**BAB 2**

**LANDASAN TEORI**

1. **Keunggulan Sistem**

Ditinjau dari segi kebutuhan pengguna terhadap aplikasi, sistem yang kami buat ini sederhana dan tidak rumit dalam penggunaanya.Fitur yang kami sajikan tidak membuat pengguna bingung dalam menggunakan aplikasi,dikarenakan tidak terlalu banyak fitur yang disediakan namun sudah mumpuni sebagai sistem pengolah data .Sehingga para pengguna menggunakan aplikasi ini lebih efisien, hemat waktu, dan tidak ruwet.Tampilan dan Bahasa yang kami sajikan lebih mudah dipahami oleh masyarakat pada umumnya, sehingga bisa di pakai oleh siapa pun .

Selain itu, setelah kami observasi dari aplikasi yang sudah ada sebelumnya, aplikasi kami ini menyajikan satu fitur yang tidak ada pada aplikasi lain, yakni fitur pembayaran denda buku yang hilang. Pada aplikasi ini, Kami buat daftar status peminjaman buku. Dalam daftar tersebut terdapat bagian status pengembalian buku yang isinya adalah status Kembali, belum Kembali, dan hilang. Ketika ada peminjam yang berstatus menghilangkan buku, maka kami sajikan fitur yang menampilkan besaran denda yang harus di bayar sesuai dengan jumlah atau pun jenis bukunya. Desain yang kami gunakan pun lebih menarik dari desain aplikasi lain dikarenakan kami sesuaikan dengan fitur yang disediakan dan kebutuhan pada system.

1. **Metodologi Pengembangan Sistem**

SDLC (*Systems Development Life Cycle*, Siklus Hidup Pengembangan Sistem) atau *Systems Life Cycle* (Siklus Hidup Sistem), dalam rekayasa sistem dan rekayasa perangkat lunak, adalah proses pembuatan dan pengubahan sistem serta model dan metodologi yang digunakan untuk mengembangkan sistem-sistem tersebut. Konsep ini umumnya merujuk pada sistem komputer atau informasi. SDLC juga merupakan pola yang diambil untuk mengembangkan sistem perangkat lunak, yang terdiri dari tahap-tahap: rencana (planning), analisis (analysis), desain (design), implementasi (implementation), uji coba (testing) dan pengelolaan (maintenance). Dalam rekayasa perangkat lunak, konsep SDLC mendasari berbagai jenis metodologi pengembangan perangkat lunak. Metodologi - metodologi ini membentuk suatu kerangka kerja untuk perencanaan dan pengendalian pembuatan sistem informasi, yaitu proses pengembangan perangkat lunak. Terdapat 3 jenis metode siklus hidup sistem yang paling banyak digunakan, yakni: siklus hidup sistem tradisional (traditional system life cycle), siklus hidup menggunakan prototyping (life cycle using prototyping), dan siklus hidup sistem orientasi objek (object-oriented system life cycle). SDLC juga merupakan metodologi umum dalam pengembangan sistem yang menandai kemajuan usaha analisis dan desain. fase - fase yang terdapat di dalam SDLC meliputi :

a. Perencanaan Sistem (Systems Planning)

b. Analisis Sistem (Systems Analysis)

c. Perancangan Sistem (Systems Design)

d. Implementasi Sistem (Systems Implementation)

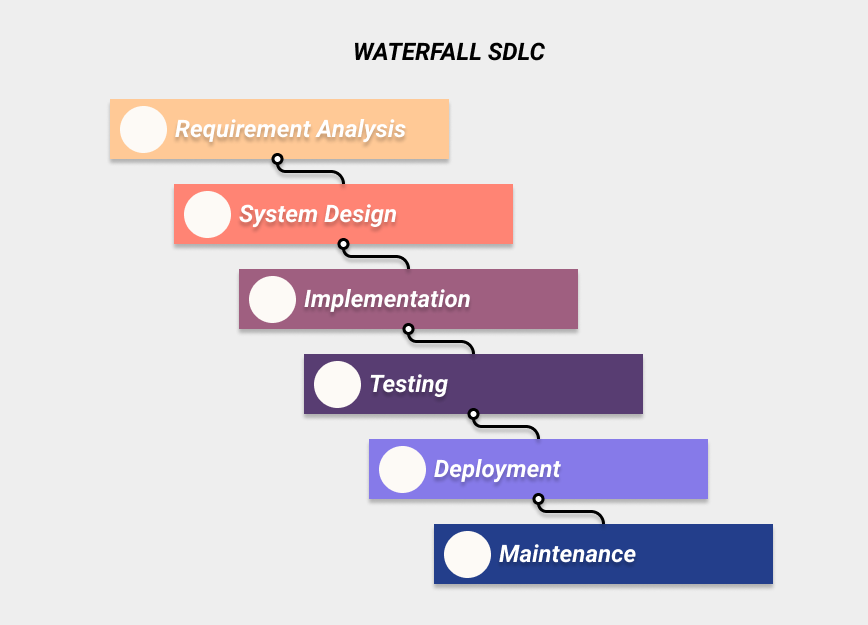
e. Pemeliharaan Sistem (Systems Maintenance)

Konsep SDLC mendasari model pengembangan perangkat lunak lainnya. Model pengembangan perangkat lunak tersebut antara lain waterfall, prototype, iterative, spiral, rapid application development (RAD) dan lainnya. Dalam penelitian ini hanya akan dibahas tiga model yaitu Waterfall, Prototyping, dan RAD. Dalam penelitian ini, penulis menggunakan data dari review beberapa jurnal dan di komparasi, dengan kata lain penulis menggunakan metode penelitian metode deskriptif. Metode deskriptif merupakan suatu metode penelitian dalam meneliti status dari sekelompok manusia, suatu objek, suatu sistem pemikiran, suatu set kondisi, ataupun suatu kelas peristiwa pada masa saat ini. adapun tujuan dari penelitian deskriptif ini yaitu untuk membuat gambaran, deskripsi atau lukisan secara sistematis, faktual dan akurat mengenai fakta, sifat serta hubungan antar fenomena yang sedang diselidiki.

* Waterfall

Model waterfall adalah model pengembangan perangkat lunak yang paling sering digunakan. Model pengembangan ini bersifat linear dari tahap awal pengembangan sistem yaitu tahap perencanaan sampai tahap akhir pengembangan sistem yaitu tahap pemeliharaan. Tahapan berikutnya tidak akan dilaksanakan sebelum tahapan sebelumnya selesai dilaksanakan dan tidak bisa kembali atau mengulang ke tahap sebelumnya. Model SDLC air terjun (waterfall) sering disebut juga model sekuensial linier (sequential linear) atau alur hidup klasik (classic life cycle). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau berurutan dimulai dari analisis, desain, pengkodean, pengujian, dan tahap pendukung (support)”.

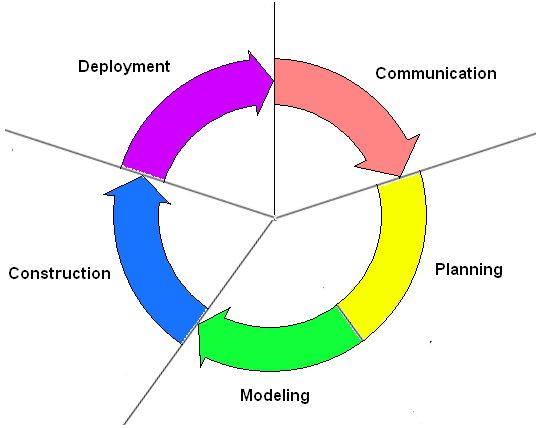
Berikut adalah gambar model air terjun:



Gambar 1. Metode Waterfall

* Prototype

Prototype Model prototyping merupakan suatu teknik untuk mengumpulkan informasi tertentu mengenai kebutuhan - kebutuhan informasi pengguna secara cepat. Berfokus pada penyajian dari aspek-aspek perangkat lunak tersebut yang akan nampak bagi pelanggan atau pemakai. Prototipe tersebut akan dievaluasi oleh pelanggan/pemakai dan dipakai untuk menyaring kebutuhan pengembangan perangkat lunak. Prototype didefinisikan sebagai alat yang memberikan ide bagi pembuat maupun pemakai potensial tentang cara sistem berfungsi dalam bentuk lengkapnya, dan proses untuk menghasilkan sebuah prototype disebut prototyping. Bentuk dari prototype pada gambar 2 merupakan versi awal dari tahapan sebuah sistem software yang digunakan dalam mempresentasikan gambaran dari ide, eksperimen dari sebuah rancangan, mencari sebanyak mungkin masalah yang ada serta penyelesaian terhadap masalah tersebut. Sistem dengan model prototype mengizinkan pengguna agar mengetahui seperti apa tahapan sistem dibuat sehingga sistem mampu beroperasi dengan baik. Metode prototype digunakan dimaksudkan untuk mendapatkan representasi dari pemodelan aplikasi yang akan dibuat. Rancangan aplikasi awal mulanya berbentuk mockup selanjutnya akan dievaluasi oleh pengguna. Setelah mockup dievaluasi pengguna tahap selanjutnya mockup menjadi bahan rujukan bagi pengembang software untuk merancang aplikasi.



Gambar 2. Model Prototype

* RAD

Rapid Application Development (RAD) adalah sebuah model Proses perkembangan perangkat lunak sekuensial linier yang menekankan siklus perkembangan yang sangat pendek (kira- kira 60 sampai 90 hari). Model RAD ini merupakan sebuah adaptasi “kecepatan tinggi” dari model sekuensial linier dimana perkembangan cepat dicapai dengan menggunakan pendekatan konstruksi berbasis komponen. RAD adalah suatu pendekatan berorientasi objek terhadap pengembangan sistem yang mencakup suatu metode pengembangan serta perangkat - perangkat lunak. RAD bertujuan mempersingkat waktu yang biasanya diperlukan dalam siklus hidup pengembangan sistem tradisional antara perancangan dan penerapan suatu sistem informasi. Pada akhirnya, RAD sama-sama berusaha memenuhi syarat-syarat bisnis yang berubah secara cepat. RAD menggunakan metode iteratif (berulang) dalam mengembangkan sistem dimana working model (model bekerja) sistem dikonstruksikan di awal tahap pengembangan dengan tujuan menetapkan kebutuhan (requirement) pengguna dan selanjutnya disingkirkan. Dalam pengembangan sistem informasi normal, memerlukan waktu minimal 180 hari, namun dengan menggunakan metode RAD, sistem dapat diselesaikan dalam waktu 30-90 hari [10]. Model RAD memiliki 3 tahapan sebagai berikut:

1. Rencana Kebutuhan (Requirement Planning)

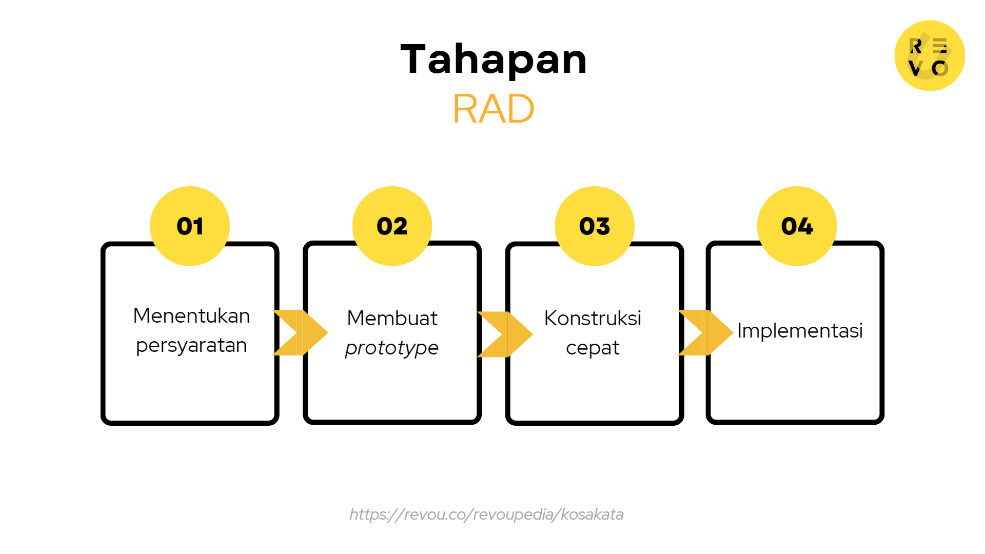
User dan analyst melakukan pertemuan untuk mengidentifikasi tujuan dari sistem dan kebutuhan informasi untuk mencapai tujuan. Pada tahap ini merupakan hal terpenting yaitu adanya keterlibatan dari kedua belah pihak.

1. Proses Desain Sistem (Design System)

Pada tahap ini keaktifan user yang terlibat menentukan untuk mencapai tujuan karena pada proses ini melakukan proses desain dan melakukan perbaikan - perbaikan apabila masih terdapat ketidaksesuaian desain antara user dan analyst. Seorang user dapat langsung memberikan komentar apabila terdapat ketidaksesuaian pada desain, merancang sistem dengan mengacu pada dokumentasi kebutuhan user yang dibuat pada tahap sebelumnya. Keluaran dari tahapan ini adalah Bangkit Indonesia, Vol. X, No. 01, Bulan Maret 2021 ISSN : 233740558 spesifikasi software yang meliputi organisasi sistem secara umum, struktur data dan yang lain.

1. Implementasi (Implementation)

Tahapan ini adalah tahapan programmer yang mengembangkan desain suatu program yang telah disetujui oleh user dan analyst. Sebelum diaplikasikan pada suatu organisasi terlebih dahulu dilakukan proses pengujian terhadap program tersebut apakah ada kesalahan atau tidak. Pada tahap ini user biasa memberikan tanggapan akan sistem yang sudah dibuat serta mendapat persetujuan mengenai sistem tersebut.



Gambar 3. Metode Rapid Aplication Development (RAD)

1. **Metodologi Pemodelan Sistem**

UML (Unified Modeling Language) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah system pengembangan software berbasis OO (Object Oriented). (Sri Dharwiyanti, Pengantar Unified Modeling Language, 2003). UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas - kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem software.

* Bangunan Dasar UML

Untuk memahami UML, kita membutuhkan model konseptual dari bahasa ini.. Metodologi UML menggunakan 3 bangunan dasar untuk mendeskripsikan sistem/perangkat lunak yang dikembangkan, yaitu:

1. Benda (Things)

2. Relasi (Relationship)

3. Diagrams

Setiap bangunan dasar dapat diterapkan sepanjang tahap pengembangan sistem. Ketiga jenis bengunan dapat digunakan secara saling melengkapi satu sama lain (Contoh lebih mendalam akan dijelaskan pada bagian-bagian selanjutnya). Secara umum UML diterapkan dalam pengembangan sistem/perangkat lunak berorientasi objek sebab metodologi UML ini umumnya memiliki keunggulan sebagai berikut : (www.kakasblog.blogspot.com, object oriented with UML,2004)

1. Uniformity

Dengan metodologi UML, para pengembang cukup menggunakan satu metodologi dari tahap analisis hingga perancangan. Hal ini tidak bias dilakukan dalam metodologi pengembangan terstruktur. Dengan perkembangan masa kini ke arah aplikasi GUI (Graphical User Interface), UML juga memngkinkan kita merancang komponen antarmuka pengguna (User Interface) secara integrasi bersama dengan perancangan perangkat lunak sekaligus dengan perancangan basis data.

2. Understandability

Dengan metodologi ini kode yang dihasilkan dapat diorganisasi ke dalam kelas - kelas yang berhubungan dengan masalah sesungguhnya sehingga lebih mudah dipahami siapapun juga.

3. Stability

Kode program yang dihasilkan relatif stabil sepanjang waktu sebab sangat mendekati permasalahan sesungguhnya di lapangan.

4. Reusability

Dengan metodologi berorientasi objek, dimungkinkan pengguna ulang kode, sehingga pada gilirannya akan sangat mempercepat waktu pengembangan perangakat lunak.

* Konsep Diagram dalam Memodelkan Sistem

Setiap sistem yang kompleks seharusnya bias dipandang dari sudut yang berbeda-beda sehingga kita bisa mendapatkan pemahaman secara menyeluruh. Untuk upaya tersebut UML menyediakan 8 jenis diagram yang dapat dikelompokkan berdasarkan sifat-sifatnya (statis dan dinamis). Ke-8 jenis diagram dalam UML tersebut adalah:

1. Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem.

2. Class Diagram

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain - lain.

3. Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada system sebagai akibat dari stimuli yang diterima. Pada umumnya statechart diagram menggambarkan class tertentu (satu class dapat memiliki lebih dari satu statechart diagram).

4. Activity Diagram

Activity diagrams bersifat dinamis. Diagram ini adalah tipe khusus dari diagram state yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya disebuah sistem. Activity diagram merupakan state diagram khusus, dimana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (internal processing). Oleh karena itu activity diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur - jalur aktivitas dari level atas secara umum.

5. Sequence Diagram

Sequence diagram bersifat dinamis. Diagram urutan ini adalah siagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu. Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah - langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang mentrigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

6. Collaboration Diagram

Collaboration Diagram bersifat dinamis. Diagram kolaborasi adalah diagram interaksi yang menekankan organisasi struktural dari objek - objek yang menerima serta mengirim pesan.

7. Component Diagram

Component Diagram bersifat statis. Diagram ini memperlihatkan organisasi serta kebergantungan pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas-kelas, antarmuka, serta kolaborasi - kolaborasi.

8. Deployment Diagram

Deployment Diagram bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan. Diagram ini memuat simpul - simpul (node) beserta komponen-komponen yang ada di dalamn Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada mesin (distributed computing). Dibuatnya berbagai jenis diagram diatas karena :

1) Setiap sistem yang kompleks selalu paling baik jika didekati melalui himpunan berbagai sudut pandang yang kecil yang satu sama lain hamper saling bebas (independent). Sudut pandang tunggal senantiasa tidak mencukupi untuk melihat isi item yang besar dan kompleks.

2) Diagram yang berbeda-beda tersebut dapat menyatakan tingkatan yang berbeda - beda dalam proses rekayasa.

3) Diagram - diagram tersebut dibuat agar model yang dibuat semakin mendekati realitas. Diagram - diagram ini ditambah dengan kemampuan dokumentasi merupakan artifacts utama UML. Data-flow diagram dan tipe diagram lain yang tidak terdapat dalam UML tidak termasuk dalam paradigma object-oriented, Activity diagram dan collaboration diagram yang terdapat dalam UML menggantikan data-flow diagram. Activity diagram juga sangat bermanfaat untuk membuat workflow.

* Tujuan Unified Modelling Language (UML)

Tujuan UML diantaranya adalah :

1) Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.

2) Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.

3) Menyatukan praktek - praktek terbaik yang terdapat dalam pemodelan.

* Tool - Tool yang mendukung Unified Modelling Language (UML)

Saat ini banyak sekali tool pendesainan yang mendukung UML, baik itu tool komersial maupun open source. Beberapa diantaranya adalah:

1) Rational Rose (www.rational.com)

2) Together (www.togethersoft.com)

3) Object Domain (www.objectdomain.com)

4) Jvision (www.object-insight.com)

5) Objecteering (www.objecteering.com)

6) MagicDraw (www.nomagic.com/magicdrawuml)

7) Visual Object Modeller (www.visualobject.com)

8) Visual Paradigm For UML (www.visualpardigm.com)

* 1. **Studi Kelayakan**
     + Kelayakan Teknis

Analisis Teknologi : Evaluasi infrastruktur teknologi yang dibutuhkan untuk menjalankan aplikasi perpustakaan berbasis web, termasuk kebutuhan server, database, dan koneksi internet.

Ketersediaan Sumber Daya : Sumber daya manusia yang memiliki keterampilan teknis yang diperlukan untuk mengembangkan, mengelola, dan memelihara aplikasi perpustakaan.

Kesesuaian dengan Standar : Sesuai dengan standar keamanan dalam administrasi peminjaman dan pengembalian buku agar lebih terstruktur

* Kelayakan Ekonomi

Analisis Biaya dan Manfaat : Menilai biaya pengembangan, implementasi, dan pemeliharaan aplikasi perpustakaan berbasis web serta manfaat yang diharapkan dari investasi tersebut.

Perhitungan ROI : Menghitung tingkat pengembalian investasi (ROI) untuk memastikan bahwa implementasi aplikasi perpustakaan akan memberikan nilai ekonomis yang cukup.

Pemodelan Keuangan : Membuat proyeksi keuangan untuk menilai kelayakan jangka panjang dari proyek ini, termasuk estimasi pendapatan dan pengeluaran yang terkait.

* Kelayakan Operasional:

Analisis Proses Bisnis : Memahami proses bisnis perpustakaan yang ada dan menentukan sejauh mana aplikasi perpustakaan berbasis web dapat meningkatkan efisiensi operasional.

Ketersediaan Sumber Daya : Menilai ketersediaan sumber daya manusia, waktu, dan infrastruktur yang diperlukan untuk mengoperasikan dan memelihara aplikasi perpustakaan dengan efektif.

Risiko Operasional : Mengidentifikasi risiko-risiko operasional yang mungkin muncul selama pengembangan dan operasional aplikasi perpustakaan dan merencanakan strategi mitigasi yang sesuai.

* Kelayakan Jadwal

Penjadwalan Proyek : Menilai ketersediaan sumber daya dan merencanakan jadwal pengembangan, implementasi, dan peluncuran aplikasi perpustakaan untuk memastikan bahwa proyek dapat diselesaikan sesuai dengan waktu yang diharapkan.

Ketergantungan Eksternal : Mengidentifikasi ketergantungan eksternal seperti pihak ketiga atau penyedia layanan yang diperlukan untuk menjalankan proyek dan mempertimbangkan faktor-faktor ini dalam penjadwalan.

Berdasarkan studi kelayakan ini, aplikasi perpustakaan berbasis web dinilai layak untuk diimplementasikan. Kelayakan teknis, ekonomi, operasional, dan jadwal telah dievaluasi dengan baik, dan proyek ini memiliki potensi untuk memberikan manfaat yang signifikan bagi perpustakaan dan pengguna. Dengan strategi implementasi dan manajemen risiko yang tepat, proyek ini memiliki peluang untuk sukses dalam jangka panjang.