa pun dapat melihat kredensial database dan kunci aplikasi, membuka celah keamanan serius yang sejalan dengan prinsip perlindungan kredensial yang telah dipelajari pada mata pelajaran Keamanan Jaringan.

## 12.4 Mengunggah Proyek ke GitHub

```
# Membuat repositori baru di GitHub terlebih dahulu (melalui website), lalu:
git remote add origin https://github.com/username/sistem-data-siswa.git
git branch -M main
git push -u origin main
```

Perintah `git remote add origin` menghubungkan repositori lokal dengan repositori di GitHub, dan `git push` mengunggah seluruh commit yang telah dibuat ke repositori tersebut, membuat kode dapat diakses secara daring — langkah pertama membangun portofolio publik yang akan dibahas lebih lanjut pada buku Kelas XII.

## 12.5 Branch: Mengembangkan Fitur Tanpa Mengganggu Kode Utama

Branch memungkinkan pengembangan sebuah fitur baru dilakukan pada jalur terpisah dari kode utama (biasanya branch bernama `main`), sehingga kode utama tetap stabil selama fitur baru sedang dikerjakan dan belum tentu selesai atau bebas kesalahan.

```
# Membuat dan berpindah ke branch baru untuk mengembangkan fitur ekstrakurikuler
git checkout -b fitur-ekstrakurikuler

# ... melakukan perubahan kode, add, dan commit seperti biasa ...

# Setelah fitur selesai dan teruji, kembali ke branch utama
git checkout main
```

```
# Menggabungkan branch fitur ke branch utama
git merge fitur-ekstrakurikuler
```

Pola kerja menggunakan branch ini adalah standar praktik dalam tim pengembangan profesional — setiap anggota tim dapat mengerjakan fitur berbeda secara paralel pada branch masing-masing tanpa saling mengganggu, kemudian menggabungkan hasil kerja mereka setelah diuji dan direview, sebuah kompetensi kolaborasi yang akan semakin relevan ketika mengerjakan proyek berkelompok pada bab-bab selanjutnya.

### *Aktivitas Pembelajaran*

71. Peserta didik menginisialisasi Git pada proyek Laravel yang telah dibangun sepanjang buku ini, membuat beberapa commit dengan pesan yang deskriptif untuk setiap fitur yang telah diselesaikan (CRUD, autentikasi, relasi, styling).
72. Peserta didik membuat akun GitHub (jika belum memiliki), membuat repositori baru, dan mengunggah proyek Laravel mereka, memverifikasi bahwa file .env tidak ikut terunggah.
73. Peserta didik berlatih membuat branch baru untuk mengembangkan satu fitur tambahan kecil (misalnya menambah kolom baru pada data siswa), kemudian menggabungkannya kembali ke branch main menggunakan git merge.

### *Rangkuman*

Git adalah sistem version control yang melacak perubahan kode melalui commit, memungkinkan pengembangan paralel menggunakan branch, dan mencegah kehilangan pekerjaan. GitHub menghosting repositori Git secara daring, memungkinkan kolaborasi dan menjadi fondasi portofolio publik. File .gitignore mengecualikan file sensitif seperti .env dari pelacakan Git sebagai praktik keamanan penting. Kompetensi ini melengkapi seluruh proyek Laravel yang telah dibangun sepanjang buku ini, siap dipresentasikan sebagai proyek tengah semester pada bab berikutnya.

### *Soal Latihan / Refleksi*

74. Jelaskan perbedaan antara git add dan git commit dalam alur kerja Git.
75. Mengapa file .env tidak boleh diunggah ke repositori Git publik, dan bagaimana Laravel mencegah hal ini secara default?
76. Jelaskan manfaat penggunaan branch dalam pengembangan sebuah fitur baru pada proyek Laravel.
77. Tuliskan urutan perintah Git untuk mengunggah sebuah proyek Laravel yang baru diinisialisasi ke repositori GitHub yang baru dibuat.

# BAB 13 — Vibecoding dengan Laravel: AI sebagai Pair Programmer

### *Tujuan Pembelajaran*

- Peserta didik mampu menerapkan vibecoding secara lebih matang dalam konteks pengembangan aplikasi Laravel.
- Peserta didik mampu menggunakan AI untuk mempercepat pembuatan boilerplate code (migration, model, controller) sambil tetap memvalidasi hasilnya.
- Peserta didik mampu menggunakan AI sebagai alat bantu debugging dan code review.

## 13.1 Dari Mengenal Menuju Menggunakan: Evolusi Vibecoding

Bab 13 buku Pemrograman Web Dasar kelas X memperkenalkan konsep vibecoding — kolaborasi antara programmer dan AI — pada level pengenalan: apa itu vibecoding, cara menyusun prompt yang efektif, dan sikap kritis dasar. Bab ini melangkah lebih jauh: menerapkan vibecoding secara nyata dalam konteks pengembangan aplikasi Laravel yang jauh lebih kompleks, di mana AI dapat membantu mempercepat pekerjaan repetitif (seperti menulis migration atau boilerplate Controller) sehingga programmer dapat fokus pada logika bisnis yang lebih penting dan spesifik terhadap kebutuhan aplikasi.

Perbedaan mendasar antara vibecoding tingkat pemula dan tingkat lanjut terletak pada **konteks yang diberikan kepada AI**. Pada tingkat pemula, permintaan cenderung berdiri sendiri ('buatkan form login'). Pada tingkat lanjut sebagaimana dibahas pada bab ini, permintaan kepada AI menyertakan konteks arsitektur yang sedang dikerjakan (pola MVC, konvensi penamaan Laravel, struktur database yang sudah ada), menghasilkan kode yang jauh lebih sesuai dan konsisten dengan proyek yang sedang dibangun.

## 13.2 Menggunakan AI untuk Mempercepat Boilerplate Code

Boilerplate code adalah kode dengan pola berulang yang hampir selalu sama strukturnya di berbagai bagian aplikasi — seperti migration, Model dasar, atau Controller resource yang telah dipelajari pada Bab 6 dan 7. Kode semacam ini sangat cocok dipercepat menggunakan AI, karena polanya dapat diprediksi dan mudah diverifikasi kebenarannya.

| Prompt Efektif untuk Laravel | Alasan Efektif |
|---|---|
| "Buatkan migration Laravel untuk tabel 'buku_perpustakaan' dengan kolom judul (string), penulis (string), tahun_terbit (integer), dan stok (integer, default 1), lengkap dengan timestamps" | Menyebutkan nama tabel, kolom, tipe data, dan constraint secara spesifik — AI tidak perlu menebak |
| "Buatkan Model Eloquent bernama Buku yang terhubung ke tabel buku_perpustakaan, dengan fillable judul, penulis, tahun_terbit, dan stok" | Menyertakan nama tabel yang sudah tidak sesuai konvensi default, mencegah AI membuat asumsi keliru |

| Prompt Efektif untuk Laravel | Alasan Efektif |
|---|---|
| "Buatkan Controller resource untuk Model Buku, mengikuti pola yang sama seperti SiswaController yang sudah saya buat sebelumnya" | Merujuk pola yang sudah ada pada proyek, menghasilkan konsistensi gaya kode |

Setelah AI menghasilkan kode, langkah **wajib** berikutnya adalah memverifikasi: apakah nama tabel pada migration sudah benar? Apakah kolom fillable pada Model sudah sesuai? Apakah Controller yang dihasilkan benar-benar mengikuti pola resource seperti pada Bab 7? Tanpa verifikasi ini, kesalahan kecil (misalnya nama kolom yang typo) dapat menyebabkan error yang membingungkan ketika aplikasi dijalankan.

## 13.3 AI sebagai Alat Bantu Debugging

Ketika menghadapi error yang membingungkan — sebuah pengalaman yang hampir pasti dialami setiap kali membangun fitur baru — AI dapat menjadi alat bantu debugging yang efektif, dengan syarat pesan error dan konteks kode disampaikan secara lengkap.

```
// Contoh error yang sering ditemui pemula Laravel:
// "SQLSTATE[42S22]: Column not found: 1054 Unknown column 'kelas_id' in 'field
list'"

// Prompt yang efektif kepada AI:
// "Saya mendapat error 'Unknown column kelas_id in field list' saat menjalankan
// $siswa->kelas_id = 1; $siswa->save(); pada Laravel. Berikut migration tabel siswa
// saya: [tempelkan isi migration]. Apa penyebab errornya dan bagaimana
memperbaikinya?"
```

Pola debugging yang efektif dengan AI mengikuti prinsip yang sama dengan debugging manual yang telah dipelajari sejak konsep troubleshooting berlapis pada mata pelajaran Jaringan Dasar: sertakan **gejala** (pesan error persis), **konteks** (kode yang relevan, bukan seluruh proyek), dan **apa yang sudah dicoba** — semakin lengkap informasi yang diberikan, semakin akurat diagnosis yang dihasilkan AI, dan semakin mudah peserta didik memverifikasi apakah solusi yang disarankan benar-benar menyelesaikan akar masalah, bukan sekadar gejalanya.

## 13.4 AI untuk Code Review Mandiri

Selain menghasilkan kode baru, AI dapat dimanfaatkan untuk meninjau (review) kode yang telah ditulis sendiri, mengidentifikasi potensi masalah yang mungkin terlewat, sebuah kebiasaan yang meniru praktik code review yang lazim dilakukan dalam tim pengembangan profesional sebelum kode digabungkan ke branch utama (sebagaimana konsep branch pada Bab 12).

- Minta AI memeriksa apakah Controller yang ditulis sudah menerapkan validasi yang memadai (menghubungkan kembali ke Bab 8).
- Minta AI mengidentifikasi potensi query yang tidak efisien, misalnya kasus N+1 query yang telah dibahas pada Bab 10.

- Minta AI memeriksa apakah ada celah keamanan dasar, seperti kolom yang seharusnya tidak fillable namun tidak dibatasi.
- Selalu validasi saran AI dengan menjalankan dan menguji kode secara langsung — AI dapat memberikan saran yang terdengar masuk akal namun sesungguhnya keliru untuk konteks spesifik proyek.

## 13.5 Batasan Vibecoding pada Proyek Berskala Tim

Ketika bekerja dalam kelompok pada proyek tengah semester (Bab 14), penggunaan AI perlu dikoordinasikan agar tidak menghasilkan gaya kode yang tidak konsisten antar anggota tim — misalnya satu anggota menggunakan pola penamaan variabel yang berbeda dari kesepakatan tim hanya karena mengikuti saran AI secara mentah. Praktik yang baik adalah menyepakati konvensi penamaan dan struktur kode di awal proyek (sejalan dengan pola MVC dan resource routing yang telah konsisten dipakai sepanjang buku ini), kemudian menggunakan AI sebagai alat bantu yang tetap mengikuti konvensi tersebut, bukan sebagai sumber keputusan arsitektur yang menggantikan diskusi tim.

### *Aktivitas Pembelajaran*

78. Peserta didik meminta AI membuatkan migration, Model, dan Controller resource untuk sebuah entitas baru (misalnya 'Ekstrakurikuler' atau 'Ruangan'), kemudian memverifikasi dan menjalankan kode tersebut, mencatat penyesuaian apa saja yang perlu dilakukan secara manual.
79. Peserta didik dengan sengaja membuat sebuah error umum pada kode Laravel (misalnya salah nama kolom atau lupa menjalankan migration), kemudian menggunakan AI untuk membantu mendiagnosis dan memperbaikinya, mendokumentasikan proses debugging tersebut.
80. Peserta didik meminta AI melakukan code review terhadap salah satu Controller yang telah dibuat sepanjang buku ini, mengevaluasi apakah saran yang diberikan valid dan relevan untuk diterapkan.

### *Rangkuman*

Vibecoding pada tingkat lanjut menekankan pemberian konteks arsitektur yang jelas kepada AI, mempercepat pembuatan boilerplate code (migration, Model, Controller) yang tetap wajib diverifikasi kebenarannya. AI juga menjadi alat bantu debugging yang efektif ketika disertai gejala, konteks kode, dan langkah yang sudah dicoba, serta dapat dimanfaatkan untuk code review mandiri terhadap kode sendiri. Pada proyek berskala tim, penggunaan AI perlu dikoordinasikan agar tetap konsisten dengan konvensi yang disepakati bersama, sebuah kompetensi yang akan langsung dipraktikkan pada proyek tengah semester berikutnya.

### *Soal Latihan / Refleksi*

81. Jelaskan perbedaan mendekati vibecoding pada tingkat pemula (Kelas X) dengan tingkat lanjut (Kelas XI) yang dibahas pada bab ini.
82. Sebutkan tiga elemen yang sebaiknya disertakan saat meminta bantuan AI untuk debugging sebuah error Laravel.
83. Mengapa boilerplate code seperti migration dan Controller resource dianggap cocok dipercepat menggunakan AI?

84. Jelaskan risiko penggunaan AI yang tidak terkoordinasi pada proyek yang dikerjakan secara tim.

# BAB 14 — Proyek Akhir Semester: Aplikasi CRUD Laravel Terintegrasi

Bab ini menjadi wadah bagi peserta didik untuk mengintegrasikan seluruh kompetensi yang telah dipelajari sepanjang buku ini — instalasi Laravel, OOP, routing dan Controller, Blade, Eloquent dan migration, CRUD lengkap, validasi, autentikasi, relasi antar tabel, styling Bootstrap, hingga version control Git — menjadi satu proyek aplikasi web yang utuh dan siap dipresentasikan.

### *Tujuan Pembelajaran*

- Peserta didik mampu merancang dan membangun aplikasi Laravel dengan minimal dua entitas berelasi.
- Peserta didik mampu menerapkan autentikasi dan validasi pada aplikasi yang dibangun.
- Peserta didik mampu mendokumentasikan dan mempresentasikan proyek menggunakan repositori GitHub.

## 14.1 Ketentuan Proyek

Peserta didik, secara individu atau berkelompok (2–3 orang, mengikuti kebijakan guru), merancang dan membangun sebuah aplikasi Laravel dengan tema bebas namun harus merepresentasikan kebutuhan nyata — misalnya sistem inventaris laboratorium sekolah, sistem peminjaman buku perpustakaan, sistem pendaftaran ekstrakurikuler, atau sistem pengelolaan data prestasi siswa. Tema yang dipilih sebaiknya memiliki minimal dua entitas data yang saling berelasi, sejalan dengan kompetensi Bab 10.

| Komponen Wajib | Bab Rujukan |
|---|---|
| Minimal dua Model dengan migration yang sesuai | Bab 3, 6 |
| Relasi antar Model (One-to-Many minimal) | Bab 10 |
| CRUD lengkap (Create, Read, Update, Delete) pada minimal satu entitas utama | Bab 7 |
| Validasi pada form tambah dan edit data | Bab 8 |
| Sistem login menggunakan Laravel Breeze, rute CRUD dilindungi middleware auth | Bab 9 |
| Tampilan menggunakan Bootstrap, responsif pada layar kecil | Bab 11 |

| Komponen Wajib | Bab Rujukan |
|---|---|
| Riwayat commit Git yang mencerminkan progres pengerjaan bertahap, diunggah ke GitHub | Bab 12 |

## 14.2 Tahapan Pengerjaan yang Disarankan

85. Rancang skema data: tentukan entitas apa saja yang dibutuhkan, kolom masing-masing, dan bagaimana relasinya (gambar diagram relasi sederhana sebelum mulai coding).
86. Buat proyek Laravel baru, inisialisasi Git sejak awal, dan lakukan commit pertama sebelum menulis fitur apa pun.
87. Buat seluruh migration dan Model sesuai rancangan, commit setelah struktur database selesai dan teruji dengan php artisan migrate.
88. Bangun CRUD untuk entitas utama menggunakan resource routing dan Controller, commit setelah setiap fitur (create, edit, delete) berhasil diuji.
89. Tambahkan validasi pada seluruh form, commit terpisah untuk perubahan ini.
90. Instal Breeze dan lindungi rute CRUD dengan middleware auth, commit setelah login/logout berfungsi.
91. Percantik tampilan menggunakan Bootstrap, commit setelah styling selesai.
92. Uji menyeluruh seluruh alur aplikasi dari sudut pandang pengguna baru (registrasi hingga menghapus data), perbaiki bug yang ditemukan.
93. Unggah proyek final ke GitHub, pastikan .env tidak ikut terunggah, dan tulis dokumentasi singkat pada README.

Melakukan commit secara bertahap pada setiap tahapan (bukan satu commit besar di akhir) mencerminkan praktik pengembangan yang baik sekaligus memudahkan guru menilai progres pengerjaan melalui riwayat commit pada GitHub, bukan hanya menilai hasil akhirnya saja.

## 14.3 Panduan Penggunaan AI (Vibecoding) pada Proyek Ini

Sejalan dengan Bab 13, peserta didik diperbolehkan dan bahkan dianjurkan menggunakan AI untuk mempercepat penulisan boilerplate code pada proyek ini, dengan syarat mendokumentasikan bagian mana yang dibantu AI beserta penjelasan pemahaman pribadi terhadap kode tersebut — sejalan dengan prinsip etika akademik yang telah ditekankan sejak buku kelas X. Dokumentasi ini dapat berupa catatan singkat pada README atau laporan terpisah yang menyebutkan, misalnya, 'Migration dan Model untuk entitas Ekstrakurikuler dibuat dengan bantuan AI, kemudian disesuaikan manual pada bagian nama kolom dan aturan validasi.'

## 14.4 Rubrik Penilaian

| Aspek | Bobot | Indikator |
|---|---|---|
| Struktur Database & Relasi | 20% | Migration rapi, relasi antar tabel diterapkan dengan benar dan digunakan secara nyata pada aplikasi |
| Fungsionalitas CRUD | 25% | Seluruh operasi Create, Read, Update, Delete berjalan tanpa error, resource routing diterapkan |
| Validasi & Autentikasi | 20% | Validasi mencegah data tidak valid tersimpan, rute CRUD terlindungi middleware auth |
| Tampilan (UI) | 15% | Menggunakan Bootstrap, rapi, responsif, pesan error/sukses ditampilkan dengan jelas |
| Version Control | 10% | Riwayat commit bertahap dan deskriptif, .env tidak terunggah, README informatif |
| Presentasi & Pemahaman | 10% | Mampu menjelaskan alur MVC aplikasi sendiri, termasuk bagian yang dibantu AI |

## *Soal Latihan / Refleksi*

94. Jelaskan mengapa merancang skema data dan relasi sebelum mulai menulis kode (langkah 1 pada tahapan pengerjaan) penting dilakukan lebih dahulu.
95. Sebutkan tiga alasan mengapa commit bertahap lebih baik dibanding satu commit besar di akhir proyek.
96. Jelaskan bagaimana kamu akan mendokumentasikan penggunaan AI pada proyekmu sendiri, sesuai prinsip etika akademik yang telah dipelajari.
97. Refleksikan: dari seluruh Bab 1–13, kompetensi mana yang menurutmu paling menantang untuk diterapkan pada proyek akhir semester ini, dan strategi apa yang kamu gunakan untuk mengatasinya?

# BAB 15 — Rangkuman dan Evaluasi Akhir

Bab ini menjadi bahan konsolidasi menyeluruh atas seluruh materi Pemrograman Web Lanjutan Kelas XI, mencakup transisi dari PHP native menuju framework (Bab 1–3), pola MVC penuh (Bab 4–7), penguatan aplikasi (Bab 8–12), dan vibecoding tingkat lanjut (Bab 13), yang seluruhnya bermuara pada proyek akhir semester (Bab 14). Bab ini dapat digunakan sebagai bahan review menjelang asesmen sumatif.

## 15.1 Peta Keterkaitan Antar Bab

Buku ini dibangun secara berlapis: pemahaman OOP (Bab 2) menjadi prasyarat mutlak sebelum instalasi dan struktur Laravel (Bab 3) dapat dipahami sepenuhnya, karena seluruh komponen Laravel (Model, Controller) adalah class PHP. Routing dan Controller (Bab 4) beserta Blade (Bab 5) membentuk separuh pola MVC (Controller dan View), sementara Eloquent dan Migration (Bab 6) melengkapi separuh lainnya (Model) — ketiganya bertemu secara utuh pada Bab 7 (CRUD Penuh).

Bab 8 (Validasi) hingga Bab 12 (Git) memperkuat proyek CRUD tersebut dari berbagai sisi: keandalan data (validasi), keamanan akses (autentikasi), kompleksitas data dunia nyata (relasi), profesionalitas tampilan (Bootstrap), dan kolaborasi/pelacakan perubahan (Git). Bab 13 (vibecoding lanjut) melengkapi seluruh kompetensi teknis dengan keterampilan bekerja secara produktif bersama AI, yang seluruhnya dipraktikkan secara terintegrasi pada proyek akhir semester di Bab 14.

## 15.2 Rekapitulasi Capaian Pembelajaran

| Bab | Kompetensi Utama | Elemen MVC Terkait |
|---|---|---|
| 1 | Transisi PHP native ke framework, konsep MVC | Pengantar |
| 2 | PHP OOP: class, object, constructor, inheritance | Prasyarat |
| 3 | Instalasi Laravel, struktur proyek, Artisan CLI | Pengantar |
| 4 | Routing dan Controller | Controller |
| 5 | Blade templating, layout, komponen | View |
| 6 | Migration dan Eloquent ORM | Model |
| 7 | CRUD penuh terintegrasi | Model-View-Controller |
| 8 | Validasi dan Form Request | Controller |
| 9 | Autentikasi dengan Laravel Breeze | Model-Controller |
| 10 | Relasi antar tabel (One-to-Many, Many-to-Many) | Model |
| 11 | Styling dengan Bootstrap | View |

| Bab | Kompetensi Utama | Elemen MVC Terkait |
|---|---|---|
| 12 | Version control dengan Git dan GitHub | Pendukung |
| 13 | Vibecoding tingkat lanjut dengan Laravel | Pendukung |

## 15.3 Arah Pembelajaran Lanjutan

Setelah menuntaskan buku ini, peserta didik telah memiliki fondasi kuat dalam membangun aplikasi web terstruktur menggunakan Laravel — kompetensi yang setara dengan kebutuhan dasar seorang junior web developer di dunia kerja. Buku Pemrograman Web Lanjutan Kelas XII akan melanjutkan kompetensi ini ke tingkat yang lebih matang: membangun dan mengonsumsi REST API, menerapkan keamanan aplikasi tingkat lanjut (mencegah XSS, CSRF, dan celah keamanan lain yang telah disinggung sepanjang buku ini), autentikasi berjenjang berbasis peran (role-based access control), deployment aplikasi Laravel ke server produksi, hingga membangun portofolio dan kesiapan menghadapi dunia kerja sesungguhnya.

### *Soal Latihan / Refleksi*

98. Jelaskan bagaimana pemahaman OOP pada Bab 2 menjadi prasyarat bagi seluruh materi Laravel pada bab-bab selanjutnya.
99. Rancang skema data sederhana (nama tabel, kolom, dan relasi) untuk sebuah aplikasi pemesanan lapangan futsal sekolah, menerapkan prinsip perencanaan yang telah dipelajari pada Bab 14.
100. Jelaskan mengapa CRUD, validasi, autentikasi, dan relasi antar tabel dianggap sebagai kompetensi fondasi yang hampir selalu dibutuhkan pada aplikasi web apa pun, terlepas dari temanya.
101. Refleksikan: bagaimana pemahaman Git dan GitHub yang dipelajari pada buku ini akan bermanfaat ketika kamu bekerja dalam tim pengembangan perangkat lunak di masa depan?

# Glosarium

**Artisan** — command-line interface bawaan Laravel untuk mempercepat pekerjaan pengembangan.

**Blade** — engine templating bawaan Laravel untuk menyusun tampilan (view).

**Boilerplate Code** — kode dengan pola berulang yang hampir selalu sama strukturnya.

**Branch** — jalur pengembangan terpisah pada Git, memungkinkan fitur dikembangkan tanpa mengganggu kode utama.

**Commit** — sebuah titik simpan pada Git yang mencatat perubahan tertentu beserta pesan penjelas.

**Composer** — pengelola paket (dependency manager) untuk PHP, digunakan menginstal Laravel.

**Controller** — komponen MVC yang menjembatani permintaan pengguna dengan Model dan View.

**Eager Loading** — teknik mengambil data relasi sekaligus untuk mengoptimalkan jumlah query.

**Eloquent** — ORM (Object-Relational Mapping) bawaan Laravel.

**Encapsulation** — prinsip OOP yang menyembunyikan detail internal sebuah class melalui visibility.

**Foreign Key** — kolom yang menghubungkan sebuah tabel dengan tabel lain dalam relasi database.

**Git** — sistem version control terdistribusi untuk melacak perubahan kode.

**Inheritance** — konsep OOP di mana sebuah class dapat mewarisi property dan method dari class lain.

**Laravel Breeze** — paket autentikasi siap pakai bawaan Laravel.

**Middleware** — lapisan yang memeriksa/memodifikasi request sebelum mencapai Controller.

**Migration** — cara mendefinisikan struktur tabel database menggunakan kode PHP.

**Model** — komponen MVC yang mengelola data dan interaksi dengan database.

**MVC** — Model-View-Controller, pola arsitektur yang memisahkan data, tampilan, dan logika penghubung.

**ORM** — Object-Relational Mapping, teknik berinteraksi dengan database menggunakan kode berorientasi objek.

**Resource Routing** — cara mendaftarkan ketujuh rute CRUD standar Laravel dalam satu baris kode.

**Vibecoding** — gaya pengembangan perangkat lunak berbasis kolaborasi intensif dengan AI.

**View** — komponen MVC yang menampilkan antarmuka kepada pengguna.

# Lampiran: Referensi Cepat Perintah Artisan

| Perintah | Fungsi |
|---|---|
| php artisan serve | Menjalankan server pengembangan lokal |
| php artisan make:model NamaModel -m | Membuat Model sekaligus migration terkait |
| php artisan make:controller NamaController --resource | Membuat Controller dengan seluruh method CRUD |
| php artisan make:migration nama_migration | Membuat file migration baru |
| php artisan make:request NamaRequest | Membuat Form Request untuk validasi terpisah |
| php artisan migrate | Menjalankan seluruh migration yang belum dijalankan |
| php artisan migrate:rollback | Membatalkan migration terakhir |
| php artisan migrate:fresh | Menghapus seluruh tabel dan menjalankan ulang migration dari awal |
| php artisan route:list | Menampilkan seluruh rute yang terdaftar |
| php artisan tinker | Membuka konsol interaktif untuk mencoba kode Laravel langsung |
| php artisan breeze:install blade | Menginstal Laravel Breeze dengan stack Blade |

# Lampiran: Referensi Cepat Eloquent dan Blade

## Query Eloquent Umum

| Kode | Fungsi |
|---|---|
| Model::all() | Mengambil seluruh data |
| Model::find($id) | Mengambil satu data berdasarkan id |
| Model::findOrFail($id) | Sama seperti find(), namun otomatis error 404 jika tidak ditemukan |
| Model::where('kolom', 'nilai')->get() | Memfilter data berdasarkan kondisi |
| Model::orderBy('kolom', 'asc')->get() | Mengurutkan data |
| Model::with('relasi')->get() | Eager loading data relasi |
| Model::create([...]) | Menambahkan data baru |
| $model->update([...]) | Memperbarui data yang sudah ada |
| $model->delete() | Menghapus data |

## Directive Blade Umum

| Directive | Fungsi |
|---|---|
| {{ $variabel }} | Menampilkan data dengan proteksi XSS otomatis |
| @if / @elseif / @else / @endif | Struktur percabangan |
| @foreach / @endforeach | Perulangan menampilkan data |
| @forelse / @empty / @endforelse | Perulangan dengan penanganan data kosong |
| @extends / @section / @yield / @endsection | Sistem layout dan pewarisan template |
| @csrf | Token keamanan pada form |
| @method('PUT'/'DELETE') | Mensimulasikan HTTP verb selain GET/POST pada form |
| @error('kolom') ... @enderror | Menampilkan pesan error validasi |

# Lampiran: Ide Proyek Tambahan untuk Latihan Mandiri

Lampiran ini menyediakan ide proyek CRUD tambahan yang dapat dikerjakan sebagai latihan mandiri di luar proyek akhir semester, masing-masing mengintegrasikan kompetensi Bab 1–13.

## Proyek A: Sistem Peminjaman Alat Laboratorium

Bangun aplikasi dengan entitas Alat (nama, kode, kondisi, stok) dan Peminjaman (siapa meminjam, alat apa, tanggal pinjam, tanggal kembali), dengan relasi One-to-Many antara Alat dan Peminjaman. Tambahkan validasi agar stok tidak dapat dipinjam melebihi jumlah yang tersedia.

## Proyek B: Sistem Nilai dan Rapor Sederhana

Bangun aplikasi dengan entitas Siswa, Mata Pelajaran, dan Nilai (tabel pivot dengan kolom tambahan berupa nilai), menerapkan relasi Many-to-Many. Tambahkan fitur menampilkan rata-rata nilai per siswa menggunakan Eloquent.

## Proyek C: Sistem Reservasi Ruangan/Lapangan Sekolah

Bangun aplikasi dengan entitas Ruangan dan Reservasi, menerapkan validasi agar tidak terjadi bentrok jadwal (dua reservasi pada ruangan dan waktu yang sama), sebuah tantangan validasi yang lebih kompleks dibanding contoh pada Bab 8.

Ketiga proyek di atas dapat dikembangkan lebih lanjut dengan menambahkan autentikasi berjenjang (admin dapat mengelola data master, pengguna biasa hanya dapat mengajukan peminjaman/reservasi), sebuah pengantar menuju role-based access control yang akan dipelajari secara mendalam pada buku Kelas XII.

# Lampiran: Troubleshooting Masalah Umum Laravel

Lampiran ini merangkum masalah-masalah umum yang sering ditemui pemula saat belajar Laravel beserta cara mengatasinya, sebagai referensi cepat saat praktik.

| Gejala | Kemungkinan Penyebab | Solusi |
|---|---|---|
| Halaman menampilkan 'SQLSTATE... Unknown database' | Database belum dibuat atau nama pada .env tidak sesuai | Buat database melalui phpMyAdmin, periksa kembali DB_DATABASE pada .env |
| Error 'Class not found' setelah membuat file baru | Autoload Composer belum diperbarui | Jalankan composer dump-autoload |
| Perubahan pada .env tidak terlihat | Laravel meng-cache konfigurasi | Jalankan php artisan config:clear |
| Error 'Target class does not exist' pada routes | Lupa menuliskan use Controller di bagian atas routes/web.php | Tambahkan statement use App\Http\Controllers\NamaController; |
| Halaman blank putih tanpa pesan error | APP_DEBUG bernilai false | Ubah APP_DEBUG=true pada .env selama masa pengembangan (bukan produksi) |
| Error 'CSRF token mismatch' saat submit form | Lupa menambahkan @csrf pada form | Tambahkan @csrf tepat setelah tag <form> |
| Data relasi menampilkan error 'Trying to get property of non-object' | Data relasi belum di-load atau relasi belum diatur di Model | Periksa method relasi pada Model dan gunakan eager loading with() |

## Lampiran: Checklist Proyek Akhir Semester

Gunakan checklist berikut sebagai panduan memastikan proyek akhir semester (Bab 14) telah memenuhi seluruh aspek yang diharapkan sebelum dipresentasikan.

| Aspek | Kriteria | Sudah? |
|---|---|---|
| Struktur Data | Minimal dua Model dengan migration rapi dan relasi yang jelas | ☐ |
| CRUD | Seluruh operasi Create, Read, Update, Delete berjalan tanpa error | ☐ |
| Validasi | Form tambah dan edit memiliki validasi yang memadai | ☐ |
| Autentikasi | Breeze terpasang, rute CRUD dilindungi middleware auth | ☐ |
| Relasi Data | Minimal satu relasi Eloquent digunakan secara nyata di aplikasi | ☐ |
| Tampilan | Menggunakan Bootstrap, rapi, dan responsif | ☐ |
| Git & GitHub | Riwayat commit bertahap, .env tidak terunggah, README informatif | ☐ |
| Dokumentasi AI | Bagian yang dibantu AI dicantumkan beserta pemahaman pribadi | ☐ |

# Lampiran: Peta Jalan Menuju Pemrograman Web Lanjutan Kelas XII

Lampiran ini memberikan gambaran bagi peserta didik mengenai kompetensi apa saja yang akan dipelajari lebih lanjut pada buku Pemrograman Web Lanjutan Kelas XII, sekaligus menunjukkan bagaimana fondasi yang telah dibangun sepanjang buku ini akan terus digunakan dan diperluas.

| Kompetensi Kelas XI (Buku Ini) | Pengembangan pada Kelas XII |
|---|---|
| CRUD dengan Blade (tampilan direnderi di server) | REST API — data dikirim dalam format JSON, dapat dikonsumsi aplikasi lain (termasuk aplikasi mobile) |
| Autentikasi dasar dengan Breeze | Middleware role-based access control — membedakan hak akses admin dan pengguna biasa |
| Validasi dasar dengan Form Request | Keamanan aplikasi lanjutan — audit XSS, CSRF, mass assignment secara menyeluruh |
| Proyek dijalankan secara lokal (php artisan serve) | Deployment ke Ubuntu Server sungguhan, lanjutan langsung dari Bab 14 buku Kelas X |
| Vibecoding untuk boilerplate dan debugging | Vibecoding untuk testing otomatis, security review, dan dokumentasi API |
| Portofolio GitHub sederhana | Portofolio profesional lengkap dengan dokumentasi, siap digunakan melamar kerja/magang |

Peserta didik yang telah menuntaskan proyek akhir semester pada Bab 14 buku ini sangat disarankan mempertahankan dan terus mengembangkan proyek tersebut — bukan memulai proyek baru dari nol — ketika memasuki buku Kelas XII, sehingga peningkatan kompetensi dapat dirasakan secara nyata pada aplikasi yang sama, layaknya bagaimana aplikasi sungguhan terus dikembangkan secara bertahap di dunia kerja.

# Lampiran: Contoh Template README Proyek

Lampiran ini menyediakan contoh struktur README.md yang dapat digunakan sebagai templat dokumentasi proyek akhir semester (Bab 14) saat diunggah ke GitHub, mengikuti praktik dokumentasi yang lazim ditemui pada proyek open-source maupun portofolio profesional.

| Bagian README | Isi yang Dicantumkan |
|---|---|
| Judul dan Deskripsi Singkat | Nama aplikasi dan satu-dua kalimat penjelasan fungsinya |
| Fitur Utama | Daftar poin fitur yang telah diimplementasikan (CRUD, autentikasi, dsb.) |
| Teknologi yang Digunakan | Laravel, MySQL, Bootstrap, beserta versinya bila relevan |
| Cara Instalasi | Langkah-langkah git clone, composer install, konfigurasi .env, migrate, hingga menjalankan php artisan serve |
| Struktur Database | Ringkasan tabel utama dan relasinya, dapat disertai diagram sederhana |
| Penggunaan AI (Vibecoding) | Bagian mana yang dibantu AI beserta penjelasan pemahaman pribadi, sesuai Bab 13 |
| Kontributor | Nama anggota kelompok (jika dikerjakan berkelompok) |

README yang baik menjadi kesan pertama bagi siapa pun yang melihat repositori proyek — baik guru yang menilai, maupun calon pemberi kerja yang meninjau portofolio GitHub di masa depan — sehingga kebiasaan menulis dokumentasi yang jelas sejak proyek sekolah menjadi investasi berharga bagi kesiapan kerja peserta didik.

# Daftar Pustaka

Laravel Documentation. The PHP Framework for Web Artisans — https://laravel.com/docs.

Otwell, Taylor & Laravel Community. Laravel Bootcamp — panduan resmi membangun aplikasi Laravel.

PHP Documentation Group. Manual Resmi PHP — Object-Oriented Programming.

Git Documentation. Pro Git Book — https://git-scm.com/book.

Bootstrap Documentation. Bootstrap 5 — Component Library.

Kemdikbudristek. Capaian Pembelajaran Mata Pelajaran Pemrograman Web Lanjutan, Kurikulum Merdeka SMK Program Keahlian TJKT.