BAB II

TINJAUAN PUSTAKA

2.1 Sistem

Sistem adalah sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisir, saling berinteraksi, saling tergantung satu sama lain, dan terpadu. Model umum sebuah sistem adalah *input*, proses, dan *output*. Hal ini merupakan konsep sebuah sistem yang sangat sederhana sebab sebuah sistem dapat mempunyai beberapa masukan dan keluaran. Selain itu, sebuah sistem mempunyai karakteristik atau sifat-sifat tertentu yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. (Sutabri,2012).

Adapun karakteristik yang dimaksud adalah sebagai berikut:

1. Komponen Sistem *(Component)*

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat dari sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar atau sering disebut “supra sistem”.
2. Batasan Sistem *(Boundary)*

   Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain atau sistem dengan lingkung luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisahkan.
3. Lingkungan Luar Sistem *(Envirotment)*

   Bentuk apapun yang ada diluar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut lingkungan luar sistem. Lingkungan luar sistem ini [illegible] menguntungkan dan dapat juga

bersifat merugikan sistem tersebut. Dengan demikian, lingkungan luar tersebut harus tetap dijaga dan dipelihara. Lingkungan luar yang merugikan harus dikendalikan. Kalau tidak, maka akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Sistem *(Interface)*

   Media yang menghubungkan sistem dengan subsistem lain disebut penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lain. Bentuk keluaran dari subsistem akan menjadi masukan untuk subsistem lain melalui penghubung tersebut. Dengan demikian, dapat terjadi suatu integrasi sistem yang membentuk satu kesatuan.

5. Masukan Sistem *(Input)*

   Energi yang dimasukkan kedalam sistem tersebut masukkan sistem, yang dapat berupa pemeliharaan *(maintenance input)* dan sinyal *(signal input).* Contoh, di dalam suatu unit sistem komputer, "Program" adalah *maintenance* input yang digunakan untuk mengoperasikan komputernya dan "Data" adalah signal *input* untuk diolah menjadi informasi.

6. Keluaran Sistem*(Output)*

   Hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain seperti sistem informasi. Keluaran yang dihasilkan adalah informasi. Informasi ini dapat digunakan sebagai masukan untuk pengambilan keputusan atau hal-hal lain yang menjadi *input* bagi subsistem lain.

7. Pengolahan Sistem *(Process)*

   Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran, contohnya adalah sistem akuntansi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem *(Objective)*

   Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat *deterministic*. Kalau suatu sistem tidak memiliki sasaran maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran dan tujuan yang telah direncanakan.

2.2 Informasi

Informasi pada dasarnya adalah himpunan data yang telah diolah menjadi sesuatu yang memiliki arti dan kegunaan lebih luas".Informasia dalah data yang diolah menjadi bentuk lebih berguna dan lebih berarti bagi yang menerimanya (Hartono, 2013).

2.3 Sistem Informasi

Sistem informasi adalah seperangkat komponen yang saling berhubungan, yang bekerja untuk mengumpulkan dan menyimpan data serta mengolahnya untuk menjadi informasi yang digunakan" (Hartono, 2013).

2.4 Pariwisata

Pariwisata adalah salah satu industri baru yang mampu mempercepat pertumbuhan ekonomi dan penyediaan lapangan kerja, peningkatan penghasilan, standar hidup serta menstimulasi sektor-sektor produktif lainnya. Selajutnya sebagai sektor yang kompleks, ia juga memiliki industri-industri klasik seperti industri kerajinan tangan dan cinderamata. Penginapan dan transportasi secara ekonomis juga dipandang sebagai industri. (Pendit, 2003)

Kepariwisataan menggambarkan beberapa bentuk perjalanan untuk memperoleh berbagai tujuan dan memuaskan berbagai macam keinginan. Pariwisata sebagai suatu gejala yang terwujud dalam beberapa bentuk, antara lain sebagai berikut :

1. Menurut jumlah orang yang berpergian :
    a. Pariwisata individu, yaitu hanya seorang atau satu keluarga yang berpergian
    b. Pariwisata rombongan, yaitu sekelompok orang yang biasanya terikat oleh hubungan-hubungan tertentu kemudian melakukan perjalanan bersama-sama
2. Menurut maksud berpergian :

    Pariwisata Rekreasi atau Pariwisata Santai, yaitu pariwisata dengan maksud kepergian untuk memulihkan kemampuan fisik dan mental setiap peserta wisata dan memberikan kesempatan rileks bagi mereka dari kebosanan dan keletihan kerja selama di tempat rekreasi.

2.5 Kabupaten Pringewu

Kabupaten pringsewu merupakan salah satu kabupaten di Provinsi Lampung hasil pemekaran dari Kabupaten Tanggamus, dan dibentuk berdasarkan undang-undang Nomor 48 Tahun 2008 Tanggal 26 November 2008 dan diresmikan pada tanggal 3 April 2009 oleh Mentri Dalam Negri. Luas wilayah yang dimiliki sekitar 625 km^2 atau 62.500 Ha. (*http://bpbd.pringsewukab.go.id/profil-kabupaten-pringsewu*)

Selanjutnya yang ditunjuk sebagai Pj. Bupati Pringsewu untuk yang pertama kali adalah Ir. H. Masdullhaq, yang memimpin pemerintahan di Kabupaten Pringsewu yang kemudian digantikan oleh H. Helmi Machmud, dan digantikan kembali oleh Sudarno Edi, dan Bupati Pringsewu saat ini dijabat oleh Hi. Sujadi Saddat. (*https://pringsewukab.bps.go.id*)

Kabupaten Pringsewu terdiri dari 9 (sembilan) Wilayah Kecamatan, kecamatan-kecamatan di Kabupaten Pringsewu :

1. Kecamatan Pardasuka
2. Kecamatan Ambarawa
3. Kecamatan Pagelaran
4. Kecamatan Pagelaran Utara

5. Kecamatan Pringsewu
6. Kecamatan Gading Rejo
7. Kecamatan Sukoharjo
8. Kecamatan Banyumas
9. Kecamatan Adiluwih.

Gambar 2.1 Peta Kabupaten Pringsewu

(sumber : https://id.wikipedia.org/wiki/Berkas:Peta_Kabupaten_Pringsewu.jpg)

Wilayah Kabupaten Pringsewu mulai tahun 2013 terdiri dari 5 Kelurahan serta 126 Pekon (desa). Pada Tahun 2013, jumlah kecamatan di Kabupaten Pringsewu menjadi sembilan kecamatan.

Secara administratif Kabupaten Pringsewu berbatasan dengan 3 (tiga) wilayah kabupaten sebagai berikut :

- Sebelah Utara berbatasan dengan Kecamatan Sendang Agung dan Kecamatan Kalirejo, Kabupaten Lampung Tengah.

- Sebelah Timur berbatasan Kecamatan Negeri Katon, Kecamatan Gedongtataan, Kecamatan Waylima dan Kecamatan Kedondong, Kabupaten Pesawaran.
- Sebelah Selatan berbatasan dengan Kecamatan Bulok dan Kecamatan Cukuh Balak, Kabupaten Tanggamus.
- Sebelah Barat berbatasan dengan Kecamatan Pugung dan Kecamatan Air Naningan, Kabupaten Tanggamus.

2.6 *Android*

Android adalah aplikasi sistem operasi untuk telepon seluler yang berbasis *Linux*. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak. (Safaat, 2012)

1. Kelebihan Android
   a. *Switching* dan multitasking yang lebih baik Android sangat mendukung multitasking aplikasi, kini hal tersebut kembali ditingkatkan. Dalam *Honeycomb* penggguna dapat dengan mudah berpindah aplikasi hanya dengan menyentuh sebuah icon pada *system bar*.
   b. Kapasitas yang lebih baik untuk beragam widget Kapabilitas terhadap beragam *widget* dijanjikan bakal makin memanjakan para penggunanya. Contohnya *widget* untuk email *Gmail* yang dipamerkan *Google*, pengguna tidak perlu membuka aplikasi *Gmail* untuk melihat isi di dalamnya.
   c. Peningkatan kemampuan copy-paste Beberapa seri *Android* terdahulu memang sudah bisa melakukan copypaste, namun beberapa pengguna masalah pemilihan teks yang agak sulit. Kini hal tersebut coba diselesaikan, selain *copypaste Google* juga menambah share it pada teks yang diseleksi.
   d. *Browser Crome* Lebih Cepat Ada satu fitur yang hilang dalam browser Chrome yang diletakkan pada Android terdahulu, kemampuan Tab. Chrome yang ada di Honeycomb kini dapat melakukan hal tersebut. Selain

itu pengguna juga bisa mensinkronisasi antara *browser* di ponsel dengan *Crome* yang ada di komputer.

e. Notifikasi yang Mudah Terdengar. Dengan layar yang lebih besar, otomatis membuat Google lebih leluasa menempatkan notifikasi pada layar.

f. Peningkatan *Drag* and *Drop* serta *Multitouch* Ukuran layar yang lebih besar, menuntut Google untuk meningkatkan kemampuan multitouch di dalam *Android*, tak terkecuali fitur *drag* and drop. Pada demo yang ditayangkan, pengguna bisa melakukan drag and drop untuk memindahkan *email* di dalam aplikasi *Gmail*.

2. Kekurangan Android

a. Koneksi Internet yang terus menerus. Kebanyakan ponsel Android memerlukan koneksi internet yang simultan atau terus menerus aktif, itu artinya anda harus siap berlangganan paket GPRS yang sesuai dengan kebutuhan dan batre yang boros karena GPRS yang terus menyala.

b. Aplikasi di Ponsel Android memang bisa didapatkan dengan mudah dan gratis, namun konsekuensinya di setiap Aplikasi tersebut, akan selalu ada Iklan yang terpampang.

2.7 UML *(Unifield Modeling Language)*

UML merupakan kumpulan diagram-diagram yang memiliki standar untuk pembangunan perangkat lunak berbasis objek. UML memiliki banyak diagram, *Use Case Diagram, Class Diagram, Sequence Diagram, dan Activity Diagram.* UML sendiri sebetulnya memiliki banyak sekali diagram selain empat yang disebutkan diatas, akan tetapi sebagaian besar memang jarang dibuat oleh pengembang aplikasi. Keempat diagram di atas merupakan diagram inti dari UML. Bagaimanapun aplikasi perangkat lunak yang dibuat, bila dibangun dengan berorientasi objek, keempat diagram tersebut harus ada. Penggunaan diagram lainnya dalam UML ditunjukan untuk mendapatkan gambaran arsitektur sistem informasi dengan sudut pandang yang berbeda, hal ini terpaut pada faktor kebutuhan.

Penting untuk dipahami, sewaktu spesialis informasi memiliki membuat permodelan arsitektur dengan pendekatan terstruktur (misalnya: membuat Diagram Konteks dan Diagram Alir data), maka permodelan arsitektur berbasiskan objek (misalnya membuat diagram-diagram UML semisal *Diagram Use Case, Class Diagram*) tidak lagi digunakan, demikian pula sebaiknya (Sulianta, 2017).

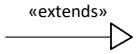
2.7.1 Use Case Diagram

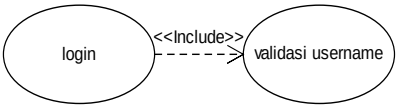
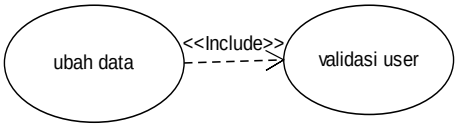
Diagram Use Case merupakan diagram yang harus dibuat pertama kali saat pemodelan perangkat lunak berorientasi objek dilakukan. *Diagram Use Case* akan menggambarkan apa yang dikerjakan oleh aktor. Yang disebut aktor disini adalah pengguna aplikasi, sama seperti pembangunan perangkat lunak terstruktur saat membuat DFD, untuk menggambarkan *Diagram Use Case* mengacu pada proses sebelumnya, yaitu analisis kebutuhan pada RPL. (Sulianta, 2017).

a. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi, walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
b. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Tabel 2.1 Simbol *Use Case Diagram*

Keterangan	Simbol	Deskripsi
Use Case		Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja diawal-awal frase nama *use case*
Aktor		Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar itu sendiri. Jadi walaupun simbol dari aktor adalah

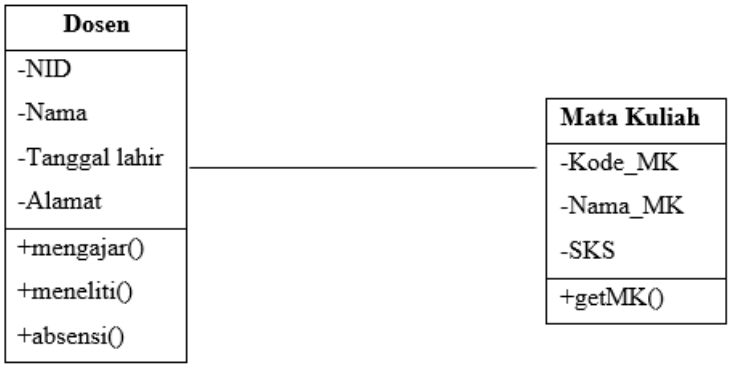
		gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda diawal frase nama aktor.
Asosiasi		Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
Ekstensi		Relasi use case tambahan ke sebuah *use case*, dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan use case yang ditambahkan, misal  Arah panah mengarah pada *use case* yang ditambahkan.
Generalisasi		Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya : 

		Arah panah mengarah pada *use case* yang menjadi generalisasinya (umum).
Menggunakan/*include/uses*	<<Include>> «uses»	Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* : a. Include berarti use case yang ditambahkan akan selalu dipanggil saat use case tambahan dijalankan, misal pada kasus berikut :  b. Include berarti use case yang tambahan akan selalu melakukan pengecekan apakah use case yang ditambahkan telah dijalankan sebelum use case tambahan dijalankan, misal pada kasus berikut :  Ke dua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi,

2.7.2 Class Diagram

Diagram kelas dibuat setelah Diagram *Use Case* dibuat terlebih dahulu. Pada pembuatan diagram ini harus menjelaskan hubungan apa saja yang terjadi antara suatu objek dengan objek lainnya sehingga terbentuklah suatu sistem aplikasi.

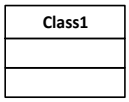
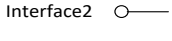
Pembuatan Diagram Kelas dibagi menjadi dua bagian, yaitu kelas itu sendiri dan relasi antar kelas. Kelas dibagi menjadi tiga bagian, yakni nama kelas, atribut kelas, serta operasi kelas (*methods*). Nama kelas adalah nama dari kelas itu sendiri, misalnya kelas mobil, kelas dosen, kelas mahasiswa, dan lain-lain. Penamaan kelas menggunakan kata benda. Atribut adalah data yang dimiliki oleh kelas tersebut. Misalnya kelas mahasiswa memiliki atribut NPM, nama, tanggal lahir, alamat, jemis kelamin dan lain sebagainya. Lalu perasi kelas adalah menunjukan apa yang kelas tersebut bisa lakukan, misalnya kelas dosen dapat melakukan operasi mengajar, absensi, peneltian, dan lain-lain (Sulianta, 2017).

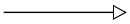
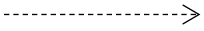


Gambar 2.3 Diagram Kelas

Relasi memilki *multiplicity*, misalnya satu dosen dapat mengajar banyak mata kuliah, dan satu mata kuliah dapat diajar oleh banyak dosen, berarti multiplicity-nya adalah banyak ke banyak.

Tabel 2.2 Simbol *Class Diagram* (Lanjutan)

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem.
Natarmuka/*interface* 	Sama dengan konsep *interface* dalam pemograman berorientasi objek.

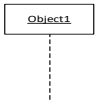
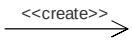
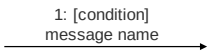
Asosiasi	Relasi antar kelas dalam makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*.
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus).
Kebergantungan 	Relasi antar kelas dengan makna kebergantungan antar kelas.
Agregasi 	Relasi antar kelas dengan maknasemua bagian (*whole-part*).

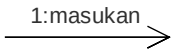
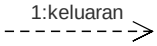
2.7.3 Sequence Diagram

Diagram sequence adalah diagram yang dibuat untuk mengetahui alur dari interaksi antar objek. Isi dari Diagram Sequence harus sama dengan use case dan Diagram Kelas. Satu *use case* tunggal akan digambarkan satu *Diagram Sequence-n*ya (Sulianta, 2017).

Tabel 2.3 Simbol *Squence Diagram*

Simbol	Deskripsi
Aktor 	Orang, proses atau sistem lain yang bernteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang dibuat itu sendiri. Jadi, walaupun simbol dari aktor adalah gambar orang, tapi aktor belum

	tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
Garis hidup 	Men yatakan kehidupan suatu objek.
Objek 	Menyatakan objek yang berinteraksi pesan.
Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi pesan.
Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri. Arah panah mengarah pada objek yang memiliki operasi atau metode karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
Pesan tipe *send*	Menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya,

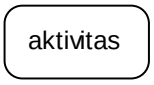
	arah panah mengarah pada objek yang dikirimi.
Pesan tipe *return* 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode yang menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.

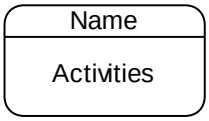
2.7.4 Activity Diagram

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapta dilakukan oleh sistem. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
b. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Tabel 2.4 Simbol Diagram Aktivity

Keterangan	Simbol	Deskripsi
Status awal		Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
Aktivitas		Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.

Percabangan		Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
Penggabungan		Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
Swimlane		Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.
Status akhir		Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

2.8 Basis Data

Basis data *(database)* adalah suatu pengorganisasian sekumpulan data yang saling terkait sehingga memudahkan aktivitas untuk memperoleh informasi. Basis data di maksudkan untuk mengatasi problem pada sistem yang memakai pendekatan berbasis berkas.

Untuk mengelola basis data diperlukan perangkat lunak yang disebut *Database Management Sistem* (DBMS). DBMS adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol, dan mengakses basis data dengan cara yang praktis dan efisien. DBMS dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda.

Terdapat beberapa elemen basis data, yaitu :

a. Database

   Database atau basis data adalah kumpulan tabel yang mempunyai ikatan antara satu tabel dengan tabel lainnya sehingga membentuk suatu bangunan data.

b. Tabel

   Tabel adalah kumpulan *record-record* yang mempunyai panjang elemen yang sama dan atribut yang sama namun berbeda data valuenya.

c. Entitas

   Entitas adalah sekumpulan objek yang terdefinisikan yang mempunyai karakteristik sama dan bisa dibedakan satu dengan lainnya. Objek dapat berupa barang, orang, tempat atau suatu kejadian.

d. Atribut

   Atribut adalah deskripsi data yang bisa mengidentifikasi entitas yang membedakan entitas tersebut dengan entitas yang lain.

2.9 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - Integrated Development Environment (IDE) untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA . Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android. Untuk dapat digunakan dalam pembuatan aplikasi android maka android studio membutuhkan plug-in yang disebut ADT, ADT adalah kepanjangan dari android development tools yang menjadi penghubung antara IDE android studio dengan Android Software Development Kit. Dikutip dari situs resminya (https://developer.android.com/studio/intro/index.html?hl=id)

2.10 Java

Java adalah sebuah bahasa pemrograman yang dapat memenuhi kebutuhan organisasi dengan mengimplemantasi aplikasi berbasis internet dan pernagkat lunak pada alat, yang terhubung melalui jaringan. Salah satu tujuan dibentuknya bahasa pemrograman Java adalah untuk dapat menulis program yang akan dijalankan pada berbagai macam sistem komputer. Hal ini disebut dengan "write once, run anywhere." Yang artinya bahasa pemrograman Java dapat ditulis sekali namun dapat digunakan dan dijalankan dimana saja (pada komputer apa saja). (Deitel,2012)

2.11 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*Database Management Sistem*) atau DBMS yang *multithread, multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GP (Solichin, 2016).

2.12 Metode Pengembangan Sistem

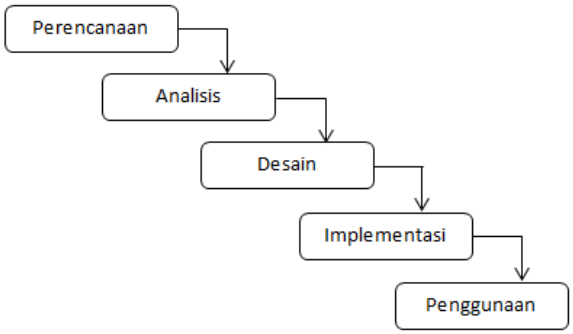
Metode pengembangan sistem sangat dibutuhkan dalam perancangan sebuah sistem karena sebelum memulai pembuatan koding – koding hendaknya merancang terlebih dahulu metode pemodelan seperti apa yang harus digunakan dengan memprioritaskan ketepatan waktu selesai dan efektifitas dalam perancangan sebuah sistem.

2.12.1 Tahapan Pengembangan Sistem

Siklus hidup sistem informasi (dikenal dengan istilah *Sistem Development Life Sycle* (SDLC) atau metode Air Terjun (*Waterfall Method*) memiliki berbagai versi yang bergantung pada seorang spesialis informasi memandang proses pengembangan sistem informasi (Sulianta, 2017). Sisklus hidup sistem versi Raymond McLeod memiliki tahap :

- Perancanaan *(Planning)*
- Analisis *(Analysis)*
- Desain *(Design)*
- Implementasi *(Implementation)*
- Penggunaan *(Use)*

Kebanyakan proses pengembangan sistem informasi mengacu pada versi H. L. Capron dan Raymnond Mc. Leod, Jr.



Gambar 2.2 Tahap Waterfall (Raymond McLeod)

2.13 Sumber Data

Sumber data adalah segala sesuatu yang dapat memberikan informasi mengenai data. Berdasarkan sumbernya, data dibedakan menjadi dua, yaitu data primer dan data sekunder.

1. Data primer

   Data primer ialah data yang diperoleh atau dikumpulkan langsung di lapangan oleh orang yang melakukan penelitian atau yang bersangkutan yang memerlukannya. Data primer di dapat dari sumber informan yaitu individu atau perseorangan seperti hasil wawancara yang dilakukan oleh peneliti. Data primer ini antara lain;

   - Catatan hasil wawancara.
   - Hasil observasi lapangan.
   - Data-data mengenai informan.

2. Data sekunder

   Data sekunder adalah data yang diperoleh atau dikumpulkan oleh orang yang melakukan penelitian dari sumber-sumber yang telah ada. Data ini digunakan untuk mendukung informasi primer yang telah diperoleh yaitu dari bahan pustaka, literatur, penelitian terdahulu, buku, dan lain sebagainya. (Hasan, 2002)

2.14 Pengujian *Software*

Pengujian *software* sangat diperlukan untuk memastikan *software*/aplikasi yang sudah/sedang dibuat dapat berjalan sesuai dengan fungsionalitas yang diharapkan. Pengembang atau penguji software harus menyiapkan sesi khusus untuk menguji program yang sudah dibuat agar kesalahan ataupun kekurangan dapat dideteksi sejak awal dan dikoreksi secepatnya. Pengujian atau testing sendiri merupakan elemen kritis dari jaminan kualitas perangkat lunak dan merupakan bagian yang tidak terpisah dari siklus hidup pengembangan software seperti halnya analisis, desain, dan pengkodean. (Shi, 2010)

2.14.1 Black box-testing

Black Box Testing berfokus pada spesifikasi fungsional dari perangkat lunak. Tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. *Black Box Testing* bukanlah solusi alternatif dari *White Box Testing* tapi lebih merupakan pelengkap untuk menguji hal-hal yang tidak dicakup oleh *White Box Testing.* (Shi, 2010)

Black Box Testing cenderung untuk menemukan hal-hal berikut:

1. Fungsi yang tidak benar atau tidak ada
2. Kesalahan antarmuka (*interface errors*)
3. Kesalahan pada struktur data dan akses basis data
4. Kesalahan performansi (*performance errors*)
5. Kesalahan inisialisasi dan terminasi

Pengujian didesain untuk menjawab pertanyaan-pertanyaan berikut:

1. Bagaimana fungsi-fungsi diuji agar dapat dinyatakan valid?
2. Input seperti apa yang dapat menjadi bahan kasus uji yang baik?
3. Apakah sistem sensitif pada input-input tertentu?
4. Bagaimana sekumpulan data dapat diisolasi?
5. Berapa banyak rata-rata data dan jumlah data yang dapat ditangani sistem?
6. Efek apa yang dapat membuat kombinasi data ditangani spesifik pada operasi sistem?

Saat ini terdapat banyak metoda atau teknik untuk melaksanakan *Black Box Testing*, antara lain:

1. *Equivalence Partitioning*
2. *Boundary Value Analysis/Limit Testing*
3. *Comparison Testing*
4. *Sample Testing*
5. *Robustness Testing*
6. *Behavior Testing*
7. *Requirement Testing*
8. *Performance Testing*
9. Uji Ketahanan (*Endurance Testing*)
10. Uji Sebab-Akibat (*Cause-Effect Relationship Testing*)