BAB II
TINJAUAN PUSTAKA

2.1 Teori Umum

2.1.1 Pengertian Rancang Bangun

Istilah tersebut merupakan konsep yang menggabungkan dua kata yaitu "rancang" dan "bangun". "Rancang" berdasarkan dari diksi atau kata kerja "perancangan", yang memiliki makna pada beberapa serangkaian proses untuk mengimplementasikan sebuah hasil dari analisis sistem ke dalam sekumpulan kode program. Konsep ini menjelaskan secara detail tentang kumpulan komponen sistem akan diimplementasikan. Dari dasar kata rancang bangun tersebut dapat digabungkan menjadi satu konsep mengenai tahapan atau suatu proses untuk mengubah hasil dari proses analisis sistem menjadi bentuk sebuah perangkat lunak atau aplikasi, baik dengan mengimplementasikan sistem baru atau mengembangkan dan memperbaiki sistem yang sudah ada. (Gunawan, Yusuf, & Nopitasari, 2021).

Proses ini melibatkan langkah-langkah seperti merancang arsitektur sistem, merinci komponen-komponen yang akan digunakan, menentukan teknologi yang tepat, dan mengembangkan kode program berdasarkan spesifikasi persyaratan sistem. Dengan demikian, rancang bangun merupakan tahapan penting dalam siklus pengembangan perangkat lunak yang memungkinkan transformasi dari konsep menjadi solusi yang dapat digunakan secara praktis.

2.1.2 Pengertian Aplikasi

Pengertian aplikasi dapat dijelaskan bahwa aplikasi merupakan sekumpulan kode program yang telah disusun atau perangkat lunak yang dibangun untuk melaksanakan tujuan tertentu atau menyediakan fitur khusus kepada pemakai (Gunawan, Yusuf, & Nopitasari, 2021). Aplikasi bisa beragam bentuknya, mulai dari yang sederhana seperti kalkulator atau kalender hingga yang kompleks seperti aplikasi perbankan *online* atau permainan video.

Secara lebih terperinci, pengertian aplikasi dapat dijelaskan sebagai berikut (Prehanto, S.Kom., M.Kom, 2020).

1. *Software* atau Perangkat Lunak yang digunakan dan dirancang untuk Tujuan yang telah ditentukan

   Aplikasi merupakan *software* yan dikembangkan untuk tujuan yang telah ditetapkan yaitu sebagai pengelola data, bermain *game*, berkomunikasi, mengedit foto, atau menjalankan tugas-tugas produktif lainnya.

2. Penyelesaian Tugas Tertentu

   Aplikasi biasanya dikembangkan untuk menyelesaikan tugas-tugas spesifik atau untuk memberikan fungsi tertentu kepada pengguna. Sebagai contoh, aplikasi yang dapat memudahkan pengguna untuk membuat dan mengelola dokumen teks, sedangkan aplikasi media sosial memfasilitasi interaksi dan berbagi informasi antar pengguna.

3. Berjalan pada Platform Tertentu

   Aplikasi dikembangkan untuk berjalan pada platform atau sistem operasi tertentu, seperti Windows, macOS, Android, iOS, dan sebagainya. Dengan demikian, aplikasi yang dirancang untuk satu platform tidak selalu dapat dijalankan pada platform lainnya tanpa modifikasi.

4. *User Interface* (Antarmuka Pengguna)

   Sebagian besar aplikasi memiliki *User Interface* (UI) yang pengguna dapat gunakan untuk berinteraksi dengan aplikasi tersebut. UI ini bisa berupa tampilan grafis dengan kumpulan tombol, list menu, dan komponen-komponen lainnya, atau dapat berupa antarmuka teks yang memanfaatkan perintah-perintah tertentu.

5. Menggunakan Fungsionalitas Perangkat

   Aplikasi dapat menggunakan berbagai fungsi dan kemampuan perangkat keras seperti kamera, mikrofon, sensor sidik jari, dan lain-lain. Hal tersebut memungkinkan aplikasi untuk menawarkan beragam fitur dan pengalaman kepada pengguna.

Dengan demikian, aplikasi adalah alat perangkat lunak yang dirancang untuk memberikan solusi atau memberikan nilai tambah dalam penggunaan perangkat elektronik sesuai dengan kebutuhan atau keinginan pengguna.

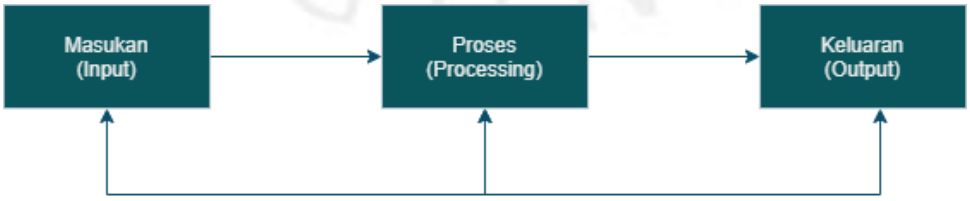
2.1.3 Pengertian Sistem

Dalam pengembangan sistem informasi, pemahaman tentang metode-metode yang digunakan tidak bisa dipisahkan dari konsep dan pengertian sistem itu sendiri. Mengerti konsep tersebut merupakan hal pertama yang krusial selama proses pengembangan sistem. Sebuah sistem tersusun dari beberapa komponen atau elemen yang saling terikat dan saling berpengaruh, bekerja bersama, dan memiliki tujuan yang telah ditetapkan. Sistem dapat berupa konsep abstrak atau fisik, dan di dalamnya terdapat komponen seperti *input*, proses, dan *output*. Tujuan utama dari sistem adalah mencapai suatu tujuan khusus melalui interaksi yang teratur dan terkoordinasi antara komponen-komponen yang ada pada sistem. Untuk lebih jelasnya ada beberapa definisi tentang apa itu sistem berikut penjelasannya:

Sistem merupakan beberapa elemen yang digabungkan untuk melakukan tujuan bersama. Sistem bisa berupa konsep abstrak atau benda nyata. Sistem merupakan kelompok komponen, baik yang berwujud fisik maupun yang tidak fisik, yang saling terhubung dan berfungsi bersama secara serasi untuk mencapai suatu tujuan khusus (Permana , Nur, & Rahma, 2022).

Sedangkan Prabowo (2020) mengartikan bahwa sistem merupakan komponen yang saling berkumpul atau beberapa elemen yang terhubung dan saling memengaruhi satu sama lain dalam bekerja sama untuk mencapai tujuan tertentu. Komponen-komponen dalam sistem umumnya meliputi masukan, proses pengolahan, dan keluaran. Berikut adalah gambar model komponen sistem:

- Suatu sub sistem terdiri dari berbagai komponen yang saling terikat dan berhubungan, berinteraksi untuk mencapai tujuan bersama.
- Elemen-elemen pada sistem saling bekerja sama secara harmonis.
- Komponen sistem mencakup input, proses, dan output, masing-masing dengan peran khusus dalam keseluruhan fungsi sistem.



Gambar 2. 1 Model Sistem (Prabowo, 2020, p. 2)

Gambar 2.1 di atas menjelaskan bahwa setiap sistem minimal harus terdiri dari tiga komponen utama, yaitu elemen masukan, proses pengelolaan, dan komponen keluaran. Selain aspek tersebut, penting juga untuk memperhatikan adanya mekanisme umpan balik atau kontrol dalam sistem tersebut.

2.1.4 Pengertian Informasi

Output atau hasil dari proses olah data yang memiliki nilai dan kegunaan bagi organisasi adalah bentuk pengertian dari informasi. Sebelum melalui proses pengolahan data dalam bentuk mentahnya tidak memiliki nilai yang signifikan tanpa melalui proses transformasi dan analisis yang tepat. Namun, ketika data diolah dengan menggunakan metode atau algoritma yang sesuai, informasi yang bernilai dapat dihasilkan (Hidayat, 2019). Informasi berguna untuk memberikan pemahaman tentang situasi, tren, pola, atau pada sebuah organisasi. Informasi merupakan aset yang cukup penting untuk organisasi, karena dapat berfungsi untuk pengambilan keputusan yang cerdas dan efektif, memungkinkan organisasi untuk mengoptimalkan operasional, meningkatkan efisiensi, dan mencapai tujuan bisnis yang ditetapkan.

2.1.5 Penjelasan Sistem Informasi

Sebuah proses yang mencakup pengumpulan, penyimpanan, dan analisis informasi untuk mencapai tujuan tertentu yang terdiri dari data masukan dan menghasilkan sebuah keluaran kemudian diterima oleh sistem lain merupakan definisi dari sistem informasi. Sistem ini juga mendukung kegiatan strategis di dalam sebuah organisasi untuk memandu tindakan dan pengambilan keputusan (Prehanto, S.Kom., M.Kom, 2020). Dalam sebuah organisasi, sistem informasi dirancang dan diterapkan untuk mendukung berbagai kegiatan dan proses bisnis yang ada. Tujuannya adalah agar kegiatan organisasi dapat berjalan dengan lancar, optimal, dan efisien, serta dapat mencapai tujuan. Sistem informasi berfungsi sebagai pendukung utama dalam mengumpulkan, mengelola, dan menganalisis data, yang kemudian diubah menjadi informasi yang bernilai.

Tujuan dan target utama dalam implementasi dan pengembangan sistem informasi adalah pada sistem informasi yang berbasis komputer. Tujuannya adalah untuk meningkatkan kecepatan, keakuratan, dan kualitas pengolahan data, dengan harapan dapat mencapai pengambilan keputusan yang lebih efisien dan efektif.

Meskipun demikian, sistem informasi berbasis komputer tidak selalu mengarah pada otomatisasi total (Permana , Nur, & Rahma, 2022). Hal ini dikarenakan ada aktivitas tertentu yang lebih baik dilakukan oleh manusia, sementara yang lain dapat dilakukan oleh mesin. Dalam praktiknya, dikembangkan sistem *hybrid* yang menggabungkan interaksi manusia dan komputer untuk mencapai hasil terbaik.

Dengan demikian, sistem informasi merupakan suatu komponen penting dalam keberhasilan dan pertumbuhan sebuah organisasi, karena membantu dalam pengolahan data dan informasi yang mendukung kegiatan bisnis serta mencapai tujuan organisasi.

2.1.6 Pengertian Penjualan

Penjualan adalah proses yang dilakukan oleh seorang penjual yang memiliki tujuan agar mendapatkan keuntungan dari transaksi penjualan barang atau jasa. Hal ini melibatkan perpindahan dari hak kepemilikan untuk barang atau jasa dari penjual ke pembeli. Terdapat sejumlah persyaratan yang harus dipenuhi agar penjualan dapat terjadi, termasuk adanya penjual dan calon pembeli, terjadinya interaksi dan persepsi, niat untuk melakukan pertukaran atau transaksi, serta adanya barang atau jasa, ataupun sebuah ide, rencana, dan prinsip yang diperdagangkan (Mulyadi, 2008).

Proses penjualan merupakan sebuah aspek yang sangat penting dan sering kali terjadi dalam operasional setiap organisasi ataupun perusahaan dagang. Proses penjualan bisa dilakukan bagi perusahaan dalam beberapa bentuk (Mulyadi, 2009) berikut merupakan dua jenis bentuk penjualan :

1. Penjualan Tunai

   Penjualan tunai adalah penjualan yang prosesnya dilakukan melalui tahap yang mewajibkan pembeli untuk membayar harga barang sepenuhnya sebelum barang diserahkan kepada pembeli. Kemudian setelah perusahaan menerima pembayaran, kepemilikan barang akan diserahkan kepada pembeli, yang kemudian transaksi penjualan tunai dicatat.

2. Penjualan Cicilan

   Penjualan cicilan merupakan proses penjualan barang atau jasa yang dilakukan melalui proses perjanjian yang pembayaran dilakukan secara bertahap atau cicilan. Hal tersebut sering dilakukan ketika barang atau jasa diserahkan kepada pembeli, penjual menerima pembayaran uang muka dan

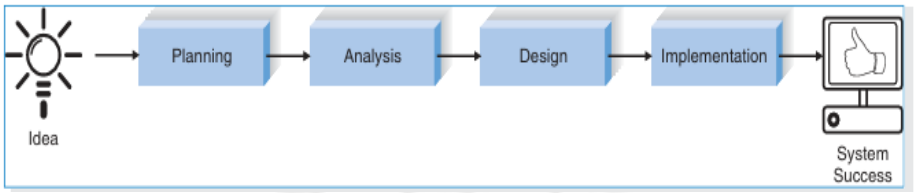
sisanya dibayar dalam bentuk cicilan. Dikarenakan penjual perlu menunggu beberapa tenggat waktu untuk menerima seluruh pembayaran, mereka mungkin menetapkan bunga atas saldo yang masih belum dibayarkan.

2.2 Teori Khusus

2.2.1 *Sistem Development Life Cycle* (SDLC)

Siklus hidup pengembangan sistem merupakan metode atau pendekatan untuk membantu dalam proses *planning*, mendesain, mengimplementasikan, menguji, dan memanajemen sistem informasi dengan cara terstruktur dan terorganisir (Valacich & George, 2017). Secara mendasar SDLC merupakan proses untuk membangun sebuah sistem baru atau juga bisa digunakan untuk mengembangkan sistem yang lama, SDLC memiliki tahapan yang harus dilakukan untuk mengembangkan sistem informasi dari awal hingga akhir.

SDLC terdiri dari empat tahap dasar dimulai dari tahapan untuk memahami mengapa suatu sistem informasi harus dibangun yaitu perencanaan hingga tahapan di mana sistem informasi sudah dibangun yaitu tahapan implementasi. Walaupun pada sebuah proyek untuk setiap fase SDLC beberapa tahap dilakukan dengan cara yang tidak sama, namun proyek tetap harus mencakup elemen-elemen dari keempat tahap ini. Masing-masing tahap memiliki rangkaian langkah-langkah yang bergantung pada metode tertentu untuk menghasilkan keluaran yang memberikan informasi yang lebih dalam mengenai proyek yang sedang dikembangkan (Dennis, Wixom, & Tegarden, 2015). Gambar di bawah menggambarkan tahapan dalam SDLC.



Gambar 2. 2 Sistem Development Life Cycle
(Dennis, Wixom, & Tegarden, 2015, p. 11)

Pada gambar 2.2. terlihat bahwa dari siklus hidup tersebut setiap fase terlihat seperti dilakukan secara berurutan, namun sebenarnya tidak, setiap fase yang dilakukan dimaksudkan untuk disesuaikan kembali dengan kebutuhan proyek. Dalam setiap fase SDLC, proyek dapat kembali ke fase sebelumnya jika diperlukan. Dalam SDLC, dimungkinkan juga untuk menyelesaikan beberapa aktivitas dalam satu fase

secara bersamaan dengan beberapa aktivitas di fase lainya (Valacich & George, 2017). Untuk lebih jelasnya berikut merupakan penjelasan dari setiap tahapan SDLC :

1. Perencanaan (Planning)

   Perencanaan atau *planning* adalah tahap untuk mengetahui lebih dalam kenapa sistem informasi perlu dibangun dan dikembangkan, yang selanjutnya menetapkan strategi pembangunan oleh tim proyek. Tahap ini memiliki dua langkah penting yaitu:

   a. Pelaksanaan awal proyek dan penilaian esensi bisnis

      1. Pelaksanaan Awal Proyek
         - Tujuan dan Ruang Lingkup: Menetapkan tujuan dan ruang lingkup proyek untuk memastikan pemahaman yang jelas tentang apa yang akan dicapai.
         - Identifikasi Stakeholders: Mengidentifikasi semua pemangku kepentingan yang terlibat dalam proyek.
         - Studi Kelayakan: Menganalisis kelayakan teknis, ekonomis, dan operasional proyek.
      2. Penilaian Esensi Bisnis
         - Analisis Bisnis: Menilai bagaimana sistem akan memberikan nilai bisnis dan mendukung tujuan strategis.
         - Analisis Biaya-Manfaat: Menghitung biaya proyek dibandingkan dengan manfaat yang diharapkan.
         - Penilaian Risiko: Mengidentifikasi dan merencanakan mitigasi risiko.

   b. Manajemen proyek

      1. Pembentukan Tim Proyek
         - Penugasan Peran: Menetapkan peran dan tanggung jawab anggota tim proyek.
         - Perencanaan Sumber Daya: Merencanakan sumber daya yang diperlukan seperti anggaran, waktu, dan tenaga kerja.
      2. Penyusunan Rencana Proyek
         - Rencana Proyek: Menyusun jadwal, tahapan, dan milestone proyek.

- Struktur Pemecahan Kerja: Memecah proyek menjadi tugas-tugas yang lebih kecil dan dapat dikelola.
- Rencana Manajemen Risiko: Menyusun strategi untuk mengidentifikasi dan mengelola risiko proye

2. Analisis (Analysis)

Dalam pengembangan sistem, tahap analisis bertujuan untuk mengidentifikasi pengguna sistem, menentukan fungsionalitas yang dibutuhkan, dan menentukan konteks penggunaan sistem. Fase ini melibatkan tiga langkah utama:

a. Strategi analisis

Proses ini untuk memahami dan mengevaluasi sistem yang ada serta mengidentifikasi kebutuhan untuk sistem baru. Ini melibatkan pemetaan proses bisnis saat ini, penilaian kekuatan dan kelemahan sistem yang ada, dan identifikasi kebutuhan bisnis yang belum terpenuhi. Strategi ini membantu menetapkan arah yang jelas untuk analisis lebih lanjut dan memastikan bahwa semua aspek penting dari sistem diperiksa secara menyeluruh.

b. Pengumpulan persyaratan

Pengumpulan persyaratan adalah langkah untuk mengidentifikasi dan mendokumentasikan kebutuhan pengguna dan fungsionalitas yang diharapkan dari sistem baru. Ini melibatkan berbagai teknik seperti wawancara, survei, observasi, dan analisis dokumen untuk mengumpulkan informasi dari pemangku kepentingan. Penggabungan hasil analisis

Penggabungan hasil analisis adalah proses untuk menyatukan semua temuan dari strategi analisis dan pengumpulan persyaratan ke dalam satu konsep yang kohesif untuk sistem baru. Ini termasuk penyusunan model bisnis, diagram alur proses, dan spesifikasi sistem yang lebih detail. Langkah ini memastikan bahwa semua kebutuhan dan masalah yang diidentifikasi selama analisis dipertimbangkan dalam desain sistem baru, dan menyediakan dasar yang kuat untuk tahap desain selanjutnya.

3. Desain (Design)

Tahap desain dalam pengembangan sistem adalah proses merancang bagaimana sistem akan bekerja dan bagaimana komponennya akan berinteraksi. Ini mencakup definisi struktur sistem, antarmuka pengguna, komponen perangkat keras dan perangkat lunak, serta database. Tujuannya adalah menghasilkan cetak biru yang mendetail untuk pengembangan dan implementasi sistem. Tahap ini memiliki empat langkah utama:

a. Desain Strategi

   Desain strategi adalah langkah untuk menentukan pendekatan umum. Proses ini mencakup pemilihan platform *software* dan *hardware*, metodologi pengembangan, standar coding, dan alat bantu pengembangan yang akan digunakan. Desain strategi memastikan bahwa pilihan teknologi dan metode yang dipilih sesuai dengan kebutuhan bisnis dan persyaratan teknis proyek.

b. Spesifikasi data *base* dan *file*

   Spesifikasi database dan file melibatkan perancangan struktur penyimpanan data yang efisien dan efektif untuk sistem. Ini mencakup definisi skema database, tabel, relasi antar tabel, dan indeks. Langkah ini juga menentukan format file, lokasi penyimpanan, dan metode akses data yang akan digunakan. Tujuannya adalah memastikan bahwa data dapat disimpan, diakses, dan dikelola dengan optimal sesuai kebutuhan sistem.

c. Perancangan arsitektur

   Perancangan arsitektur adalah proses mendefinisikan struktur keseluruhan dari sistem, termasuk komponen-komponen utama dan interaksinya. Ini mencakup pemetaan modul-modul sistem, lapisan aplikasi, komponen middleware, dan infrastruktur jaringan. Tujuannya adalah memastikan bahwa sistem dirancang dengan cara yang terstruktur, scalable, dan dapat diandalkan, serta memenuhi kebutuhan kinerja dan keamanan.

d. Pengembangan desain program

   Pengembangan desain program adalah langkah untuk merinci spesifikasi teknis dari masing-masing modul perangkat lunak. Proses ini mencakup pembuatan diagram alur, pseudocode, dan spesifikasi fungsional untuk setiap komponen perangkat lunak. Tujuannya adalah memberikan

panduan yang jelas bagi pengembang untuk menulis kode program yang memenuhi persyaratan sistem dan desain yang telah ditentukan.

Hasil dari fase desain, termasuk desain arsitektur, antarmuka, spesifikasi *database* dan file, serta desain program, diserahkan kepada tim pemrograman untuk diimplementasikan. Setelah fase ini selesai, analisis kelayakan dan rencana proyek diperbarui, dan keputusan mengenai kelanjutan proyek diambil oleh sponsor proyek dan komite persetujuan.

4. Implementasi (Implementation)

   Tahap implementasi dalam pengembangan sistem adalah proses mewujudkan desain menjadi sistem yang berfungsi penuh. Proses ini mencakup pembuatan kode program, pengujian, instalasi sistem, dan pelatihan pengguna. Tujuannya adalah memastikan bahwa sistem yang dikembangkan dapat berjalan sesuai dengan spesifikasi yang telah ditentukan dan siap digunakan oleh pengguna. Tahap ini memiliki tiga langkah utama:

   a. Konstruksi sistem

      Konstruksi sistem adalah langkah di mana pengembang menulis kode program berdasarkan desain yang telah dibuat. Proses ini mencakup pengkodean, *debugging*, dan unit testing untuk memastikan bahwa setiap komponen perangkat lunak berfungsi dengan benar. Langkah ini juga melibatkan integrasi komponen-komponen yang berbeda untuk membentuk sistem yang utuh.

   b. Instalasi sistem

      Instalasi sistem adalah proses mengimplementasikan perangkat lunak dan perangkat keras yang telah dibangun ke lingkungan operasional. Proses ini mencakup konfigurasi server, instalasi aplikasi, migrasi data, dan pengaturan jaringan. Tujuannya adalah memastikan bahwa sistem baru terpasang dengan benar dan siap untuk digunakan oleh pengguna.

   c. Rencana dukungan sistem

      Rencana dukungan sistem melibatkan penyiapan dukungan operasional dan pemeliharaan untuk memastikan sistem tetap berjalan dengan baik setelah implementasi. Proses ini mencakup pelatihan pengguna, dokumentasi sistem, prosedur *backup*, dan rencana pemeliharaan berkala.

Tujuannya adalah memberikan bantuan yang diperlukan kepada pengguna dan memastikan sistem dapat dioperasikan dan dipelihara dengan efisien.

2.2.2 Metode *Rapid Application Development* (RAD)

Metode ini merupakan pendekatan pengembangan perangkat lunak yang menekankan pada perancangan dan implementasi prototipe dengan siklus waktu yang cepat serta memanfaatkan *feedback* dari *user* yang sering terjadi selama tahapan pengembangan dan pengujian yang memakan waktu. Dengan mengadopsi pendekatan ini, pengembang dapat melaksanakan beberapa iterasi dan pembaruan terhadap perangkat lunak dengan cepat, tanpa harus memulai proses pengembangan dari awal setiap kali. Model RAD muncul sebagai alternatif saat para pengembang menyadari bahwa model pengembangan *waterfall* tradisional tidak selalu efisien (Saputra & Aprilian, 2020, hal. 63).

Metode ini memiliki beberapa elemen penting sehingga menjadikannya metodologi yang berbeda dengan metode lain, seperti penggunaan prototipe, pengembangan secara *iteratif* dengan *time boxing*, anggota tim yang terlibat, pendekatan manajemen, dan penggunaan *tools* RAD. Berikut merupakan penjelasan setiap elemen yang ada di RAD (Prabowo, 2020).

1. Prototipe (Prototyping)

   Pembuatan prototipe adalah salah satu bagian penting dari metode RAD gunanya untuk memvalidasi desain terhadap kebutuhan pengguna. Selain itu tujuannya adalah untuk menghasilkan modul dan model perangkat lunak awal dengan cepat untuk mendapatkan umpan balik dan menyempurnakan kebutuhan.

2. Pengembangan Iterasi (Iterative Development)

   Iterative Development adalah proses di mana sistem dikembangkan dalam siklus pendek untuk menciptakan versi yang lebih fungsional. Setiap versi dievaluasi oleh klien untuk menentukan persyaratan versi selanjutnya.

3. Tinju waktu *(*Time Boxing)

   Proses ini merupakan langkah yang mana beberapa fitur tertentu pengembangannya di *hold* untuk versi aplikasi di masa depan supaya versi saat ini dapat selesai tepat waktu. Ketepatan waktu menjadi kunci dalam RAD,

karena jika tidak, ruang lingkup proyek dapat menjadi terlalu besar sehingga memperpanjang waktu iterasi pembangunan.

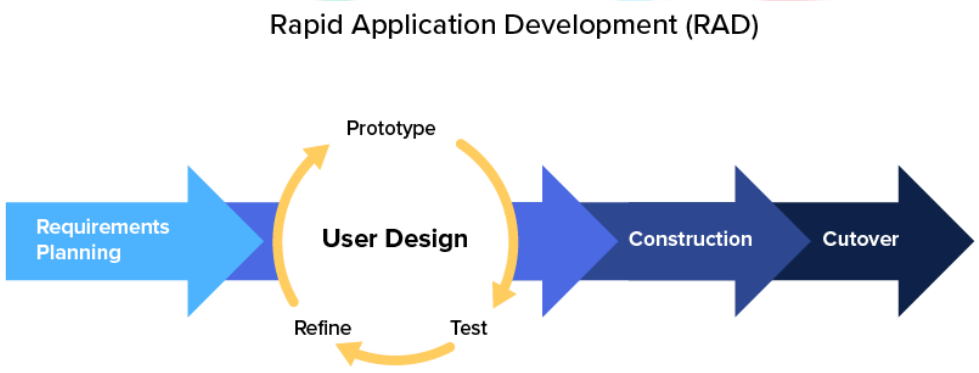
4. Anggota Tim (Team Member)

   Metode ini menyarankan pekerjaan untuk jumlah anggota tim yang tidak terlalu banyak tetapi anggota yang cukup berpengalaman, fleksibel, termotivasi dan mampu melakukan berbagai pekerjaan.

5. Alat RAD (RAD Tools)

   Salah satu tujuan utama dari metode ini adalah bertujuan untuk memanfaatkan teknologi terbaru supaya mempercepat proses pengembangan. Meskipun teknologi pada tahun 1980-an mungkin sudah usang, fokus RAD terhadap penggunaan alat-alat terbaru tetap sama pentingnya pada masa kini seperti saat metodologi tersebut pertama kali dikembangkan.

Terdapat 4 fase dalam pengembangan dengan metode RAD di antaranya adalah *Requirements*, *Planning, User Design, Rapid Construction,* dan *cutover* (Hussain, Tyagi, & Peng, 2023). Pada gambar 2.3 menampilkan tahapan dari metode RAD.



Gambar 2. 3 Rapid Application Development methodology
(Hussain, Tyagi, & Peng, 2023, p. 6)

1. *Requirements Planning*

   Tahap ini dimulai dengan pertemuan di lokasi proyek. Meskipun tahap desain relatif singkat dibandingkan dengan metode manajemen proyek lainnya, tahap ini adalah langkah penting untuk keberhasilan proyek secara keseluruhan. Pada tahap ini, pengembang, pelanggan (pengguna perangkat lunak), dan anggota tim saling berkomunikasi untuk menetapkan tujuan dan harapan proyek, serta

menyelesaikan masalah yang ada dan potensi masalah selama proses pengembangan.

Analisis dasar pada tahap ini meliputi:

- Mengidentifikasi masalah yang ada
- Menentukan persyaratan sistem
- Memenuhi persyaratan dengan persetujuan semua pihak

Penting bagi setiap bagian yang terlibat dalam pengembangan untuk mengevaluasi tujuan dan persyaratan proyek serta bobotnya. Persetujuan dari semua *stackholder* utama dan pengembang dapat menghindari masalah komunikasi dan permintaan perubahan yang mahal. Pengalaman ini memastikan bahwa tidak ada potensi masalah yang terlewatkan.

2. *User Design*

   Setelah desain ditentukan, dilanjutkan dengan pengembangan dan membangun desain sistem melalui iterasi prototipe. Proses ini merupakan inti dari pendekatan RAD yang membedakannya dengan metode manajemen proyek lainnya. Pada proses ini, klien dapat berdiskusi dan mengevaluasi kepada pengembang untuk memastikan bahwa persyaratan sistem sudah tepat pada pengembangan desain.

   Sama halnya dengan proses perancangan *software* yang mana setiap evaluasi dapat diperbaiki pada setiap desain, dan pada tahap *user design* pengguna dapat mengevaluasi desain prototipe yang sudah dibuat untuk memastikan bahwa prototipe tersebut memenuhi persyaratan sistem. Semua kesalahan dan *bug* diperbaiki dalam proses berulang. Pengembang mendesain dan membuat prototipe, pengguna mengujinya, dan kemudian mereka bertemu untuk mendiskusikan efektivitas dan tidak efektifan metode tersebut. Metode ini memungkinkan pengembang untuk memodifikasi model hingga mereka puas dengan desainnya. Pengembang perangkat lunak dan pelanggan dapat belajar dari pengalaman dan memastikan bahwa hal-hal tertentu tidak akan salah.

3. *Rapid Construction*

Fase ketiga ini cukup penting karena klien terus memberikan masukan selama proses berjalan. Pada tahap ini, prototipe dan sistem beta yang dihasilkan dari fase *User Design* ditransformasikan menjadi *working model*. Semua masalah yang penting dalam sistem dan perubahan ditangani dalam fase *User Design*, memungkinkan pengembang untuk menghasilkan *working model* akhir lebih cepat dibandingkan dengan metode manajemen proyek tradisional. Tahap ini dibagi menjadi beberapa langkah kecil:

- Mempersiapkan Konstruksi Cepat
  Mengatur segala kebutuhan untuk memulai pengembangan.
- Mengembangkan Program dan Aplikasi
  Membuat aplikasi sesuai desain yang telah disetujui.
- Pengkodean
  Menulis kode program sesuai dengan spesifikasi.
- Pengujian Unit, Integrasi, dan Sistem
  Memastikan setiap bagian aplikasi berfungsi dengan baik dan terintegrasi dengan sempurna.

Tim pengembangan perangkat lunak, termasuk pemrogram, penguji, dan pengembang, bekerja bersama pada tahap ini untuk memastikan semua berjalan lancar dan hasil akhirnya sesuai dengan harapan dan tujuan pelanggan. Langkah ini sangat penting karena pelanggan tetap dapat memberikan kontribusi mereka selama proses berlangsung. Mereka dapat menyarankan perubahan dan ide-ide baru untuk menyelesaikan masalah yang muncul.

4. *Cutover*

Cutover atau peralihan adalah tahap dalam perakitan di mana produk akhir diluncurkan. Fase ini mencakup peralihan data dan kode program dari lingkungan pengembangan ke lingkungan produksi. Seluruh perubahan telah disesuaikan, dan pengguna dapat terus memantau sistem untuk meminimalisir kesalahan pada sistem tersebut.

2.2.3 *Object-Oriented Analysis and Design* (OOAD)

Konsep OOAD merupakan pendekatan untuk mengembangkan sistem *software* yang objektif pada untuk sebuah objek sebagai unit dasar analisis dan desain (Wazlawick, 2014). Pendekatan ini melibatkan beberapa langkah utama, yaitu:

1. Analisis Berorientasi Objek (Object-Oriented Analysis)
   Mengidentifikasi entitas atau objek dalam sistem berdasarkan persyaratan dan spesifikasi yang telah dikumpulkan. Ini mencakup pemahaman tentang fungsi sistem dan bagaimana berbagai objek berinteraksi satu sama lain.
2. Desain Berorientasi Objek (Object-Oriented Design)
   Mengembangkan perancangan rinci berdasarkan hasil analisis. Ini mencakup pembuatan diagram dan model yang menunjukkan struktur dan perilaku objek dalam sistem, serta bagaimana mereka berinteraksi untuk memenuhi persyaratan sistem.

OOAD tidak hanya tentang menggunakan bahasa pemrograman berorientasi objek, tetapi juga tentang mengadopsi pola pikir dan teknik yang mendukung prinsip-prinsip berorientasi objek (Wazlawick, 2014). Beberapa prinsip penting dalam OOAD termasuk:

- Enkapsulasi
  Menyembunyikan detail implementasi dari objek dan hanya mengekspos apa yang diperlukan melalui antarmuka yang ditentukan.
- Abstraksi
  Mengidentifikasi esensi dari objek dan mengabaikan detail yang tidak relevan untuk memahami dan memodelkan sistem.
- Pewarisan
  Mengatur kelas-kelas dalam hierarki yang memungkinkan kelas anak mewarisi sifat dan perilaku dari kelas induk.
- *Polimorfisme*
  Memungkinkan objek-objek yang tidak berhubungan untuk merespon terhadap masukan yang sama dengan cara yang tepat dengan tipe mereka.

Pendekatan OOAD juga menekankan pentingnya penggunaan teknik seperti delegasi dan penugasan tanggung jawab untuk menciptakan kode yang lebih modular, dapat digunakan kembali, dan memiliki kopling rendah antara komponen-komponen sistem. Dalam praktiknya, OOAD tidak hanya tentang membuat diagram atau menggunakan alat *Computer-Aided Software Engineering* (CASE), tetapi juga tentang benar-benar memahami dan menerapkan prinsip-prinsip berorientasi objek.

2.2.4 *Unified Modelling Language* (UML)

Unified Modeling Language (UML) adalah standar industri untuk pemodelan sistem perangkat lunak, yang memvisualisasikan, menentukan, membuat, dan mendokumentasikan komponen sistem. UML mencakup berbagai diagram, seperti diagram kelas, urutan, kasus penggunaan, aktivitas, komponen, deployment, objek, dan paket, yang membantu menggambarkan aspek-aspek struktural dan perilaku dari sistem. Dengan UML, pengembang dapat membuat cetak biru yang standar dan mudah dipahami oleh semua pemangku kepentingan, memfasilitasi perencanaan, perancangan, dan verifikasi sistem perangkat lunak secara efisien. (Dennis, Wixom, & Tegarden, 2015). Di dalam UML terdapat dua jenis konsep diagram yaitu *Structure Diagrams* dan *Behavioral Diagrams*. *Structure Diagrams* mencakup berbagai jenis diagram seperti kelas, objek, dan lainnya, sementara *Behavioral Diagrams* termasuk aktivitas, urutan, komunikasi, dan lainnya. Pada tabel 2.1 merupakan Ringkasan mengenai *Structure Diagrams* dan *Behavioral Diagrams.*

Tabel 2. 1 *Jenis diagram UML*

Nama Diagram	Fungsi	Fase Primer
	Structure Diagrams	
Class	Mengilustrasikan cetak biru atau template yang mendefinisikan struktur dan perilaku objek.	*Analysis, Design*
Object	Mengilustrasikan suatu entitas yang memiliki atribut (data) dan metode (fungsi atau perilaku) yang bekerja pada data tersebut. Objek	*Analysis, Design*

	merupakan instansiasi dari kelas, di mana kelas adalah cetak biru atau template yang mendefinisikan atribut dan metode umum yang dimiliki oleh objek-objek dari kelas tersebut.	
Package	Elemen yang digunakan untuk mengelompokkan elemen-elemen model terkait, seperti kelas, diagram, dan sub-paket lainnya, menjadi satu unit yang terorganisir	*Analysis, Design, Implementation*
Deployment	Merupakan jenis diagram yang menunjukkan konfigurasi fisik dari perangkat keras dan perangkat lunak sistem.	*Physical, Design, Implementation*
Component	Digunakan untuk menggambarkan bagaimana komponen perangkat lunak saling berinteraksi dan bagaimana mereka disusun dalam sistem.	*Physical Design, Implementation*
Composite Structure Design	Jenis diagram yang digunakan untuk menggambarkan struktur internal kelas, komponen, atau kolaborasi.	*Analysis, Design*
Profile	Mekanisme perluasan yang memungkinkan pengguna untuk menyesuaikan UML agar sesuai dengan kebutuhan domain tertentu atau metode pemodelan tertentu.	-
Behavioral Diagrams		
Activity	Menggambarkan aliran aktivitas atau langkah-langkah dalam suatu proses bisnis atau alur kerja.	*Analysis, Design*

Sequence	Menunjukkan bagaimana objek berinteraksi satu sama lain dalam urutan waktu tertentu melalui pesan.	*Analysis, Design*
Communication	Menunjukkan interaksi antara objek dan urutan pesan yang ditukar, menekankan struktur kolaborasi antar objek.	*Analysis, Design*
Interaction Overview	Menggabungkan elemen-elemen dari diagram aktivitas dan diagram interaksi (seperti sequence diagram) untuk memberikan gambaran umum tentang alur kontrol dalam skenario interaksi kompleks.	*Analysis, Design*
Timing	Menunjukkan perubahan keadaan atau kondisi dari satu atau lebih objek sepanjang waktu, menekankan waktu dan durasi peristiwa.	*Analysis, Design*
Protocol State Machine	Menggambarkan urutan hukum dari pesan yang diterima dan dikirim oleh objek, menekankan aturan protokol komunikasi.	*Analysis, Design*
Use-Case	Menggambarkan fungsionalitas sistem dari perspektif pengguna (aktor) dan interaksi antara pengguna dan sistem.	*Analysis*,

Sumber : System Analysis & Design An Object-Oriented Approach with UML, 2015

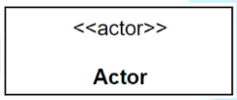
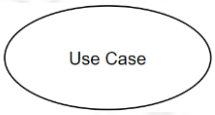
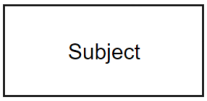
Berdasarkan permodelan diagram yang ada di UML berikut merupakan penjelasan mengenai diagram dan notasi-notasi yang dipakai sebagai alat untuk mengembangkan aplikasi atau sistem :

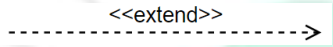
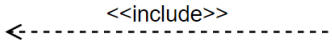
a. *Use Case* Diagram

Use case merupakan salah satu cara yang sederhana dan lugas untuk menggambarkan fungsionalitas inti pada sistem dengan berbagai jenis aktor internal maupun eksternal yang saling berinteraksi dengan sistem yang diimplementasikan. Kasus Diagram dapat menangani komunikasi antara pengguna dan pemangku kepentingan untuk menjelaskan atau menerima pesan dari sistem.

Diagram kasus dapat digunakan untuk pengumpul dan menentukan persyaratan yang lebih spesifik untuk sistem, dan mampu membuat pengguna supaya menghasilkan persyaratan yang lebih spesifik (Dennis, Wixom, & Tegarden, 2015). Diagram kasus dapat memberikan gambaran yang lebih detail tentang bagaimana interaksi antara aktor (pengguna) serta fungsi-fungsi sistem yang akan digunakan oleh mereka. Hubungan tersebut terdiri atas hubungan *association, include, extend,* dan *generalization.* Pada tabel 2.2 menjelaskan aturan penggunaan notasi dan sintaksis untuk *use case* diagram.

Tabel 2. 2 *Sintaksis dan notasi Use case Diagram*

Notasi	Nama Notasi	Keterangan
 	*Actor*	• Simbol aktor pada use case diagram mewakili entitas eksternal yang berinteraksi dengan sistem.
	*Use case*	• Simbol use case pada use diagram kasus mengilustrasikan fungsionalitas atau modul yang tersedia dalam sistem yang bisa digunakan oleh aktor.
	*Boundary*	• Simbol boundary pada use case diagram menunjukkan batasan sistem yang memisahkan apa yang ada di dalam sistem (use case dan aktor internal) dari entitas eksternal (aktor luar).

	Association Relationship	• Simbol garis asosiasi pada use case diagram menunjukkan hubungan antara pengguna (Actors) dengan use case, menandakan seorang atau beberapa aktor terlibat dan berinteraksi dengan use case yang terkait.
	*Extend Relationship*	• Simbol <<extend>> pada use case diagram menunjukkan tambahan memperluas fungsionalitas use case utama tertentu. Garis putus-putus dengan panah menunjuk ke use case yang *extend*.
	*Include Relationship*	• Notasi atau simbol <<include>> menunjukkan bahwa *use case* utama selalu menyertakan perilaku dari use case tambahan. Garis putus-putus dengan panah menunjuk ke use case yang disertakan.
	*Generalization Relationship*	• Simbol panah segitiga pada use case diagram menunjukkan hubungan generalisasi atau pewarisan, di mana use case atau aktor di bagian bawah panah mewarisi karakteristik dari use case atau aktor di bagian atas panah.

Sumber : System Analysis & Design An Object-Oriented Approach with UML, 2015

b. Deskripsi Use Case

Format ini digunakan untuk merincikan alur sebuah kasus penggunaan. Dokumentasi ini dibuat untuk meningkatkan pemahaman pengguna tentang keseluruhan proses bisnis dan bagaimana sistem mendukungnya, dengan tujuan menciptakan sistem yang komprehensif yang sesuai dengan kebutuhan (Satzinger, Jackson, &

Burd, 2016). Metode ini membantu dalam menjelaskan bagaimana sebuah sistem berinteraksi dengan pengguna untuk menyelesaikan tugas atau fungsionalitas tertentu, memastikan pemahaman yang mendalam tentang fungsionalitas sistem dan bagaimana hal tersebut sesuai dengan kebutuhan pengguna dalam suatu skenario atau proses tertentu. Gambar 2.4 di bawah merupakan contoh deskripsi *use case* yang dikembangkan dari sebuah fitur "Buat Akun Pelanggan".

Use case name:	Buat akun pelanggan.	
Scenario:	Buat akun pelanggan online.	
Triggering event:	Pelanggan baru ingin membuat akun secara online	
Brief description:	Pelanggan online membuat akun pelanggan dengan memasukkan informasi dasar dan kemudian menindaklanjuti dengan satu atau lebih alamat dan kartu kredit atau debit.	
Actors:	Pelanggan	
Related use cases:	-	
Stakeholders:	Accounting, Staf penjualan, Sales.	
Preconditions:	Subsistem Akun Pelanggan harus tersedia. Layanan otorisasi kredit/debit harus tersedia.	
Postconditions:	Pelanggan harus dibuat dan disimpan. Satu atau lebih Alamat harus dibuat dan disimpan. Informasi kartu kredit/debit harus divalidasi. Akun harus dibuat dan disimpan. Alamat dan Rekening harus dikaitkan dengan Pelanggan.	
Flow of activities:	**Actor**	**System**
	1. Pelanggan ingin membuat akun pelanggan dan memasukkan informasi dasar pelanggan. 2. Pelanggan memasukkan satu atau lebih alamat. 3. Pelanggan memasukkan kartu kredit/debit informasi.	1.1 Sistem menciptakan pelanggan baru. 1.2 Perintah sistem untuk pelanggan memasukkan alamat. 2.1 Sistem membuat alamat. 2.2 Sistem meminta kredit/debit kartu. 3.1 Sistem membuat akun. 3.2 Sistem memverifikasi otorisasi untuk kartu kredit/debit. 3.3 Sistem mengasosiasikan pelanggan, alamat, dan akun. 3.4 Sistem mengembalikan pelanggan yang valid Detail akun.
Exception conditions:	1.1 Data dasar pelanggan tidak lengkap. 2.1 Alamatnya tidak valid. 3.2 Informasi kredit/debit tidak valid.	

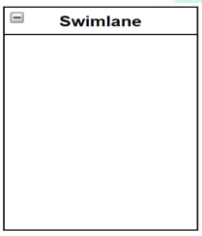
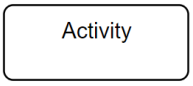
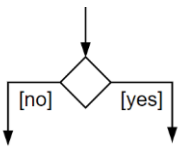
Gambar 2. 4 Contoh Use Case Description
Sumber : Systems Analysis and Design in a Changing World, Seventh Edition, 2016

c. *Activity* Diagram

Dalam mendokumentasikan rincian *use case*, salah satu metode yang sering digunakan adalah menggunakan *activity* diagram atau diagram aktivitas. *Activity* diagram digunakan untuk memodelkan berbagai proses yang terlibat dalam suatu *use case*. Tujuan dari permodelan ini adalah untuk mendokumentasikan cara interaksi antara

pengguna dan sistem, serta bagaimana pengguna dapat berinteraksi dengan sistem untuk menyelesaikan aktivitas atau tugas tertentu dalam satu *use case.* Diagram aktivitas membantu dalam menggambarkan urutan langkah atau prosedur yang dilakukan oleh pengguna atau sistem untuk mencapai tujuan yang diinginkan dalam skenario *use case* tersebut (Satzinger, Jackson, & Burd, 2016). Berikutnya pada tabel 2.4 merupakan sintkasi dan simbol notasi yang dipakai dalam memodelkan diagram aktivitas.

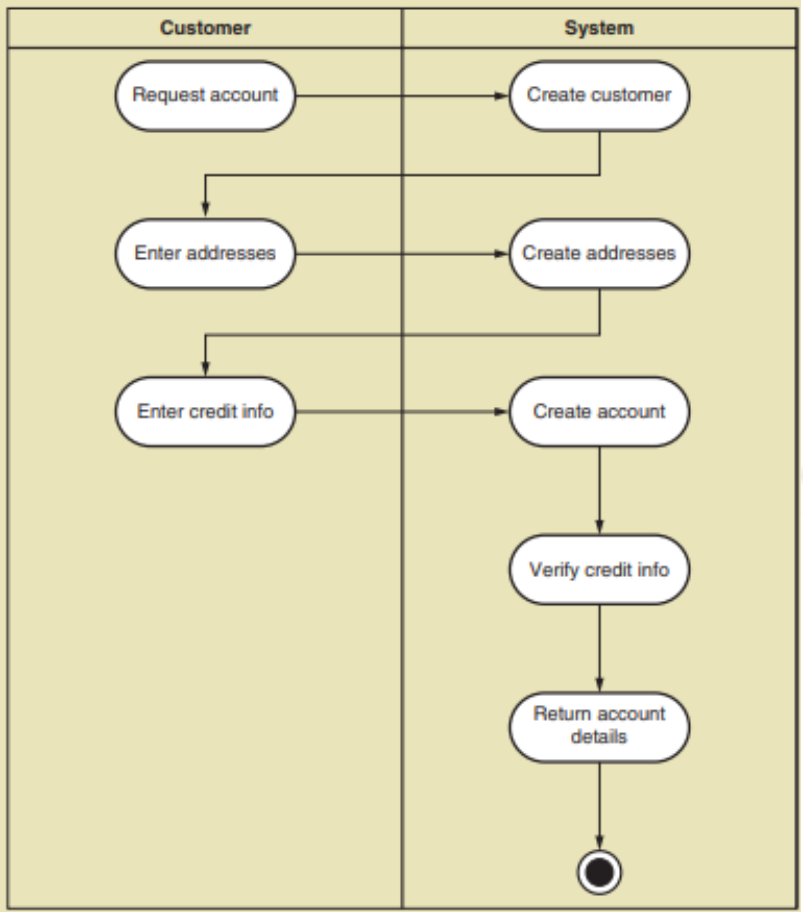
Tabel 2. 3 *Sintaksis Activity Diagram*

Notasi	Nama Notasi	Keterangan
	*Swimlane*	• Mewakili aktor yang melakukan aktivitas-aktivitas • Jika terdapat beberapa *swimlane* digambarkan sejajar dengan *swimlane* lainya.
	*Initial Node*	• Mewakili alur awal berjalannya sistem dimulai. • Hanya memiliki satu garis untuk berinteraksi dengan *activity*
	*Final Node*	• Mewakili berakhirnya alur sistem atau aktivitas.
	*Activity*	• Mewakili aktivitas-aktivitas individu dalam alur kerja
	*Synchronization bar*	• Digambarkan dengan simbol garis tebal dan solid
	*Decision activity*	• mewakili aktivitas-aktivitas individu dalam alur kerja. • Digambarkan dengan bentuk belah ketupat / berlian

→	*Transition arrow*	• Sebagai penghubung yang mewakili urutan dari setiap *activity*.

Sumber : Systems Analysis and Design in a Changing World, Seventh Edition, 2016

Dalam permodelan *activity diagram* terdapat beberapa kasus, terkadang *activity diagram* dapat menggantikan bagian alur aktivitas dalam *use case description* dan terkadang dibuat untuk melengkapi *use case description* (Satzinger, Jackson, & Burd, 2016, p. 137). Gambar 2.4 merupakan contoh *activity diagram* untuk *use case* "Buat Akun Pelanggan". Dalam contoh ini, terdapat dua *swimlane,* satu untuk pelanggan dan satu lagi untuk sistem. Pelanggan memiliki tiga aktivitas, dan sistem memiliki lima aktivitas.

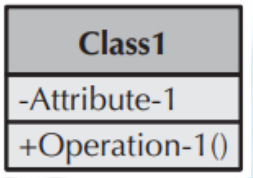


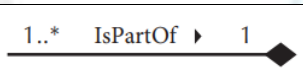
Gambar 2. 5 Contoh Activity Diagram *(Satzinger, Jackson, & Burd, 2016)*

d. Diagram Kelas (Class Diagram)

Diagram kelas merupakan representasi kelas-kelas yang saling berelasi dan konsisten di dalam sistem sepanjang waktu. Diagram ini mengilustrasikan kelas-kelas yang mencakup perilaku dan status, serta hubungan yang terbentuk di antara kelas-kelas tersebut. Bagian ini menguraikan unsur-unsur dari diagram kelas, berbagai metode untuk menyederhanakan diagram kelas, dan alternatif lain dari diagram kelas yang disebut diagram objek. Diagram kelas adalah model statis yang membantu dalam memvisualisasikan struktur dasar dari sistem yang terdiri dari objek-objek yang terlibat beserta hubungan antar objek tersebut (Dennis, Wixom, & Tegarden, 2015). Pada tabel 2.5 menunjukkan sintaksis dan notasi dalam diagram ini.

Tabel 2. 4 *sintaksis dan notasi diagram kelas*

Notasi/Sintaksis	Nama notasi	Keterangan
	Kelas	• *Container class* merupakan blueprint atau template untuk membuat objek. • Dalam penamaan kelas nama kelas ditulis menggunakan *font bold*. • Terdapat berapa atribut di bagian *container* tengahnya • Bagian bawah dari simbol kelas menampilkan daftar operasi atau metode yang dimiliki oleh kelas tersebut. • Tidak mencantumkan operasi atau metode yang umum untuk semua kelas secara langsung.
Attribute name	Atribut	• Merupakan properti yang menunjukkan kondisi atau status suatu objek. • Bisa diturunkan dari atribut yang berbeda, ditunjukkan dengan cara tertentu. • Ditandai dengan garis miring di depan nama atribut.
Operation Name ( )	Operasi	• Menunjukkan aktivitas atau fungsi yang bisa dieksekusi oleh kelas.

		• Bisa dikategorikan sebagai konstruktor, kueri, atau operasi yang memperbarui data. • Memiliki tanda kurung yang mungkin mencakup parameter atau detail yang diperlukan untuk menjalankan operasi.
	Asosiasi	• Menunjukkan koneksi antara berbagai kelas atau antara satu kelas dengan dirinya sendiri. • Bisa melibatkan satu atau lebih kelas. • Mencakup simbol multiplisitas yang mengindikasikan jumlah minimum dan maksimum instance dari suatu kelas yang dapat berhubungan dengan instance dari kelas lain..
	Generalisasi	• Adalah bentuk hubungan antara beberapa kelas..
	Agregasi	• Merupakan bagian logis dari koneksi antara beberapa kelas atau antara satu kelas dengan dirinya sendiri. • Adalah bentuk khusus dari asosiasi.
	Komposisi	• Merupakan komponen fisik dari hubungan antara beberapa kelas atau antara satu kelas dengan dirinya sendiri. • Adalah jenis khusus dari asosiasi

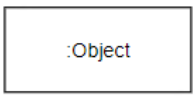
Sumber : System Analysis & Design An Object-Oriented Approach with UML, 2015

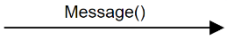
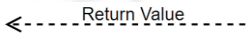
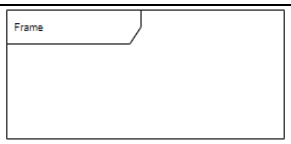
e. *Squence* Diagram

Diagram urutan digunakan untuk memvisualisasikan aliran informasi yang masuk dan keluar dari bagian sistem. Diagram ini mencatat masukan dan keluaran, serta mengidentifikasi interaksi antara aktor dan sistem. *Squence* diagram merupakan alat untuk membantu desain awal antar muka pengguna dengan mengidentifikasi informasi spesifik yang mengalir dari pengguna ke dalam sistem dan

informasi yang mengalir keluar dari sistem kembali ke pengguna (Satzinger, Jackson, & Burd, 2016). Manfaat utama dari diagram urutan adalah kemampuan untuk menata pesan dari atas ke bawah untuk menekankan urutan pengaktifan. Pada tabel 2.6 merupakan daftar sintaksis dan notasi dalam diagram urutan.

Tabel 2. 5 sintaksis dan simbol sequence diagram

Sintaksis/Notasi	Nama notasi	Keterangan
	*Actor*	• Mewakili individu (atau peran) yang berinteraksi dengan sistem. • Berpartisipasi dalam urutan dengan mengirim dan/atau menerima pesan. • Digambarkan dengan simbol berupa figur batang.
	*Object*	• Terlibat dalam urutan dengan mengirim dan/atau menerima pesan. • Diilustrasikan dengan simbol persegi panjang yang memuat nama objek. • Titik dua sebelum nama kelas yang digarisbawahi adalah bagian dari notasi objek yang sering digunakan, namun opsional, untuk menunjukkan bahwa objek tersebut adalah instance dari kelas yang tidak disebutkan namanya.

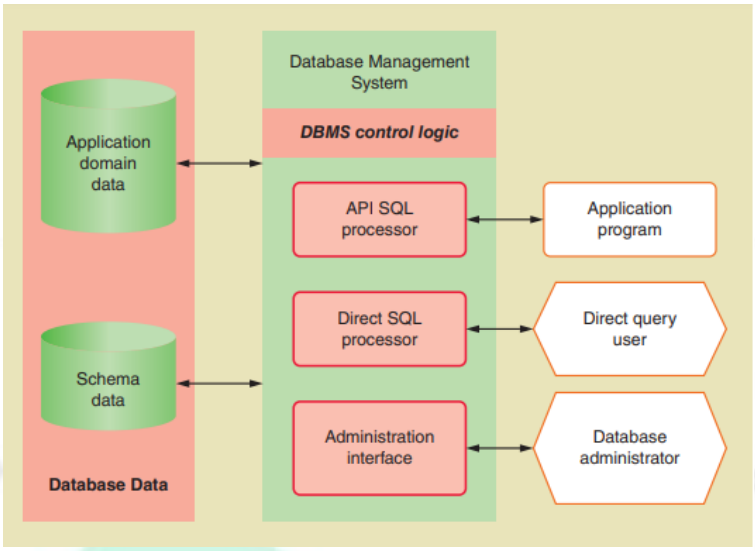
	Object Lifeline	• Menampilkan *sequence message* dari bagian atas menuju bagaian bawah. • Diilustrasikan dengan garis putus-putus vertikal.
	*Activity Lifeline*	• Menunjukkan ketika suatu objek mengirim atau menerima pesan. • Diilustrasikan dengan simbol persegi panjang sempit yang terletak di atas garis hidup.
	*Message*	• Mengirim informasi dari satu objek ke objek lainnya. • Panggilan operasi diberi label dengan pesan yang dikirim dan ditandai dengan panah solid.
	*Return*	• Mengirim kembali nilai dari satu objek ke objek lainnya. • Diberi keterangan dengan nilai yang dikembalikan dan digambarkan dengan panah putus-putus.
	*Frame*	• Menandai bagian dari keseluruhan diagram urutan.

	• Bingkai loop menampilkan sekelompok pesan dalam satu putaran. • Frame alternatif menggambarkan logika if-else atau switch, memungkinkan aktivasi kumpulan pesan yang berbeda. • Frame Opt adalah pemanggilan opsional dari sekelompok pesan.

Sumber : Systems Analysis and Design in a Changing World, Seventh Edition, 2016

2.2.5 Databases dan Sistem Pengelolaan Database (DBMS)

Database adalah kelompok data yang terintegrasi yang disimpan, dikelola, dan diatur secara terpusat. Biasanya, basis data menyimpan informasi mengenai puluhan hingga ratusan kelas atau entitas. Pengelolaan dan pengendalian basis data dilakukan melalui sistem *Database Management System* (DBMS). DBMS adalah bagian perangkat lunak dari sistem yang umumnya diperoleh dan diinstal secara terpisah dari komponen perangkat lunak sistem lainnya, seperti sistem operasi (Satzinger, Jackson, & Burd, 2016). Contoh dari sistem manajemen basis data modern meliputi Microsoft SQL ServerTM, Oracle, dan MySQL.



Gambar 2. 6 Komponen database dan DBMS dengan aktor yang berinteraksi
Sumber : Satzinger, Jackson, & Burd, 2016, p. 259

Gambar 2.5 mengilustrasikan komponen *database* pada umumnya. Biasanya ketika mengacu pada data *database*, yang ditunjukkan di sebelah kiri, dan DBMS, yang ditunjukkan di tengah. Data *database* terdiri dari dua penyimpanan informasi terkait, data domain aplikasi, yang merupakan data aktual untuk domain masalah, dan skema, yang berisi informasi deskriptif tentang data domain aplikasi. Alasan mengapa menyertakan semua komponen ini ketika membahas *database* adalah, karena data domain aplikasi tidak dapat diakses tanpa DBMS dan data skema.

Data skema atau skema data diperlukan untuk merinci struktur dan aturan yang mengatur cara mengakses domain aplikasi data. Skema sering disebut sebagai meta data atau informasi tentang data itu sendiri yang mencakup beberapa hal seperti:

- Deskripsi informasi mengenai data yang disimpan dalam penyimpanan data fisik.
- Komponen *database* yang mencakup pengorganisasian setiap item data dalam kelompok tingkat tinggi, seperti tabel.
- Hubungan antar tabel, seperti penunjuk dari objek pelanggan ke objek penjualan terkait.
- Rincian item data individual seperti tipe, panjang, lokasi, indeks item data, serta kontrol akses dan konten yang mencakup nilai yang diperbolehkan

untuk setiap item data, ketergantungan nilai di antara beberapa item data, dan daftar pengguna yang memiliki izin untuk membaca atau memperbarui item data.

DBMS memiliki empat komponen utama yaitu *Application Programing Interface* (API), antarmuka SQL atau *query* langsung, antarmuka administratif, serta serangkaian program dan sub rutin akses data yang mendasarinya. Ilustrasi pada gambar 2.5 penggunaan *Structured Query Language* (SQL)sebagai bahasa kueri untuk mengakses *database*. Pengguna, program aplikasi, dan administrator tidak langsung mengakses data domain. Mereka berinteraksi dengan antarmuka DBMS dan memberitahu jenis data apa yang mereka butuhkah, menggunakan nama-nama yang terdefinisi dalam skema.

DBMS menggunakan skema untuk memverifikasi keberadaan data yang diminta dan hak akses pengguna yang bersangkutan. Jika permintaan tersebut valid, DBMS akan mengambil informasi terkait objek data dari skema dan menggunakan informasi ini untuk mengakses data domain atas nama program atau pengguna yang membutuhkan data tersebut. *Database* dan DBMS menyediakan sejumlah kemampuan yang krusial dalam mengakses dan mengelola data, termasuk hal-hal berikut:

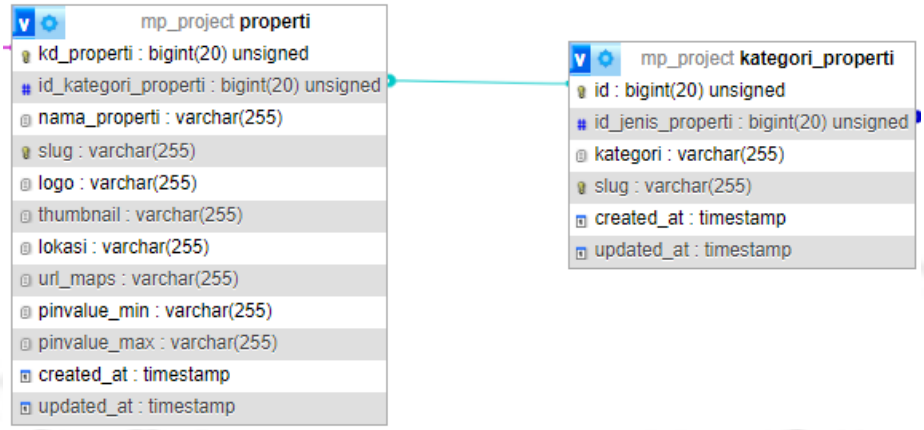
- Kemampuan untuk banyak pengguna dan program aplikasi mengakses data secara bersamaan.
- Akses ke data tanpa memerlukan penulisan program aplikasi, dapat dilakukan melalui proses langsung menggunakan SQL.
- Penyediaan kontrol akses dan aturan konten yang konsisten dan seragam.
- Integrasi data yang tersimpan di berbagai server yang tersebar di lokasi-lokasi berbeda.

2.2.6 Relational Database Management System

Relational Database Management System (RDBMS) adalah jenis DBMS yang mengatur penyimpanan data dalam struktur tabel atau hubungan. Tabel dalam *database* relasional menyerupai tabel konvensional, berbentuk struktur data dua dimensi yang terdiri dari kolom dan baris. Meski demikian, istilah yang digunakan dalam *database* relasional sedikit berbeda dari istilah yang biasa

digunakan dalam tabel konvensional (Satzinger, Jackson, & Burd, 2016). Satu baris dalam tabel relasional disebut sebagai baris, tupel, atau catatan, sementara kolom dalam tabel disebut sebagai atribut atau *field*. Setiap sel dalam tabel disebut sebagai nilai atribut, nilai *field*, atau elemen data. Seperti yang telah disebutkan, akses dan pengeditan data dalam *database* relasional dilakukan melalui bahasa *query* umum yang dikenal sebagai Structured Query Language (SQL).

Setiap tabel dalam *database* relasional harus memiliki kunci unik yang bisa disebut dengan *Primary Key*. *Primary Key* adalah suatu atribut atau kumpulan atribut yang nilainya hanya muncul satu kali dalam seluruh baris tabel. *Primary Key* secara unik mengidentifikasi setiap objek di kelas. Untuk tabel *database* relasional, *Primary Key* secara unik mengidentifikasi setiap baris dalam tabel. sering kali, terdapat beberapa kelompok atribut yang unik di setiap baris dan dapat berfungsi sebagai *Foreign Key*. Jika ada beberapa atribut unik atau kumpulan atribut yang merupakan *Foreign Key*, maka perancang *database* harus memilih salah satu kunci yang mungkin sebagai *Foreign Key*. Pada gambar 2.6 merupakan contoh relasi tabel pada DBMS MYSQL.



Gambar 2. 7 Contoh Relasi tabel DBMS MYSQL

Pada gambar 2.6 menunjukkan tabel yang saling berelasi yaitu tabel properti dan tabel kategori_properti, tabel properti memiliki atribut *Foreign Key* yaitu id_kategori_properti yang menjadi kunci rujukan ke atribut *Primary Key* tabel kategori_properti yaitu atribut id.

2.2.7 *Black Box Testing*

Pengujian *black box* dilakukan tanpa memerlukan pengetahuan tentang struktur internal sistem yang sedang diuji. Dalam pengujian ini, fokusnya adalah pada spesifikasi sistem dan tidak melibatkan pemeriksaan kode program. Pengujian *black box* dilakukan dari perspektif pengguna akhir. Insinyur pengujian yang melaksanakan pengujian *black box* hanya mengetahui *input* yang diberikan dan *output* yang diharapkan, tanpa memahami bagaimana perangkat lunak memproses *input* tersebut menjadi *output*. Pengujian *black box* nyaman untuk dilakukan karena menggunakan produk yang sudah selesai dan tidak memerlukan pemahaman tentang cara pembuatannya. Laboratorium pengujian independen dapat melaksanakan pengujian *black box* untuk memastikan sistem berfungsi dengan baik dan kompatibel (Desikan & Ramesh, 2009).

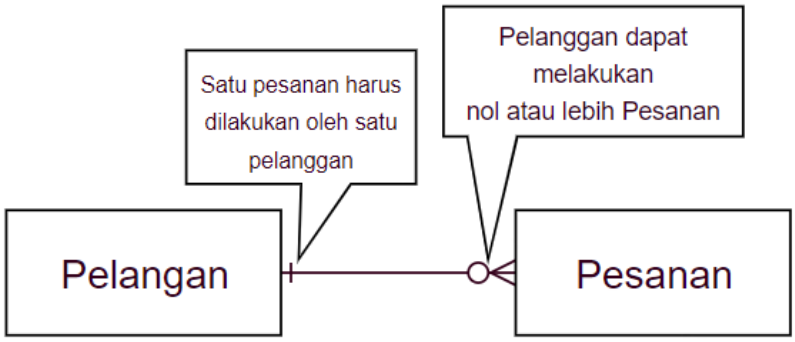
Pengujian *black box* menggunakan spesifikasi untuk membuat kasus uji secara sistematis, menghindari pengulangan, dan memastikan cakupan pengujian yang lebih menyeluruh. Dalam pengujian *black box*, yang menargetkan fungsionalitas eksternal, perlu menyusun serangkaian tes yang efektif untuk memeriksa sebanyak mungkin fungsionalitas eksternal, menemukan sebanyak mungkin cacat, dalam waktu yang singkat. Meskipun ini mungkin tampak seperti harapan yang tidak realistis, teknik-teknik yang akan dibahas dalam bagian ini akan membantu mencapai tujuan tersebut. Bagian ini akan membahas berbagai teknik untuk menghasilkan skenario pengujian yang efektif untuk pengujian *black box* (Desikan & Ramesh, 2009). Teknik-teknik tersebut meliputi :

- Pengujian berdasarkan persyaratan
- Pengujian positif dan negatif
- Analisis batas nilai
- Tabel keputusan
- Pembagian ekivalensi
- Pengujian berbasis status
- Pengujian kompatibilitas
- Pengujian dokumentasi pengguna
- Pengujian domain

2.2.8 *Entity Relationship Diagram* (ERD)

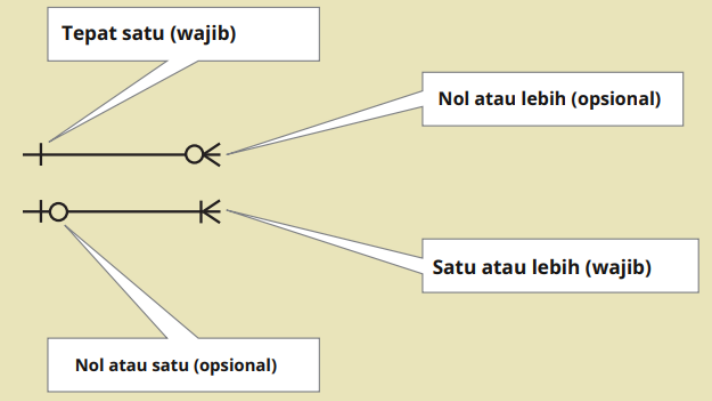
Dalam pengembangan sistem sangat menekankan pada keperluan data untuk sistem baru dan menggunakan istilah "entitas data" untuk menggambarkan hal-hal yang dibutuhkan sistem untuk menyimpan informasi. Persyaratan data ini mencakup entitas data itu sendiri, atribut-atribut yang terkait, dan keterkaitannya (disebut sebagai "asosiasi" dalam UML) antara entitas data tersebut. Model yang sering digunakan oleh analis basis data adalah *Entity Relationship Diagram* (ERD). ERD bukanlah diagram bagian dari UML, meskipun beberapa kali digunakan dan mempunyai sintaksis dan notasi yang sama dengan diagram kelas (Satzinger, Jackson, & Burd, 2016).

Pada diagram hubungan entitas, simbol kotak panjang menggambarkan entitas dari sebuah data, dan garis yang mengoneksikan persegi panjang menunjukkan relasi antar entitas data. Pada Gambar 2.7 menunjukkan contoh diagram hubungan entitas yang disederhanakan dengan dua entitas data, Pelanggan dan Pesanan. Setiap Pelanggan dapat melakukan banyak Pesanan, dan setiap Pesanan dilakukan oleh satu Pelanggan. Kardinalitasnya adalah satu-ke-banyak satu arah dan satu-ke-satu ke arah yang lain. Simbol kaki gagak pada garis di sebelah entitas data Pesanan menunjukkan banyak pesanan.

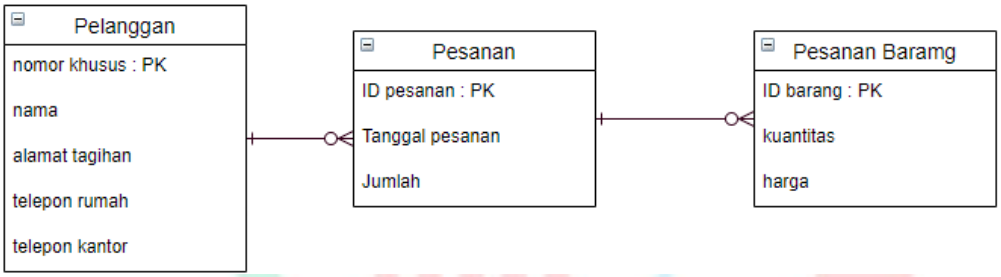


Gambar 2. 8 Contoh ERD Sederhana
Sumber : (Satzinger, Jackson, & Burd, 2016, p. 100)

Namun simbol lain pada garis hubungan juga mewakili batasan kardinalitas minimum dan maksimum. Pada gambar 2.8 merupakan penjelasan simbol hubungan ERD.



Gambar 2. 9 Simbol kardinalitas hubungan ERD
Sumber : (Satzinger, Jackson, & Burd, 2016, p. 101)

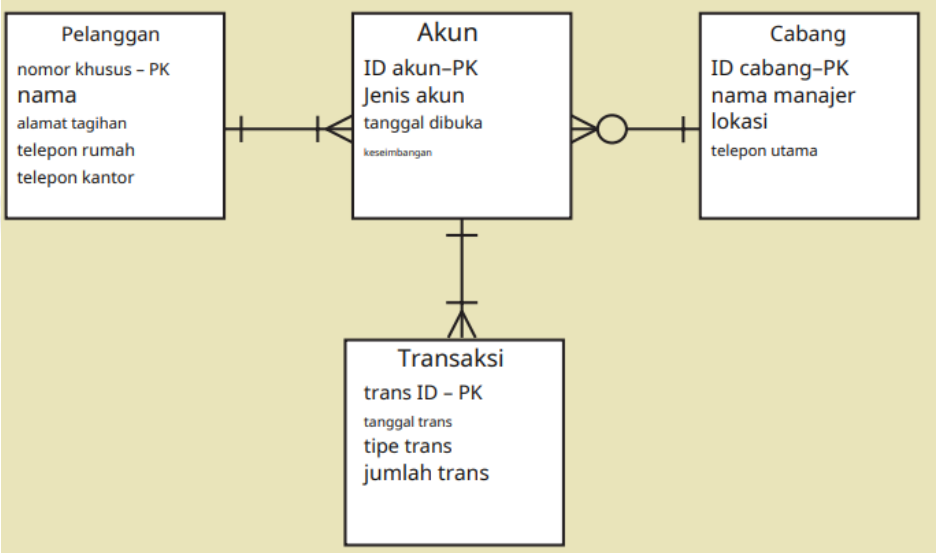


Gambar 2. 10 Contoh ERD dengan atribut
Sumber : (Satzinger, Jackson, & Burd, 2016, p. 101)

Gambar 2.9 menunjukkan model yang diperluas untuk menyertakan item pesanan (satu atau lebih item spesifik yang disertakan dalam pesanan). Setiap pesanan berisi minimal satu item dan maksimal banyak item (tidak mungkin ada pesanan jika tidak berisi minimal satu item). Misalnya, suatu pesanan mungkin mencakup kemeja, sepasang sepatu, dan ikat pinggang, dan masing-masing item ini dikaitkan dengan pesanan tersebut. Contoh ini juga memperlihatkan beberapa atribut setiap entitas data, Pelanggan memiliki nomor pelanggan, nama, alamat penagihan, dan beberapa nomor telepon. Setiap pesanan memiliki ID pesanan, tanggal pemesanan, dan sebagainya. Setiap item pesanan memiliki ID barang, kuantitas, dan harga.

Contoh lain ditunjukkan pada gambar 2.10. ERD ini diperuntukkan bagi bank yang mempunyai banyak cabang. Setiap cabang memiliki satu atau lebih akun. Setiap akun dimiliki oleh satu pelanggan dan menghasilkan satu atau lebih transaksi. Ada beberapa masalah lain yang perlu dipertimbangkan dalam contoh bank. Pertama, tidak ada entitas data bernama Bank. Hal ini dikarenakan ERD menunjukkan kebutuhan penyimpanan data pada bank. Hanya ada satu bank. Oleh

karena itu, Bank tidak perlu dimasukkan ke dalam model. Hal tersebut adalah aturan umum yang berlaku untuk ERD. Jika sistem ini diperuntukkan bagi regulator bank negara, maka Bank akan menjadi entitas data yang penting karena terdapat banyak bank yang berada di bawah yurisdiksi regulator negara.



Gambar 2. 11 Contoh ERD dengan atribut
Sumber : (Satzinger, Jackson, & Burd, 2016, p. 102)

2.3 Tinjauan Studi

Dalam penulisan penelitian ini dilakukan perbandingan dengan beberapa penelitian sebelumnya antara lain :

1. Tinjauan studi pertama diambil dari penelitian yang berjudul **"Sistem Informasi Pemasaran Dan Kredit Rumah Menggunakan Metode RAD"** yang ditulis oleh Mita Maulita, Marina Elsera, dan , Fachrul Rozi Lubis pada jurnal Journal of Information Technology Research Vol. 3, No. 1 Juli Tahun 2022. Penelitian ini difokuskan pada pengembangan sistem informasi untuk memfasilitasi pemasaran dan pemberian kredit rumah di perumahan Grand Mekar Sari Park yang memiliki lokasi strategis. Metode RAD (Rapid Application Development) dipilih sebagai pendekatan utama dalam penelitian ini karena keunggulannya dalam pendekatan yang inkremental dan cocok untuk proyek dengan batas waktu pengerjaan yang singkat. Pada bab metode penelitian, penelitian tersebut menjelaskan secara rinci langkah-langkah RAD, mulai dari perencanaan syarat-syarat hingga tahapan workshop desain

RAD dan implementasi. Hasil dari penelitian tersebut menunjukkan antarmuka sistem yang berhasil dikembangkan, termasuk halaman utama, pendaftaran, login, promosi, data booking, dan data pembelian. Penelitian ini juga menghadirkan berbagai tampilan halaman untuk 43 pengguna (pembeli), admin, dan pemilik properti.

Dengan demikian, penelitian ini berhasil menciptakan sistem yang memberikan bantuan dalam pemasaran dan promosi properti, serta memudahkan pengelolaan data bagi berbagai pihak yang terlibat, termasuk pembeli, admin, dan pemilik properti. Selain itu, tinjauan studi dalam penelitian ini juga mengaitkan temuan dan pendekatan dari penelitian-penelitian terdahulu dengan tujuan memperkaya pemahaman dan mendukung pengembangan sistem yang direncanakan. Dengan memanfaatkan hasil analisis dan temuan dari penelitian-penelitian sebelumnya, penelitian ini dapat memperoleh wawasan yang lebih dalam dan memperbaiki metode dan pendekatan yang akan digunakan. Hal ini memungkinkan peneliti untuk mengoptimalkan hasil yang akan dicapai dalam pengembangan sistem informasi yang sedang dilakukan. Sebagai kesimpulan, penelitian tersebut memberikan gambaran yang komprehensif tentang penggunaan metode RAD dalam pengembangan sistem informasi pemasaran dan kredit rumah, serta menyoroti pentingnya memanfaatkan temuan-temuan dari penelitian-penelitian terdahulu sebagai landasan untuk pengembangan yang lebih baik di masa depan.

2. Tinjauan studi kedua diambil dari penelitian yang berjudul **"Aplikasi Pemasaran Perumahan Pt. Griya Abee Makmur Ragajaya Citayam Kabupaten Bogor Menggunakan Metode Rapid Application Development (Rad) Berbasis Web"** yang ditulis oleh Muhammad Risqi Vramasatya, NM Faizah, dan Widyat Nurcahyo pada jurnal Jurnal Indonesia : Manajemen Informatika dan Komunikasi Vol 3 No 2, Juli-Desember (2022). Penelitian tersebut membahas tentang pengembangan aplikasi pemasaran perumahan berbasis web menggunakan metode Rapid Application Development (RAD). Tujuannya adalah untuk mengatasi tantangan dalam pemasaran properti, mengingat persaingan bisnis di industri pengembang

properti semakin ketat. Dalam tahap perancangan, Unified Modeling Language 44 (UML) digunakan untuk menggambarkan proses dan interaksi dalam aplikasi, termasuk proses login admin, penambahan data rumah, dan galeri gambar. Aplikasi ini dirancang untuk membantu pengembang properti mempromosikan perumahan, memberikan informasi lengkap kepada calon pembeli, serta menyediakan sarana pemesanan yang mudah. Keunggulan dari aplikasi berbasis web ini adalah memungkinkan pengguna, termasuk admin dan calon pembeli, untuk dengan mudah mengakses informasi perumahan dan melakukan pemesanan dengan efisien. Penggunaan teknologi web dan metode RAD dalam pengembangan aplikasi bertujuan untuk mengatasi tantangan dalam mempercepat proses pemasaran perumahan. Hasil dan pembahasan jurnal menguraikan berbagai diagram yang menggambarkan proses dan interaksi dalam aplikasi, seperti proses login admin, penambahan data rumah, dan galeri gambar. Aplikasi ini diharapkan dapat meningkatkan efisiensi dan efektivitas pemasaran perumahan serta memberikan layanan yang lebih baik kepada calon pembeli. Tinjauan studi ini memberikan wawasan yang berharga tentang bagaimana aplikasi berbasis web dapat mengoptimalkan proses pemasaran perumahan.

Dengan memanfaatkan metode RAD, pengembang dapat menghasilkan aplikasi dengan cepat tanpa mengorbankan kualitas. Selain itu, penggunaan UML dalam tahap perancangan memungkinkan pemodelan yang lebih baik dan pemahaman yang lebih mendalam tentang proses aplikasi. Keunggulan utama dari aplikasi ini adalah kemampuannya untuk menyediakan informasi yang lengkap dan akses yang mudah bagi calon pembeli. Dengan adanya fitur pemesanan online, proses pembelian properti menjadi lebih efisien dan praktis. Hal ini dapat membantu pengembang properti untuk meningkatkan penjualan dan memperluas pangsa pasar mereka. Selain itu, hasil penelitian ini juga memberikan pandangan yang jelas tentang bagaimana teknologi web dapat digunakan sebagai alat untuk meningkatkan efisiensi dalam industri properti. Dengan adopsi teknologi yang tepat, perusahaan properti dapat 45 mengoptimalkan operasi mereka dan memberikan layanan yang lebih baik kepada pelanggan mereka.

3. Tinjauan studi ketiga diambil dari penelitian yang berjudul **"Sistem Informasi Pemasaran Properti Berbasis Website Pada FAV Multi Sarana Bekasi"**. Terbit pada Juni 2019 Vol 3 No 2 oleh Bina Insani Ict Journal ditulis oleh Hafiz Al Farisyi, dan , Endang Retnoningsih. Penelitian ini membahas tentang pengembangan Sistem Informasi Pemasaran Properti Berbasis Website untuk FAV Multi Sarana di Bekasi. Dalam latar belakang penelitian, peneliti menyoroti peran teknologi informasi yang semakin penting dalam mendukung berbagai aspek bisnis, terutama dalam pemasaran properti. Dengan kemajuan teknologi informasi, akses terhadap informasi dan transaksi online menjadi lebih mudah dan efisien. Namun, FAV Multi Sarana, sebuah perusahaan pemasaran properti di Bekasi, menghadapi tantangan karena tidak memiliki website resmi. Sebagai gantinya, mereka bergantung pada layanan pihak ketiga yang tidak sepenuhnya memenuhi kebutuhan mereka. Konsep e-commerce, model UML, Entity Relationship Diagram (ERD), dan bahasa pemrograman PHP diangkat sebagai fondasi teknologi yang relevan dalam pembangunan sistem informasi pemasaran properti.

   Penelitian terdahulu juga menjadi landasan yang penting. Penelitian sebelumnya menyoroti masalah yang serupa dalam industri pemasaran properti dan menunjukkan bahwa sistem informasi yang terkomputerisasi dapat mengatasi kendalakendala yang muncul akibat pengelolaan manual yang kurang efisien. Metode penelitian yang digunakan adalah SDLC dengan model waterfall. Pendekatan ini dipilih karena telah terbukti efektif dalam pengembangan sistem informasi. Langkah-langkah pengembangan sistem, mulai dari analisis kebutuhan hingga pemeliharaan sistem, dijelaskan secara rinci. Tinjauan literatur tentang konsep-konsep seperti ERD, use case diagram, activity diagram, *sequence* diagram, dan *class* diagram menjadi dasar perancangan sistem yang kokoh. Hasil dari penelitian ini adalah pembangunan sistem informasi pemasaran properti berbasis website yang terintegrasi dengan baik. Melalui perancangan yang matang dan implementasi yang cermat, sistem ini berhasil mengatasi beberapa masalah yang dihadapi oleh FAV Multi Sarana, seperti kesulitan konsumen dalam mencari informasi

properti, keterbatasan informasi properti yang tersedia, dan kurangnya integrasi data properti dengan anggota.

Peneliti menyimpulkan bahwa sistem informasi yang dibangun mampu mengurangi penggunaan kertas, meningkatkan produktivitas penjualan, mempercepat proses pencarian data, dan memudahkan pembuatan laporan. Saran-saran diberikan untuk pengembangan sistem dan manajerial di masa mendatang, termasuk perlu dukungan dalam penerapan sistem, pelatihan bagi staf, pembaruan antivirus secara rutin, pembaruan aplikasi, dan keamanan database secara berkala. Secara keseluruhan, tinjauan literatur ini memberikan landasan teoritis yang kokoh bagi pengembangan sistem informasi pemasaran properti berbasis website untuk FAV Multi Sarana di Bekasi. Dengan menggabungkan konsep-konsep teknologi informasi yang relevan dan penelitian terdahulu, penelitian ini berhasil memberikan solusi yang efektif untuk meningkatkan kinerja perusahaan dalam pemasaran properti di era digital.

4. Tinjauan studi keempat diambil dari penelitian yang berjudul **"Implementasi Sistem Informasi Pemasaran Rumah Dengan Pendekatan Sistem Berorientasi Objek Pada Developer Property"** 47 yang ditulis oleh Mustar Aman, Riyanto, Suroso, Ipang Sasono, Yunianto Agung Nugroho pada Jurnal ICT : Information Communication & Technology Vol. 20, N0.1, Juli 2021, pp. 156-164. Penelitian ini fokus pada implementasi sistem informasi pemasaran rumah dengan pendekatan berorientasi sistem pada *Developer Property* di Kabupaten Tangerang. *Developer Property* tersebut menghadapi tantangan dalam memaksimalkan penjualan perumahan, karena sistem pemasarannya masih mengandalkan teknik konvensional seperti penyebaran brosur dan pengolahan data manual. Dalam menjalankan penelitian ini, penulis menggunakan metode deskriptif kualitatif. Pendekatan ini memungkinkan peneliti untuk mendapatkan pemahaman yang mendalam tentang sistem pemasaran yang ada, termasuk tantangan dan peluang yang dihadapi oleh *Developer Property*. Dengan teknik pengumpulan data melalui observasi, wawancara, dan studi pustaka, penelitian ini mampu memberikan gambaran yang komprehensif tentang kondisi pemasaran properti saat ini dan apa yang

diperlukan untuk meningkatkannya. Hasil penelitian ini adalah pengembangan perangkat lunak berbasis web yang bertujuan untuk menyederhanakan dan memodernisasi proses pemasaran properti.

Perangkat lunak ini menyajikan informasi yang detail tentang unitunit perumahan yang ditawarkan, termasuk spesifikasi bangunan, daftar harga, lokasi, stok unit, dan rincian pembayaran. Dengan demikian, calon pembeli dapat dengan mudah mengakses informasi yang mereka butuhkan untuk membuat keputusan pembelian. Analisis SWOT juga dilakukan untuk mengevaluasi kekuatan, kelemahan, peluang, dan ancaman dalam sistem pemasaran yang ada. Temuan dari analisis ini memberikan wawasan yang berharga bagi Developer Property dalam merumuskan strategi pemasaran yang lebih efektif dan berkelanjutan. Misalnya, penekanan pada kekuatan seperti interaksi langsung dengan calon pelanggan dapat menjadi dasar bagi perusahaan untuk membangun hubungan yang lebih dekat dengan konsumen. 48 Namun, penelitian ini juga mengidentifikasi sejumlah faktor penghambat dalam pemasaran properti, seperti sumber daya manusia terbatas dan kurangnya koordinasi antara tim pemasaran. Hal tersebut menunjukkan perlunya perbaikan dalam manajemen organisasi dan pengembangan karyawan untuk meningkatkan kinerja pemasaran.

Rekomendasi yang dihasilkan dari penelitian ini mencakup berbagai aspek, mulai dari penambahan karyawan sesuai bidangnya, penerapan fungsi manajemen yang lebih baik, hingga pengembangan sistem pembayaran yang lebih terstruktur. Selain itu, penggunaan teknologi informasi, seperti pembangunan sistem pemasaran berbasis web, juga ditekankan sebagai langkah penting untuk memodernisasi proses pemasaran. Dengan demikian, penelitian ini tidak hanya memberikan kontribusi dalam memperbaiki sistem pemasaran properti Developer Property di Kabupaten Tangerang, tetapi juga memberikan wawasan yang berharga bagi industri properti secara keseluruhan. Dengan memanfaatkan teknologi informasi dan menerapkan praktik manajemen yang baik, perusahaan properti dapat memaksimalkan potensi pasar mereka dan meningkatkan daya saing mereka dalam industri yang semakin kompetitif ini.

5. Tinjauan studi kelima diambil dari penelitian yang berjudul **"Rancang Bangun Sistem Informasi Pemasaran Perumahan Berbasis Web Menggunakan View 360° Pada Perumahan P T. Anindya Inti Perkasa (Aip)"** yang ditulis oleh Alfredo Hidayah Tuwllah, Arfan Yunus, pada Jurnal Jurnal Ilmu Komputer Kharisma pada 30 September 2020. Penelitian ini membahas tentang rancang bangun sistem informasi pemasaran perumahan berbasis web menggunakan teknologi *view* 360° pada perumahan PT. Anindya Inti Perkasa (AIP). Tujuan utamanya adalah menciptakan media informasi pemasaran perumahan yang lebih realistis dan dinamis, memungkinkan calon pembeli untuk melihat lokasi perumahan serta rumah dan ruangannya 49 secara menyeluruh melalui tampilan 360°, tanpa harus datang langsung ke lokasi perumahan.

   Dalam penelitian ini, penulis menggunakan metode pembuatan aplikasi dengan memanfaatkan perangkat keras khusus seperti kamera 360°, tripod, dan laptop, serta perangkat lunak seperti *code* editor dengan bahasa pemrograman PHP. Proses pengumpulan data dilakukan melalui observasi dan wawancara, sementara pengujian sistem menggunakan metode Black Box. PT. Anindya Inti Perkasa (AIP) merupakan perusahaan pengembang perumahan yang didirikan pada tahun 2015. Salah satu proyek terbesarnya adalah Griya Anindya Residence yang direncanakan akan dibangun di atas lahan seluas ±16.000m2. Proyek ini akan menawarkan 58 unit rumah dengan berbagai tipe dan merupakan salah satu lokasi strategis di Kabupaten Tangerang. Namun, meskipun lokasinya menjanjikan, perusahaan menghadapi kendala dalam pemasaran karena mengandalkan sistem konvensional seperti direct selling dan distribusi brosur. Dalam upaya meningkatkan efektivitas pemasaran, penelitian ini mengusulkan pengembangan aplikasi web berbasis view 360°. Melalui aplikasi ini, calon pembeli dapat mengakses informasi tentang perumahan, melihat panorama 360° dari lokasi dan rumah-rumah yang ditawarkan, serta mendapatkan detail spesifikasi teknis dengan lebih rinci. Analisis teoritis dalam penelitian ini mencakup konsep rancang bangun sistem, definisi perumahan, dan konsep fotografi virtual reality. Metode pengembangan sistem menggunakan model

Waterfall yang menggambarkan alur kerja tahapan pembuatan aplikasi secara linier. Arsitektur sistem menggunakan bahasa pemrograman PHP dengan *database* MySQL, yang diakses oleh pengguna melalui web server seperti XAMPP. Selain itu, *flowchart* sistem juga digambarkan untuk memvisualisasikan alur kerja aplikasi web. Pengujian sistem dilakukan dengan metode Black Box, yang berfokus pada fungsionalitas aplikasi tanpa memperhatikan detail implementasi.

Hasil pengujian menunjukkan bahwa aplikasi dapat berjalan sesuai yang diharapkan, 50 mudah dipahami oleh pengguna, dan memberikan manfaat yang baik dalam memberikan informasi kepada calon pembeli. Kesimpulan dari penelitian ini adalah bahwa sistem informasi pemasaran perumahan berbasis web menggunakan *view* 360° pada PT. Anindya Inti Perkasa (AIP) berhasil memenuhi tujuan penelitian. Dengan adanya aplikasi ini, pemasaran perumahan menjadi lebih efektif dan efisien, memungkinkan calon pembeli untuk menjelajahi properti tanpa harus datang ke lokasi fisiknya. Hal ini diharapkan dapat membantu perusahaan dalam meningkatkan penjualan dan daya saingnya dalam industri properti.