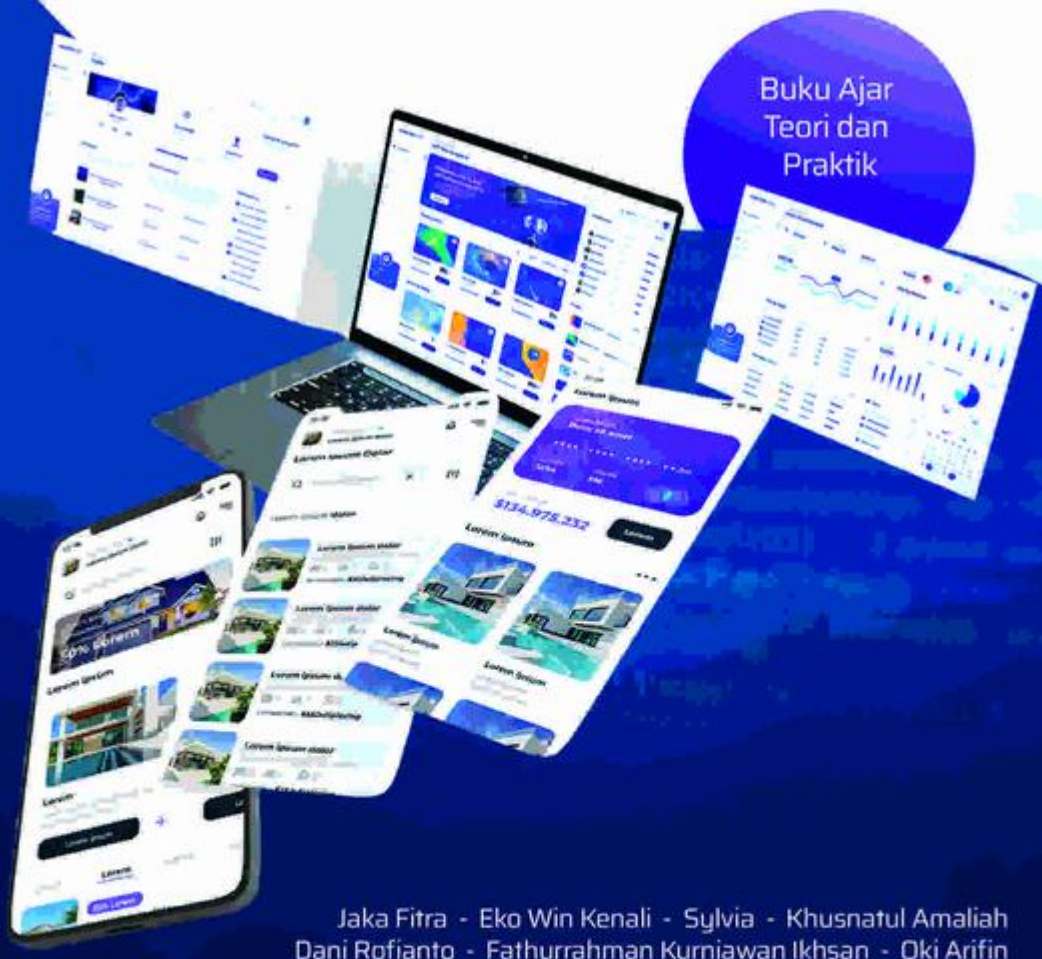
litnus

• PERANCANGAN •

PERANGKAT LUNAK

Buku Ajar
Teori dan
Praktik



Jaka Fitra - Eko Win Kenali - Sylvia - Khusnatul Amaliah
Dani Rofianto - Fathurrahman Kurniawan Ikhsan - Oki Arifin

PERANCANGAN PERANGKAT LUNAK

Buku Ajar
Teori dan
Praktik

Jaka Fitra - Eko Win Kenali - Sylvia
Khusnatul Amaliah - Dani Rofianto
Fathurrahman Kurniawan Ikhsan - Oki Arifin

litnus

PERANCANGAN PERANGKAT LUNAK

Buku ajar

Ditulis oleh:

JAKA FITRA, EKO WIN KENALI, SYLVIA, KHUSNATUL AMALIAH, DANI ROFIANTO, FATHURRAHMAN KURNIAWAN IKHSAN, OKI ARIFIN.

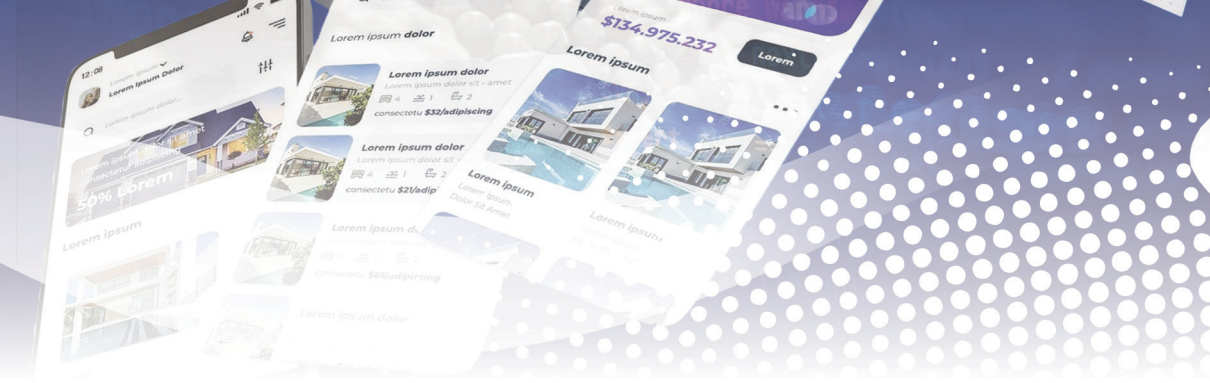
Diterbitkan, dicetak, dan didistribusikan oleh
PT. Literasi Nusantara Abadi Grup
Perumahan Puncak Joyo Agung Residence Kav. B11 Merjosari
Kecamatan Lowokwaru Kota Malang 65144
Telp : +6285887254603, +6285841411519
Email: literasinusantaraofficial@gmail.com
Web: www.penerbitlitnus.co.id
Anggota IKAPI No. 340/JTI/2022

Cetakan I, Desember 2025

Perancang sampul: Rosyiful Aqli
Penata letak: Muhammad Ridho Naufal

ISBN : 978-634-234-871-0

xii + 200 hlm. ; 15,5x23 cm.



Prakata

Di era digital yang bergerak dengan kecepatan luar biasa, perangkat lunak telah menjadi tulang punggung hampir setiap aspek kehidupan kita. Dari aplikasi *mobile* yang kita gunakan setiap hari, sistem manajemen *enterprise* yang kompleks, hingga kecerdasan buatan yang membentuk masa depan, kualitas dan efisiensi sebuah sistem perangkat lunak sangat bergantung pada fondasi yang kokoh: **perancangan**.

Buku ini hadir sebagai panduan komprehensif yang dirancang untuk membekali Anda dengan pemahaman mendalam tentang prinsip, metodologi, dan praktik terbaik dalam perancangan perangkat lunak. Kami percaya bahwa perancangan yang matang bukan hanya tentang menggambar diagram atau menulis spesifikasi teknis; ini adalah seni dan ilmu untuk menerjemahkan kebutuhan bisnis dan pengguna menjadi solusi teknis yang elegan, fungsional, dan berkelanjutan.

Dalam perjalanan ini, kita akan menjelajahi berbagai konsep penting:

- Kita akan memulai dengan **Prinsip Dasar Perancangan UI/UX**, memahami bagaimana menciptakan antarmuka yang intuitif dan mudah digunakan, serta pentingnya aksesibilitas dan desain berpusat pada pengguna.
- Selanjutnya, kita akan menyelami **Metodologi dan Notasi Pemodelan Perangkat Lunak**, dengan fokus pada Bahasa Pemodelan Terpadu

(UML) sebagai alat vital untuk memvisualisasikan struktur dan perilaku sistem.

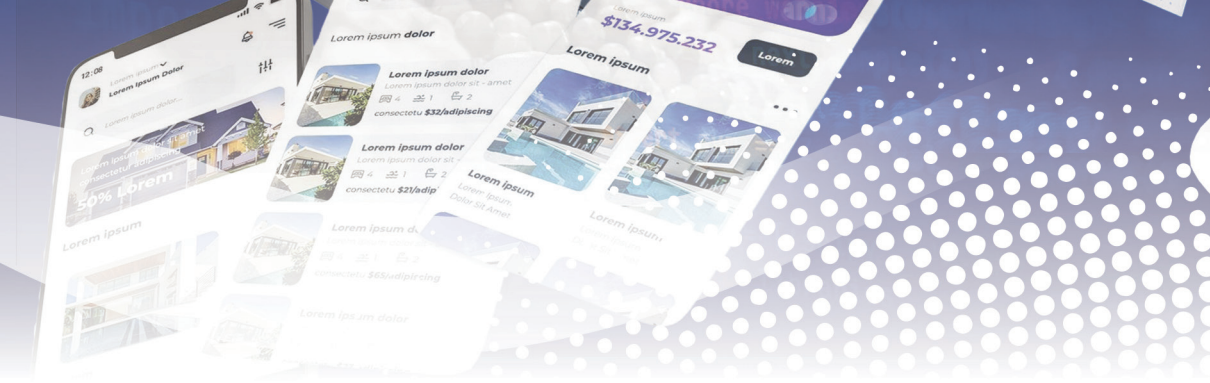
- Kita juga akan memahami **Perancangan Antarmuka Pengguna (UI)** yang detail, termasuk perbedaan esensial antara *wireframe*, *mockup*, dan *prototyping*.
- Dari sana, kita akan mendaki ke tingkat yang lebih tinggi yaitu **Perancangan Arsitektur Perangkat Lunak**, belajar bagaimana membangun fondasi sistem yang kuat, skalabel, dan tangguh.
- Pentingnya **Dokumentasi Perancangan Perangkat Lunak** juga akan dibahas, memastikan bahwa keputusan desain Anda terekam dengan jelas dan dapat dikomunikasikan secara efektif ke seluruh tim dan *stakeholder*.
- Tak lupa, kita akan membahas **Analisis Kompetitor**, sebuah bab krusial yang akan membimbing Anda dalam memahami lanskap pasar dan menemukan peluang diferensiasi produk Anda.
- Terakhir, kita akan mengulas berbagai **Metode Pengembangan Perangkat Lunak**, dari model tradisional seperti Waterfall hingga pendekatan adaptif seperti Agile (Scrum, Kanban, Extreme Programming), yang akan membentuk cara Anda dan tim Anda bekerja.

Setiap bab dirancang untuk tidak hanya memberikan teori, tetapi juga wawasan praktis, studi kasus singkat, dan tugas yang mendorong Anda untuk mengaplikasikan konsep-konsep yang dipelajari. Kami berharap, setelah menyelesaikan buku ini, Anda tidak hanya memiliki pengetahuan yang kuat tentang perancangan perangkat lunak, tetapi juga kepercayaan diri untuk menghadapi tantangan pengembangan sistem yang kompleks di dunia nyata.

Selamat belajar dan berinovasi!

Bandar Lampung, Agustus 2025

Penulis



Daftar Isi

BAB 3

Metodologi Perancangan Perangkat Lunak—45

BAB 4

NOTASI PERANCANGAN: PENGANTAR UML (UNIFIED MODELING LANGUAGE)—65

BAB 5

Perancangan Antarmuka Pengguna (User Interface Design)—99

BAB 6

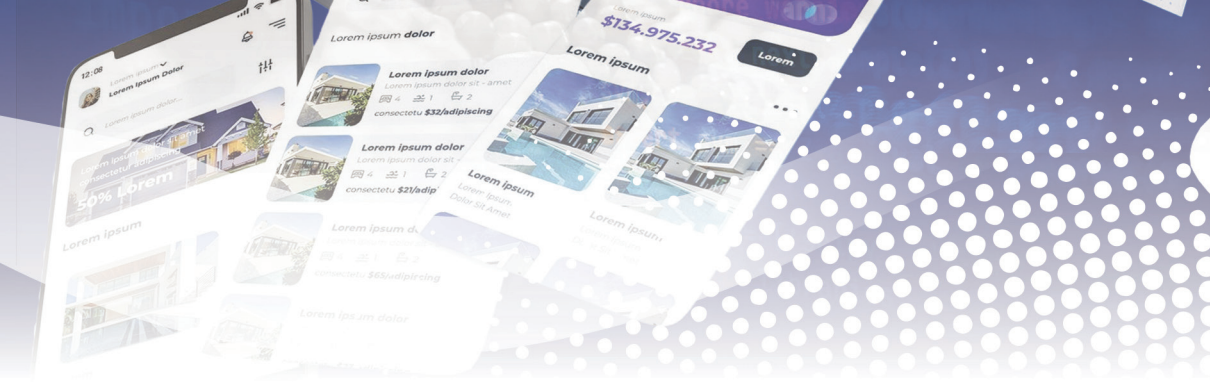
Dokumentasi Perancangan Perangkat Lunak—127

BAB 7

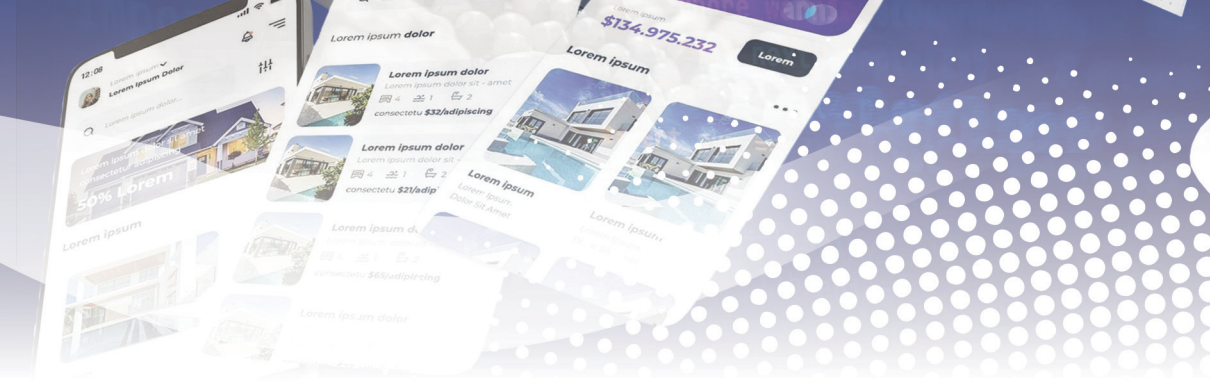
Analisis Kompetitor—145

BAB 8

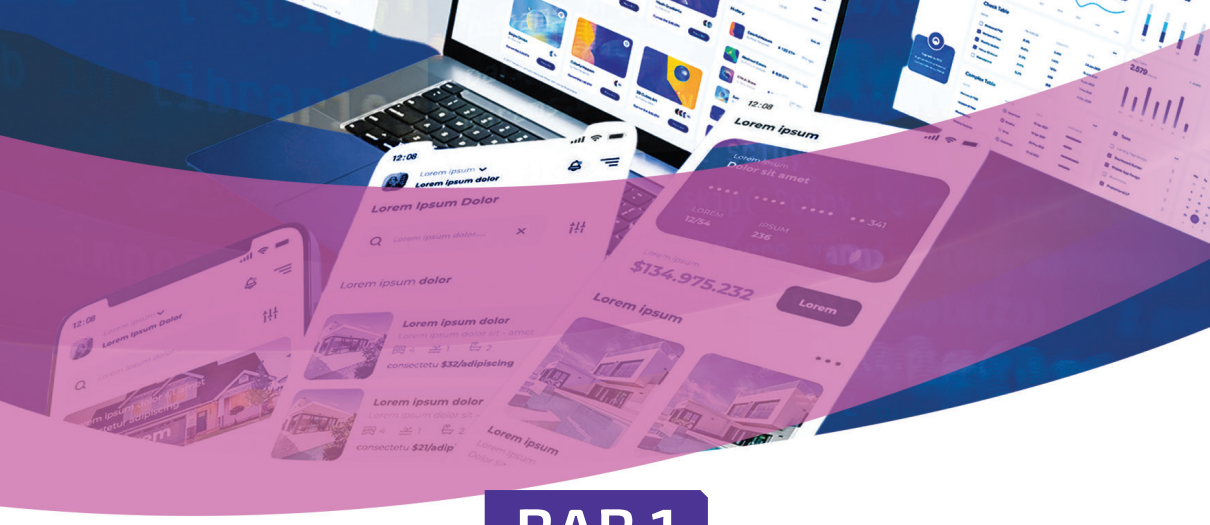
Metode Pengembangan Perangkat Lunak—167



Daftar Gambar



Daftar Tabel



BAB 1

Pengantar Perancangan Perangkat Lunak

1.1 Pengertian dan Ruang Lingkup Perancangan Perangkat Lunak

1.1.1. Definisi Perancangan Perangkat Lunak

Perancangan perangkat lunak merupakan proses mendefinisikan arsitektur, modul, antarmuka, dan data untuk sebuah sistem untuk memenuhi persyaratan yang ditentukan**.** Bayangkan membangun sebuah rumah. Sebelum tukang mulai memaku papan dan menyusun batu bata, ada tahap di mana arsitek dan insinyur sipil membuat gambar denah, spesifikasi bahan, dan rencana pondasi. Begitulah peran perancangan perangkat lunak. Ini bukan tentang langsung menulis kode, melainkan tentang membuat "cetak biru" yang terperinci dan logis sebelum proses pengkodean dimulai

Perancangan perangkat lunak (*software design*) adalah tahap penting dalam proses pengembangan perangkat lunak yang bertujuan untuk mendefinisikan struktur, komponen, antarmuka, dan data dari sistem yang akan dibangun. Tahap ini menjadi jembatan antara analisis kebutuhan dan implementasi sistem.

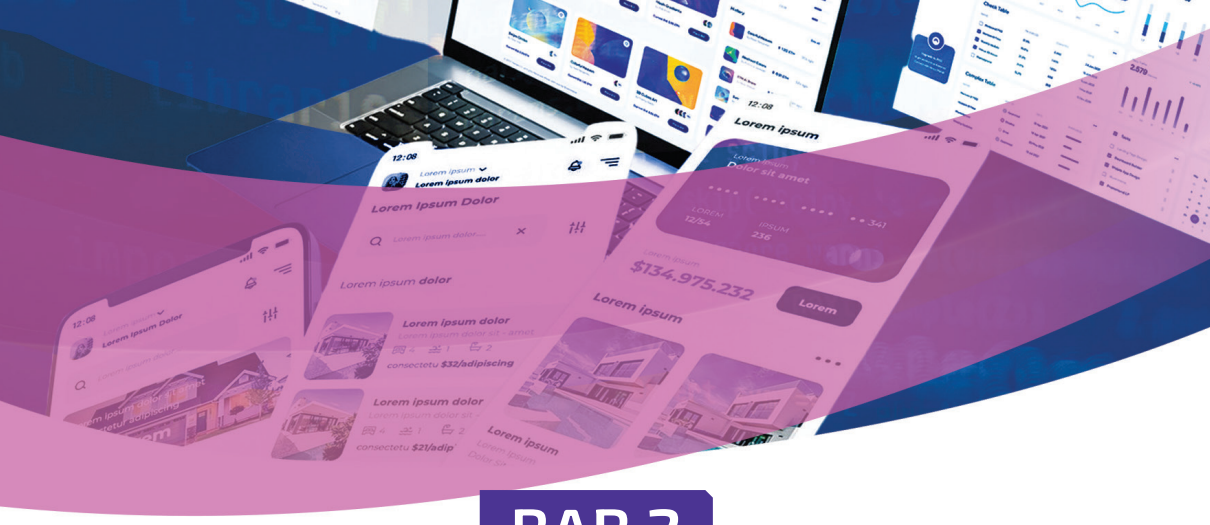
Mari kita analogikan dengan lebih sederhana. Ketika Anda ingin membuat kue, pertama Anda menentukan resep (kebutuhan: ingin kue

cokelat yang manis dan lembut). Setelah itu, Anda tidak langsung mencampur semua bahan. Anda akan memikirkan "bagaimana" cara membuatnya: urutan pencampuran bahan, suhu oven, dan alat apa yang dibutuhkan. Proses "memikirkan bagaimana" ini adalah perancangan. Dalam pengembangan perangkat lunak, tahap ini melibatkan pemikiran strategis untuk menyelesaikan masalah yang kompleks. Ini termasuk memecah sistem besar menjadi komponen-komponen yang lebih kecil dan mudah dikelola, menentukan hubungan antar komponen, dan merancang interaksi pengguna.

1.1.2. 1.1.2. Tujuan dan Pentingnya Perancangan

- Memastikan Kesesuaian Kebutuhan: Perancangan membantu memastikan bahwa perangkat lunak yang dibangun akan memenuhi semua kebutuhan fungsional dan non-fungsional yang telah diidentifikasi. Tanpa perancangan yang cermat, ada risiko besar bahwa produk akhir tidak sesuai dengan harapan pengguna atau tidak menyelesaikan masalah yang sebenarnya. Perancangan berfungsi sebagai validasi awal untuk memastikan semua persyaratan tercakup dengan benar.

  Contoh Praktis: Misalnya, Anda diminta membangun aplikasi kasir untuk UMKM. Kebutuhannya adalah bisa mencatat penjualan, menghitung total, dan mencetak struk. Tanpa perancangan, pengembang mungkin langsung membuat antarmuka dan kode. Tapi saat diuji, ternyata aplikasi tidak bisa mengelola diskon atau retur barang karena tidak dirancang untuk itu. Dengan perancangan awal, kita bisa membuat skema fitur-fitur ini dan memastikan desain mengakomodasi semua kebutuhan, bahkan yang belum terlihat jelas di awal. Ini seperti membuat daftar belanja yang lengkap sebelum pergi ke supermarket.
- Mengurangi Kompleksitas: Perangkat lunak modern seringkali sangat kompleks. Perancangan memungkinkan tim untuk memecah sistem besar menjadi bagian-bagian yang lebih kecil dan lebih mudah dikelola, yang disebut modul. Pendekatan modular ini membuat setiap pengembang dapat fokus pada satu bagian tertentu tanpa harus



BAB 2

Prinsip Dasar Perancangan Perangkat Lunak

Setelah memahami apa itu perancangan perangkat lunak dan posisinya dalam siklus pengembangan, kini saatnya kita menyelami "bagaimana" perancangan yang baik itu dibuat. Perancangan yang efektif tidak hanya terjadi begitu saja; ia dibangun di atas serangkaian prinsip dan pola yang telah terbukti. Prinsip-prinsip ini bertindak sebagai pedoman yang membantu kita menciptakan perangkat lunak yang tidak hanya berfungsi, tetapi juga mudah dipelihara, fleksibel, dan berkualitas tinggi.

2.1 2.1. Prinsip Desain Umum

Prinsip-prinsip desain umum adalah fondasi dari setiap perancangan perangkat lunak yang solid. Memahami dan menerapkan prinsip-prinsip ini akan membantu Anda membangun sistem yang kuat dan adaptif.

Prinsip Kohesi (Cohesion) dan Kopling (Coupling)

Dua konsep ini sering dibahas bersama karena mereka saling melengkapi dalam menentukan kualitas struktur modular perangkat lunak.

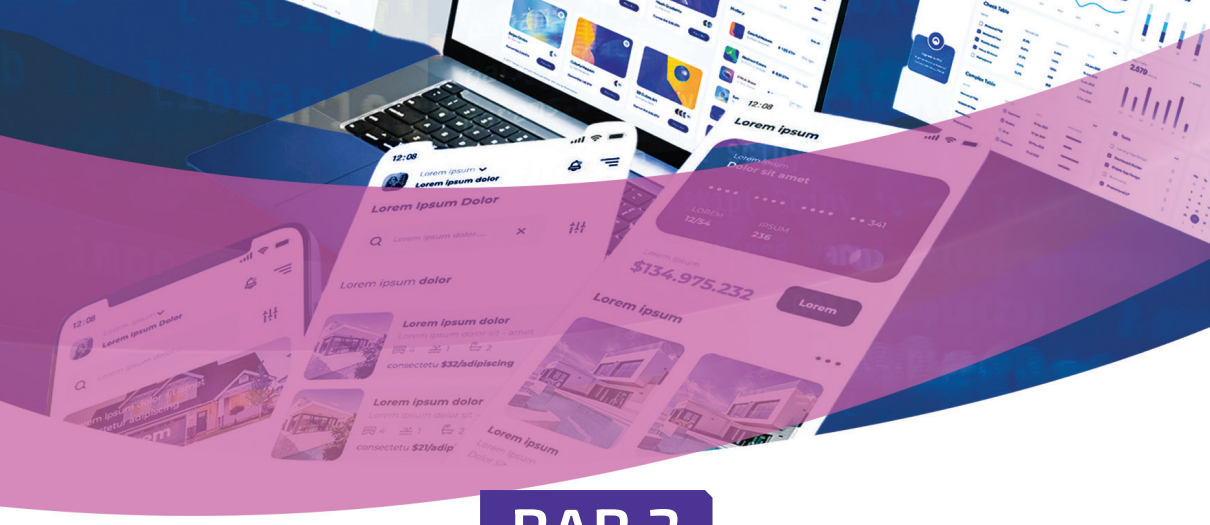
A. Kohesi (Cohesion)

Kohesi mengukur seberapa kuat elemen-elemen di dalam satu modul atau unit saling terkait dan berfokus pada satu tujuan. Modul dengan kohesi

tinggi berarti semua elemen (fungsi, data) di dalamnya bekerja sama secara erat untuk menyelesaikan satu tugas atau menyediakan satu layanan spesifik.

- **Tujuan Kohesi Tinggi:**
  - **Mudah Dipahami:** Karena modul hanya melakukan satu hal, lebih mudah untuk memahami fungsionalitasnya.
  - **Mudah Dipelihara:** Perubahan pada satu bagian modul tidak mungkin memengaruhi bagian lain yang tidak terkait.
  - **Mudah Digunakan Kembali:** Modul yang spesifik dapat lebih mudah digunakan di bagian lain dari sistem atau di proyek lain.
  - **Mengurangi Kompleksitas:** Mengisolasi fungsionalitas membantu mengurangi kebingungan dan interaksi yang tidak perlu antar bagian.
- **Jenis-jenis Kohesi (dari yang terburuk ke terbaik):**

1. Coincidental Cohesion: Elemen-elemen dalam modul tidak memiliki hubungan logis sama sekali. (Contoh: Modul yang berisi fungsi hitungTotalPesanan() dan kirimEmailPromo()). Hindari ini!
2. Logical Cohesion: Elemen-elemen dikelompokkan berdasarkan kategori logis, tetapi tidak harus beroperasi bersamaan. (Contoh: Modul Utilitas yang berisi semua fungsi yang "berguna" seperti validasiInput(), formatTanggal(), enkripsiData()).
3. **Temporal Cohesion:** Elemen-elemen dikelompokkan karena dieksekusi pada waktu yang sama. (Contoh: Fungsi inisialisasiAplikasi() yang melakukan *setup* database, *load* konfigurasi, dan *start* layanan).
4. **Procedural Cohesion:** Elemen-elemen dikelompokkan karena mengikuti urutan prosedur tertentu. (Contoh: Modul untuk Proses Checkout yang berisi langkah 1: validasiKeranjang, langkah 2: hitungPajak, langkah 3: prosesPembayaran).
5. **Communicational Cohesion:** Elemen-elemen beroperasi pada data yang sama. (Contoh: Modul ManajemenPengguna yang berisi fungsi tambahPengguna(), hapusPengguna(), editPengguna() karena semuanya beroperasi pada objek Pengguna).



BAB 3

Metodologi Perancangan Perangkat Lunak

Perancangan perangkat lunak bukanlah proses yang tunggal dan kaku. Sepanjang sejarah pengembangan perangkat lunak, berbagai pendekatan dan filosofi telah muncul untuk memandu tim dalam merancang sistem yang kompleks. Metodologi perancangan perangkat lunak adalah kerangka kerja sistematis yang mendefinisikan prinsip, teknik, dan praktik untuk mengubah kebutuhan (apa yang harus dilakukan sistem) menjadi solusi desain (bagaimana sistem akan melakukannya).

Memilih metodologi perancangan yang tepat sangat penting karena akan memengaruhi cara tim bekerja, kualitas desain yang dihasilkan, dan kemampuan sistem untuk beradaptasi di masa depan. Dalam bab ini, kita akan menjelajahi beberapa metodologi perancangan yang paling umum digunakan.

3.1 Metodologi Terstruktur (Structured Design)

Metodologi Terstruktur (Structured Design) adalah pendekatan perancangan perangkat lunak yang berfokus pada dekomposisi sistem menjadi modul-modul hierarkis dan terorganisir dengan baik. Pendekatan ini sangat populer di era awal pengembangan perangkat lunak, terutama dalam konteks model pengembangan linear seperti Model Air Terjun. Ia

menekankan pada desain *top-down* (dari gambaran besar ke detail) dan penggunaan diagram alir data (*Data Flow Diagrams*–DFD) serta struktur diagram.

Penjelasan dan Karakteristik

Perancangan Terstruktur menitikberatkan pada dua konsep utama:

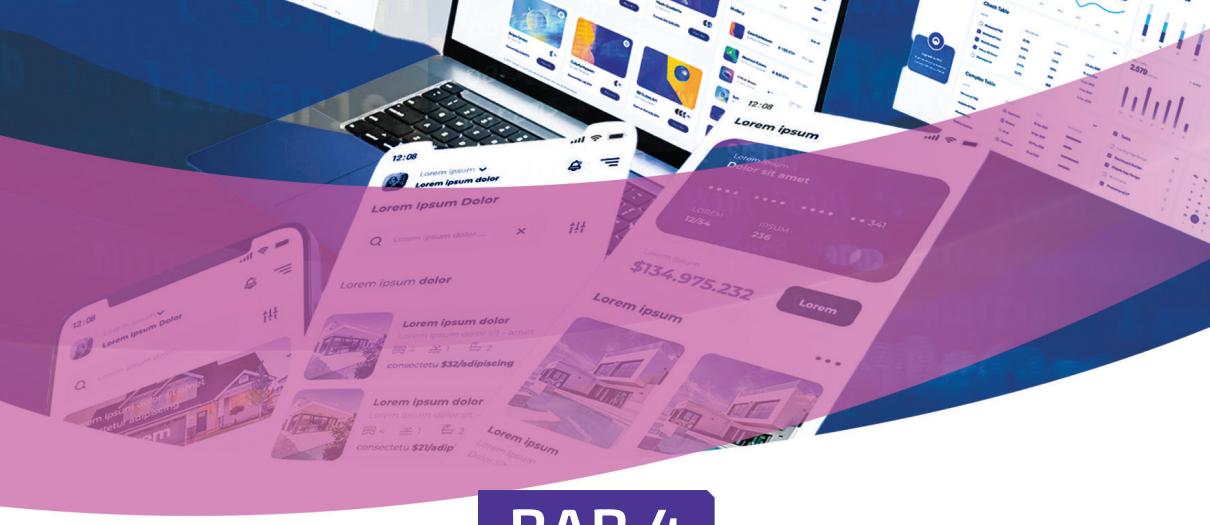
- **Dekomposisi Fungsional:** Sistem dipecah menjadi fungsi-fungsi atau proses-proses yang lebih kecil dan dapat dikelola. Setiap fungsi kemudian dipecah lagi menjadi sub-fungsi, dan seterusnya, hingga mencapai tingkat detail yang memadai.
- **Aliran Data:** Bagaimana data bergerak di antara fungsi-fungsi atau modul-modul ini sangat ditekankan. DFD adalah alat utama untuk memvisualisasikan aliran data dan transformasi yang terjadi.

Karakteristik Utama Metodologi Terstruktur:

- **Fokus pada Proses:** Lebih menonjolkan bagaimana data diproses dan ditransformasikan oleh fungsi-fungsi, daripada pada data itu sendiri.
- **Top-Down Design:** Dimulai dari gambaran umum sistem dan secara bertahap merinci ke bawah hingga modul-modul individual.
- **Moduliratas:** Mendorong pembagian sistem menjadi modul-modul yang kohesif tinggi dan kopling rendah.
- **Dokumentasi yang Kuat:** Menghasilkan banyak dokumentasi seperti DFD, kamus data (*data dictionary*), dan spesifikasi proses.
- **Cocok untuk Sistem Prosedural:** Sangat efektif untuk sistem yang berorientasi pada fungsi atau proses bisnis yang jelas dan berurutan.

Kelebihan Metodologi Terstruktur:

- **Sistematis dan Teratur:** Memberikan langkah-langkah yang jelas dan terdefinisi dalam proses perancangan.
- **Mudah Dipahami untuk Proyek Sederhana:** Alurnya yang linear dan fokus pada fungsi membuatnya mudah diikuti untuk sistem dengan persyaratan yang stabil.



BAB 4

NOTASI PERANCANGAN: PENGANTAR UML (UNIFIED MODELING LANGUAGE)

Perancangan perangkat lunak adalah tentang mengkomunikasikan ide. Sebagus apapun desain yang Anda miliki di kepala, jika Anda tidak bisa mengkomunikasikannya dengan jelas kepada tim pengembang, *stakeholder*, atau bahkan kepada diri sendiri di masa depan, maka desain itu kurang bermanfaat. Di sinilah Unified Modeling Language (UML) berperan.

UML adalah bahasa pemodelan visual standar yang paling banyak digunakan dalam rekayasa perangkat lunak. Ia menyediakan sekumpulan notasi grafis untuk menggambarkan, memvisualisasikan, membangun, dan mendokumentasikan artefak-artefak dari sebuah sistem perangkat lunak. Dengan UML, kita bisa membuat "cetak biru" yang dapat dimengerti oleh siapa pun yang memahami UML, tanpa harus bergantung pada deskripsi teks yang panjang dan mungkin ambigu.

4.1 Pengenalan UML dan Blok Bangunan UML

UML (Unified Modeling Language) adalah bahasa grafis standar, yang tidak bergantung pada bahasa pemrograman tertentu, untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan artefak dari sistem *software* yang kompleks. UML membantu memodelkan sistem, baik dari perspektif struktural maupun perilaku.

Bayangkan UML seperti "bahasa gambar teknik" untuk perangkat lunak. Sama seperti seorang arsitek yang menggunakan denah dan diagram struktural untuk merancang sebuah bangunan, seorang perancang perangkat lunak menggunakan berbagai diagram UML untuk memodelkan sistem perangkat lunak.

- **Mengapa UML Penting?**
  - **Standarisasi:** Menyediakan cara standar untuk memodelkan sistem, sehingga desainer dari berbagai tim atau organisasi dapat memahami desain satu sama lain.
  - **Visualisasi:** Memungkinkan visualisasi konsep abstrak menjadi representasi grafis yang lebih mudah dipahami.
  - **Komunikasi:** Memfasilitasi komunikasi yang efektif antara semua pihak yang terlibat dalam proyek (analis, desainer, pengembang, penguji, *stakeholder*).
  - **Mengurangi Ambiguitas:** Notasi yang jelas dan konsisten membantu mengurangi kesalahpahaman.
  - **Dokumentasi:** Menjadi bagian integral dari dokumentasi sistem.

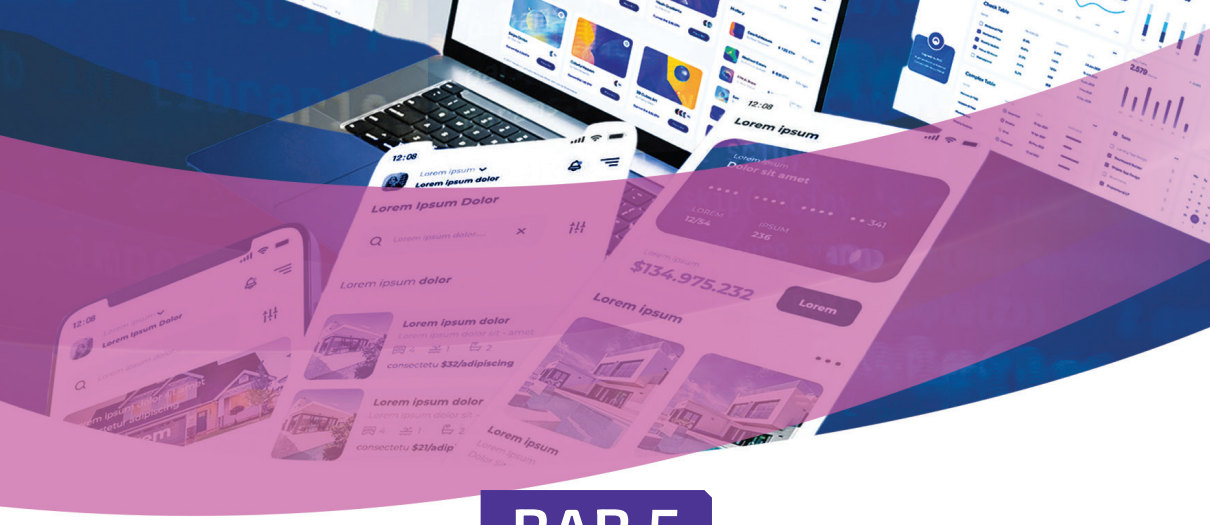
Elemen Dasar UML (Things, Relationships, Diagrams)

UML dibangun dari tiga blok bangunan dasar yang dapat digabungkan untuk membuat diagram yang kompleks:

A. Things (Hal/Entitas)

Things adalah elemen-elemen fundamental yang merepresentasikan bagian-bagian dari model. Ini adalah *noun* dalam bahasa pemodelan. UML memiliki empat jenis *things*:

1. Structural Things (Hal Struktural): Merepresentasikan elemen statis atau konseptual yang membentuk struktur sistem. Mereka adalah "bagian" dari sistem.
   - Contoh:
     - Kelas (Class): Blueprint untuk objek, mendefinisikan atribut dan operasi.



BAB 5

Perancangan Antarmuka Pengguna (User Interface Design)

Kita telah membahas fondasi teoritis dan metodologi perancangan perangkat lunak. Kini, saatnya kita melangkah ke bagian yang paling terlihat dan dirasakan langsung oleh pengguna: Antarmuka Pengguna (User Interface–UI). Perancangan UI adalah seni dan ilmu tentang menciptakan antarmuka yang tidak hanya indah secara visual, tetapi juga intuitif, efisien, dan menyenangkan untuk digunakan. Ini erat kaitannya dengan Pengalaman Pengguna (User Experience–UX), yang mencakup seluruh pengalaman seseorang saat berinteraksi dengan produk.

Perancangan antarmuka yang buruk dapat membuat perangkat lunak sehebat apapun secara fungsional menjadi tidak berguna bagi pengguna. Sebaliknya, UI/UX yang baik dapat membuat teknologi yang kompleks terasa sederhana dan mudah diakses.

5.1 Prinsip Dasar UI/UX

Perancangan UI/UX yang efektif didasarkan pada beberapa prinsip fundamental yang memandu desainer untuk menciptakan pengalaman yang optimal bagi pengguna. Memahami prinsip-prinsip ini adalah langkah pertama untuk membangun antarmuka yang sukses.

A. Konsep Usability (Kegunaan)

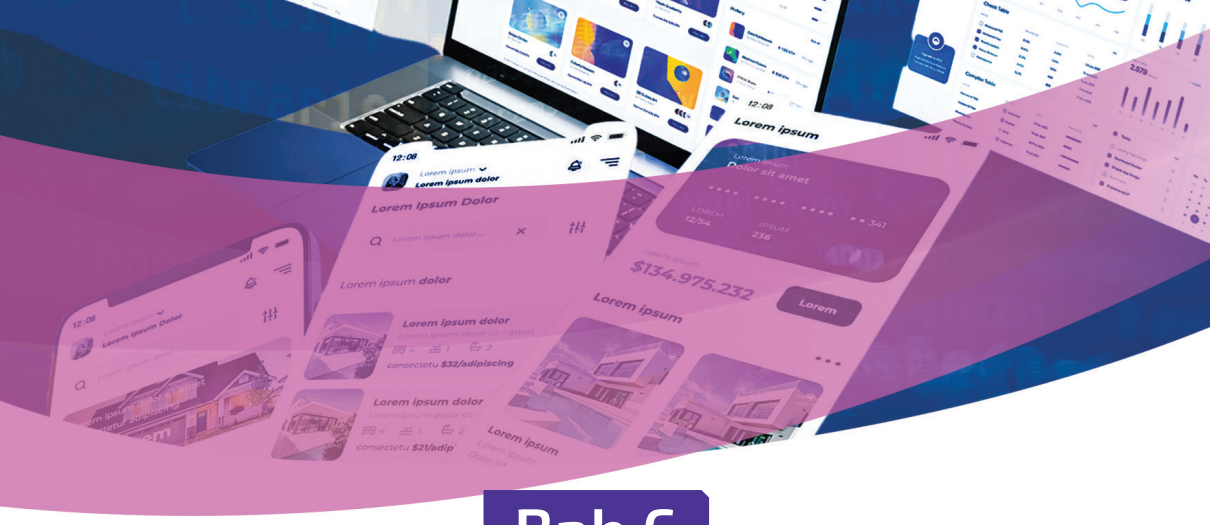
Usability (Kegunaan) mengacu pada sejauh mana suatu produk dapat digunakan oleh pengguna tertentu untuk mencapai tujuan tertentu dengan efektivitas, efisiensi, dan kepuasan dalam konteks penggunaan tertentu. Singkatnya, produk tersebut mudah digunakan.

- Efektivitas: Apakah pengguna bisa menyelesaikan tugas mereka dengan sukses?
  - › Contoh: Pengguna dapat menemukan tombol "Daftar" dan berhasil membuat akun.
- Efisiensi: Seberapa cepat dan dengan berapa banyak usaha pengguna bisa menyelesaikan tugas?
  - › Contoh: Proses pendaftaran hanya membutuhkan 3 langkah singkat, bukan 10 langkah yang membingungkan.
- Kepuasan: Apakah pengguna merasa senang dan nyaman saat menggunakan produk?
  - › Contoh: Antarmuka yang bersih, responsif, dan memberikan *feedback* visual yang menyenangkan saat interaksi.

Aspek Kunci Usability:

1. Kemudahan Dipelajari (Learnability): Seberapa mudah pengguna baru dapat mempelajari cara menggunakan sistem.
2. Efisiensi Penggunaan (Efficiency): Seberapa cepat pengguna dapat melakukan tugas setelah mereka mempelajarinya.
3. Kemudahan Mengingat (Memorability): Seberapa mudah pengguna dapat mengingat cara menggunakan sistem setelah beberapa waktu tidak menggunakannya.
4. Penanganan Kesalahan (Error Handling): Seberapa baik sistem membantu pengguna menghindari kesalahan dan bagaimana ia pulih dari kesalahan.
5. Kepuasan Pengguna (Satisfaction): Tingkat kesenangan pengguna saat menggunakan sistem.

- Contoh Praktis Usability: Tombol yang jelas dengan label yang mudah dimengerti, *form* pendaftaran yang tidak terlalu panjang, navigasi yang



Bab 6

Dokumentasi Perancangan Perangkat Lunak

Kita telah mempelajari berbagai metodologi dan notasi untuk perancangan perangkat lunak, mulai dari fondasi hingga arsitektur dan antarmuka pengguna. Namun, semua usaha perancangan ini akan menjadi kurang efektif jika tidak didokumentasikan dengan baik. Dokumentasi perancangan perangkat lunak adalah catatan tertulis atau visual yang menjelaskan keputusan desain, struktur, perilaku, dan semua aspek penting dari sebuah sistem perangkat lunak.

Dokumentasi yang efektif bukan hanya sebuah formalitas, tetapi merupakan aset yang sangat berharga dalam siklus hidup pengembangan perangkat lunak. Ia memastikan bahwa pengetahuan tentang sistem tidak hanya tersimpan di kepala para desainer atau pengembang, tetapi dapat diakses, dipahami, dan digunakan oleh seluruh tim, *stakeholder*, dan bahkan tim di masa depan.

6.1 Pengantar Dokumentasi Perancangan Perangkat Lunak

A. Definisi dan Pentingnya Dokumentasi Perancangan

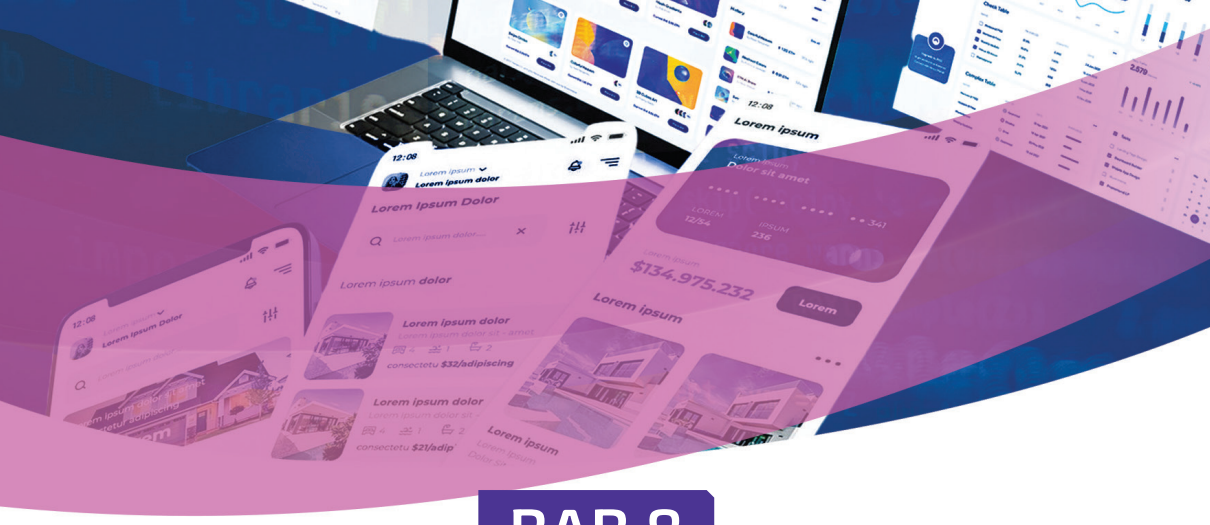
Dokumentasi perancangan perangkat lunak adalah kumpulan artefak (dokumen, diagram, model) yang menjelaskan desain internal dan eksternal

dari sebuah sistem perangkat lunak. Ini adalah cetak biru teknis yang digunakan oleh pengembang untuk membangun sistem, penguji untuk memverifikasi fungsionalitas, manajer untuk melacak kemajuan, dan *stakeholder* untuk memahami solusi yang diusulkan.

Dokumentasi ini mencakup spektrum yang luas, mulai dari diagram UML yang merinci kelas dan interaksi, hingga spesifikasi arsitektur yang menjelaskan bagaimana komponen-komponen besar berinteraksi, hingga panduan gaya UI yang memastikan konsistensi visual.

Mengapa Dokumentasi Perancangan Penting? Pentingnya dokumentasi tidak dapat diremehkan, karena ia mendukung berbagai aspek proyek:

1. Jembatan Komunikasi: Dokumentasi berfungsi sebagai media komunikasi standar antara semua anggota tim (desainer, pengembang, penguji), serta dengan *stakeholder* non-teknis (manajer proyek, klien, pengguna akhir). Ini memastikan semua pihak memiliki pemahaman yang sama tentang "apa" dan "bagaimana" sistem akan dibangun.
2. Basis Pengetahuan dan Memori Organisasi: Pengetahuan tentang sistem yang kompleks seringkali terdistribusi di antara individu-individu. Dokumentasi menangkap pengetahuan ini, mencegah hilangnya informasi penting ketika anggota tim keluar atau berpindah proyek. Ini menjadi "memori" kolektif bagi organisasi.
3. Panduan Implementasi: Bagi pengembang, dokumentasi perancangan adalah panduan utama tentang bagaimana membangun setiap bagian sistem. Ini mengurangi ambiguitas, mempercepat proses pengkodean, dan membantu menjaga konsistensi dalam implementasi.
4. Dasar untuk Pengujian: Penguji menggunakan dokumentasi desain untuk memahami fungsionalitas yang diharapkan dan perilaku sistem, sehingga mereka dapat merancang *test case* yang akurat dan komprehensif.
5. Mempermudah Pemeliharaan dan Evolusi: Seiring waktu, perangkat lunak perlu diperbaiki (*bug fixes*), ditingkatkan (*enhancements*), dan diadaptasi terhadap perubahan kebutuhan. Dokumentasi yang baik sangat krusial untuk kegiatan pemeliharaan ini, memungkinkan

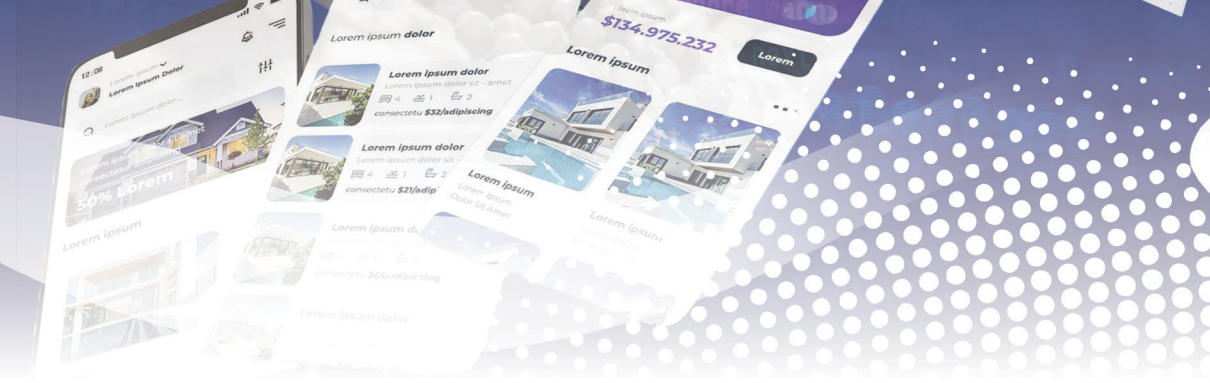


BAB 8

Metode Pengembangan Perangkat Lunak

Setelah melalui tahap perancangan dan analisis yang mendalam, langkah selanjutnya dalam siklus hidup pengembangan perangkat lunak adalah menentukan bagaimana semua pekerjaan tersebut akan diorganisir dan dilaksanakan. Di sinilah **metode pengembangan perangkat lunak** berperan. Metode pengembangan perangkat lunak, atau sering disebut juga metodologi atau *framework*, adalah kerangka kerja yang terstruktur dan sistematis yang memandu tim dalam proses perencanaan, desain, implementasi, pengujian, *deployment*, dan pemeliharaan produk perangkat lunak.

Memilih metode yang tepat adalah keputusan strategis yang dapat sangat memengaruhi keberhasilan proyek. Tidak ada satu pun metode yang "terbaik" untuk semua skenario; pilihan terbaik bergantung pada berbagai faktor seperti ukuran proyek, kompleksitas, stabilitas persyaratan, dan budaya tim. Bab ini akan menguraikan beberapa metode pengembangan perangkat lunak yang paling umum, beserta kelebihan dan kekurangannya, untuk membantu Anda memahami kapan dan mengapa satu metode mungkin lebih cocok daripada yang lain.



Lampiran

Contoh : TEMPLATE DOKUMEN PERANCANGAN SISTEM

1. Halaman Judul

- Judul Mini Project
- Nama Mahasiswa
- NPM
- Program Studi / Mata Kuliah
- Dosen Pengampu
- Tanggal

2. Daftar Isi

(Sesuaikan setelah isi dokumen dibuat)

3. Latar Belakang

Jelaskan alasan pemilihan topik dan permasalahan yang ingin diselesaikan.

Contoh:

Dalam aktivitas perkuliahan, mahasiswa sering kali kesulitan mengelola berbagai tugas dari banyak mata kuliah. Oleh karena itu, dibutuhkan aplikasi yang dapat membantu manajemen tugas secara efektif.

4. Tujuan

Uraikan tujuan dari pengembangan sistem ini.
Contoh:
Tujuan dari mini project ini adalah untuk membangun sistem manajemen tugas yang dapat digunakan oleh mahasiswa dalam mencatat, memonitor, dan menerima pengingat tugas-tugas mereka.

5. Deskripsi Sistem

Berikan gambaran umum tentang sistem yang akan dibangun.
Contoh:
Sistem ini memiliki dua jenis pengguna: Mahasiswa dan Admin. Mahasiswa dapat menambahkan tugas, mengatur tenggat, dan melihat daftar tugas. Admin dapat mengelola data pengguna dan melakukan pemantauan.

6. Diagram Use Case

- Sertakan diagram dan penjelasan masing-masing use case.

Contoh Aktor:

- Mahasiswa: login, tambah tugas, ubah tugas, hapus tugas
- Sistem: mengirim pengingat

7. Diagram Class

- Sertakan class diagram
- Penjelasan atribut dan relasi antar class

Contoh Class:

- Mahasiswa: id_mahasiswa, nama, email, password
- Tugas: id_tugas, judul, deskripsi, deadline, status
- Notifikasi: id_notifikasi, isi, waktu_kirim, status

8. Wireframe

- Gambar atau tangkapan layar struktur UI awal (bisa berupa sketsa tangan atau tools desain)
- Penjelasan singkat tiap tampilan

Contoh tampilan:

- Login Page
- Dashboard Page
- Task Form Page

9. Mockup UI

- Gambar mockup final sistem
- Sertakan keterangan elemen penting: tombol, field, layout, navigasi

10. Prototype UI

- Gambar prototype final sistem
- Sertakan keterangan elemen penting: tombol, field, layout, navigasi

11. Alur Navigasi

- Penjelasan alur pengguna dari halaman satu ke halaman lain
- Bisa menggunakan flowchart atau diagram alur

12. Analisis Aplikasi Kompetitor dan Evaluasi Awal

- Masukan dari pengguna awal atau rekan
- Rencana perbaikan atau peningkatan fitur
- Aplikasi Kompetitor (analisis, design, kelebihan, dan kekurangan)

13. Metode Pengembangan Perangkat Lunak Yang digunakan

14. Kesimpulan dan Rencana Pengembangan

- Ringkasan dari dokumen ini
- Rencana untuk implementasi atau pengembangan lanjutan

15. Referensi

Mini Project: Rancang Sistem Anda Sendiri

Ini merupakan tahap akhir dari proses pembelajaran di mana mahasiswa

diberikan kesempatan untuk menerapkan seluruh pengetahuan yang telah dipelajari dari bab sebelumnya ke dalam sebuah mini project. Mahasiswa akan merancang sistem informasi sederhana berdasarkan topik pilihan mereka, lengkap dengan dokumen perancangan, diagram, dan antarmuka pengguna.

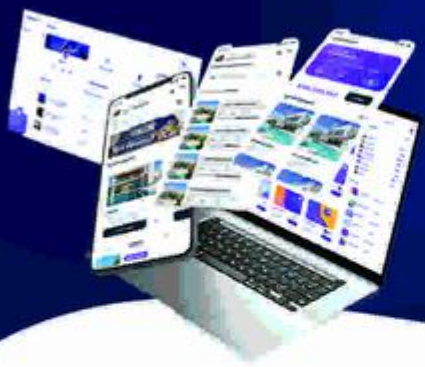
Penentuan Topik

Langkah pertama dalam mini project ini adalah menentukan topik atau permasalahan yang ingin diselesaikan dengan sistem informasi. Topik dapat berasal dari kebutuhan nyata di lingkungan sekitar, kampus, UMKM, organisasi, atau berdasarkan ide inovatif mahasiswa.
Contoh topik:

- Sistem Manajemen Tugas Mahasiswa
- Aplikasi Absensi Online
- Sistem Reservasi Kelas
- Sistem Inventaris Barang Laboratorium
- Sistem Monitoring Tugas Akhir

Kriteria topik:

- Relevan dengan kebutuhan nyata
- Memungkinkan untuk diimplementasikan dalam skala kecil
- Memiliki alur proses yang jelas dan dapat didokumentasikan



PERANCANGAN PERANGKAT LUNAK

Buku Ajar Teori dan Praktik

Di era digital yang bergerak dengan kecepatan luar biasa, perangkat lunak telah menjadi tulang punggung hampir setiap aspek kehidupan kita. Dari aplikasi mobile yang kita gunakan setiap hari, sistem manajemen enterprise yang kompleks, hingga kecerdasan buatan yang membentuk masa depan, kualitas dan efisiensi sebuah sistem perangkat lunak sangat bergantung pada fondasi yang kokoh: perancangan.

Buku ini hadir sebagai panduan komprehensif yang dirancang untuk membekali Anda dengan pemahaman mendalam tentang prinsip, metodologi, dan praktik terbaik dalam perancangan perangkat lunak. Kami percaya bahwa perancangan yang matang bukan hanya tentang menggambar diagram atau menulis spesifikasi teknis; ini adalah seni dan ilmu untuk menerjemahkan kebutuhan bisnis dan pengguna menjadi solusi teknis yang elegan, fungsional, dan berkelanjutan.

Setiap bab dirancang untuk tidak hanya memberikan teori, tetapi juga wawasan praktis, studi kasus singkat, dan tugas yang mendorong Anda untuk mengaplikasikan konsep-konsep yang dipelajari. Kami berharap, setelah menyelesaikan buku ini, Anda tidak hanya memiliki pengetahuan yang kuat tentang perancangan perangkat lunak, tetapi juga kepercayaan diri untuk menghadapi tantangan pengembangan sistem yang kompleks di dunia nyata.

litnus

literasinusantaraofficial@gmail.com
www.penerbitlitnus.co.id
Literasi Nusantara
literasinusantara_
085755971589

Teknik +17

ISBN 978-634-234-871-0

9 786342 348710