BAB II

LANDASAN TEORI

2.1 Sistem

Menurut Julian Chandra & Muhammad Rajab F (2017). Sistem adalah sekelompok elemen yang terintegrasi dengan maksud yang sama untuk mencapai satu tujuan. Sistem juga merupakan suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu.

2.2 Informasi

Menurut Julian Chandra & Muhammad Rajab F (2017). Sumber dari informasi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian (*event)* adalah sesuatu yang terjadi pada saat tertentu. Untuk memperoleh informasi yang berguna, tindakan yang pertama adalah mengumpulkan data, kemudian mengolahnya sehingga menjadi informasi. Dari data-data tersebut informasi yang didapatkan lebih terarah dan penting karena telah dilalui berbagai tahap dalam pengolahannya diantaranya yaitu pengumpulan data, data apa yang terkumpul dan menemukan informasi yang diperlukan. Informasi adalah data yang diolah menjadi bentuk yang berguna bagi pemakainya. Dapat disimpulkan bahwa informasi adalah hasil dari pengolahan data dalam bentuk yang lebih berguna dan lebih berarti bagi penerimanya yang menggambarkan suatu kejadian-kejadian yang nyata yang digunakan untuk pengambilan keputusan.

2.3 Sistem Informasi

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan.(Agus Rahardi, 2018)

2.4 Basis Data

Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat. Sistem informasi tidak dapat dipisahkan dengan kebutuhan akan basis data apapun bentuknya, entah berupa *file* teks ataupun *Data Management System* (DBMS), (Rosa A.S, M.Shalahuddin,2016).

Kebutuhan basis data dalam sistem informasi
meliputi:

1.Memasukkan, menyimpan, dan mengambil data

2.Membuat laporan berdasarkan data yang telah disimpan

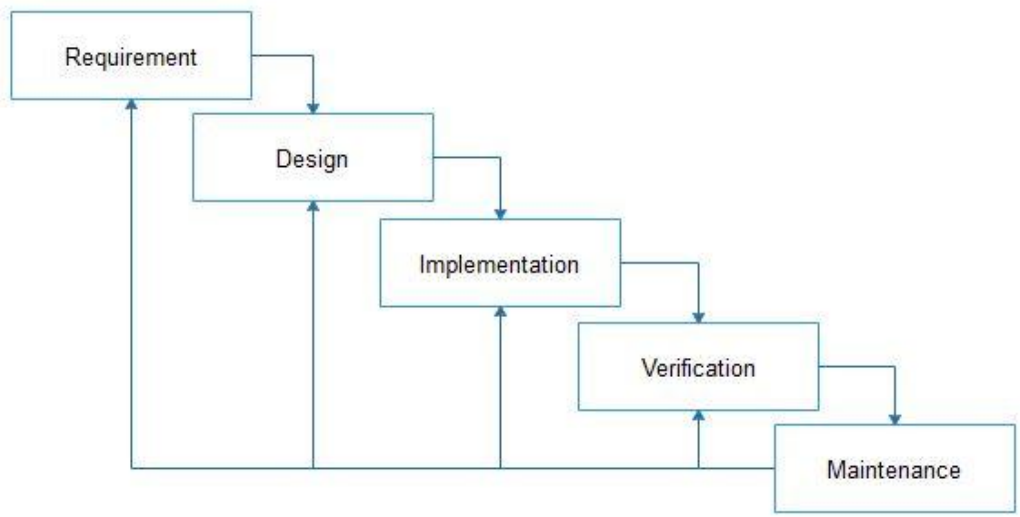
Tujuan dari dibuatnya table-tabel disini adalah untuk menyimpan data ke dalam tabel-tabel agar mudah diakses. Oleh karena itu, untuk merancang tabel-tabel yang akan dibuat maka dibutuhkan pola pikir penyimpanan data nantinya jika dalam bentuk baris-baris data (*record*) dimana setiap baris terdiri dari beberapa kolom, (Rosa A.S, M.Shalahuddin, 2016).

2.5 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.

(Halimah & Anggi Andriyadi, 2019) Model air terjun (*waterfall*) sering juga disebut model sekuensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model *waterfall* menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*).

Berikut adalah gambar model *waterfall:*



Gambar 2.1 model *waterfall*

1. *requirement system*

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user.* Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. *system design*

   Desain perangkat lunak adalah proses multi langkah yang focus pada desain pembuatan program perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain

perangkat lunak yagn dihasilkan pada tahap ini juga perlu didokumentasikan.

3. *implementation*

   Desain harus ditranslasikan ke dalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. *integration & testing*

   Pengujian focus pada perangkat lunak secara dari lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*) dan memastikan keluaran yagn dihasilkan sesuai dengan yang diinginkan.

5. *Operation and Maintenance*

   Dalam tahapan ini, sistem diinstal dan mulai digunakan. Selain itu juga memperbaiki *error* yang tidak ditemukan pada tahap pembuatan. Dalam tahap ini juga dilakukan pengembangan sistem seperti penambahan fitur dan fungsi baru.

2.6 *Unified Modeling Language* (UML)

Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967. Bahasa pemrograman ini kurang berkembang dan dikembangkan lebih lanjut, namun dengan kemunculannya telah memberikan sumbangan yang sebesar pada developer pengembangan bahasa pemrograman berorientasi objek selanjutnya, (Rosa A.S, M.Shalahuddin, 2016).

Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman Smaltalk pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman beorientasi objek yang lainnya seperti C objek, C++, Eiffel, CLOS. Secara actual, pengguna bahasa pemrograman berorientasi objek

pada saat itu masih terbatas, namun telah banyak menarik perhatian disaat itu. Sekitar lima tahun setelah Smalltalk berkembang, maka berkembang pula metode pengembangan berorientasi objek. Metode yang pertama diperkenalkan oleh Sallay Shlaer dan Stephen Mellor (Shlaer-Mellor, 1988) dan Peter Coad dan Edward Yourdon (Coad-Yourdon,1991), diikut oleh Grady Booch (Booch,1991), James R. Rumbaugh, Michael R. Blaha, William Lorensen, Frederick Eddy, William Premerlani (Rumbaugh-Blaha- Premerlani-Eddy-Lorensen,1991), dan masih banyak lagi.

Karena banyaknya metodologi-metodologi yang berkembang pesat saat itu, maka muncullah ide untuk membuat sebuah bahasa yang dapat dimengerti semua orang. Usaha penyatuan ini banyak mengambil dari metodologi-metodologi yang berkembang saat itu. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep sepeerti konsep *Object Modelling Technique* (OMT) dari Rumbaugh dan Booch (1991), konsep *The Classes, Responsibilities, Collaborators* (CRC) dari Rebecca Wirfs-Brock (1990), konsep pemikiran Ivar Jacobson, dan beberapa konsep lain nya dimana James R. Rumbaigh, Grady Booch, dan Ivar Jacobson bergabung dalam sebuah perusahaan yang bernama *Rational Software Corporation* menghasilkan bahasa yang disebuat dengan *Unified Modelling Language* (UML).

Pada 1996, *Object Management Group* (OMG) mengajukan proposal agar adanya standardisasi pemodelan berorientasi objek dan pada bulan September 1997 UML diakomodisi oleh OMG sehingga sampai saat ini UML telah memberikan kontribusinya yang cukup besar didalam metodologi berorientasi objek dan hal-hal yang terkait didalamnya.

Secara fisik, UML adalah sekumpulan spesifikasi yang dikeluarkan oleh OMG. UML terbaru adalah UML 2.3 yang terdiri dari 4 macam spefikasi, yaitu *Diagram Interchange Specification*, UML Infrastructure, UML Superstructure, dan Object Constraint Language (OCL). Seluruh spesifikasic tersebut dapat diakses diwebsite http://www.omg.org, (Rosa A.S, M.Shalahuddin, 2016).

2.6.1 Diagram UML

Untuk mendapatkan banyak pandangan terhadap sistem informasi yang akan dibangun, UML menyediakan beberapa diagram visual yang menunjukkan berbagai aspek dalam sistem. Ada beberapa diagram yang disediakan dalam UML antara lain :

1. Use Case Diagram

Use case atau diagram *Use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendefinisikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu, (Rosa A.S, M.Shalahuddin, 2016).

syarat penamaan pada use care adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut aktor dan *use case*.

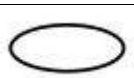
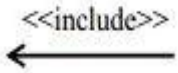
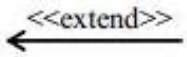
1. Aktor merupakan orang, proses, atau sistem laen yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar kesan antar unit atau aktor.

Berikut adalah tabel simbol-simbol yang ada diagram *use case*:

Tabel 2.1 Simbol *Use Case Diagram*

No	Simbol	Nama	Keterangan
1		**Actor**	Menspesifikasikan himpunan peran yang pengguna mainkan ketika berinteraksi dengan use case.

Tabel 2.2 Simbol *Use Case Diagram* (Lanjutan)

No	Simbol	Nama	Keterangan
2		**Use case**	Deskripi dari urutan aksi – aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu actor.
3		**Association**	Apa yang menghubungkan antara objek satu dengan objek lainnya.
4		**Include**	Menspesifikasikan bahwa use case sumber secara eksplisit.
5		**Extend**	Menspesifikasikan bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.

2. Activity Diagram

Diagram aktivitas atau *avtivity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem, (Rosa A.S, M.Shalahuddin, 2016).

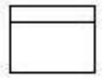
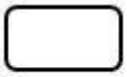
Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didenfinisikan.

2. Uratan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antar muka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut adalah tabel simbol-simbol yang ada pada diagram aktivitas:

Tabel 2.3 Simbol *Activity Diagram*

No	Simbol	Nama	Keterangan
1		**Swimlane**	Menunjukkan siapa yang bertanggung jawab dalam melakukan aktivitas dalam suatu diagram.
2		**Action**	Langkah-langkah dalam sebuah activity. Action bias terjadi saat memasuki activity, meninggalkan activity, atau pada event yang spesifik.
3		**Initial State**	Menunjukkan dimana aliran kerja dimulai.
4		**Activity Final Node**	Menunjukkan dimana aliran kerja diakhiri.

Tabel 2.4 Simbol *Activity Diagram* (lanjutan)

5		**Decision Node**	Menunjukkan suatu keputusan yang mempunyai satu atau lebih transisi sesuai dengan suatu kondisi.
6		**Control Flow**	Menunjukkan bagaimana kendali suatu aktivitas terjadi pada aliran kerja dalam tindakan tertentu.

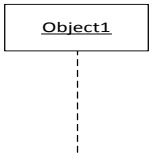
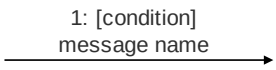
3. Squence Diagram

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendefinisikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstalasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

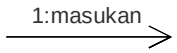
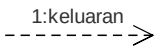
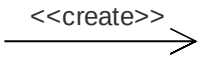
Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalananya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan makan diagram sekuen yang harus dibuat juga semakin banyak, (Rosa A.S, M.Shalahuddin, 2016).

Berikutnya adalah table simbol-simbol yang ada pada Squence Diagram:

Tabel 2.5 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
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 *call* 	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.

Tabel 2.6 Simbol *Squence Diagram* (lanjutan)

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.
Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.

4. Class Diagram

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk mebangun sitem. Kelas memiliki apa yang disebut atribut dan metode atau operasi, (Rosa A.S, M.Shalahuddin, 2016).

1. Atribut merupakan varibel-varibel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

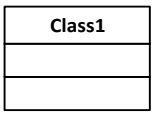
Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan didalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Banyak berbagai kasus, perancangan kelas yang dibuat tidak sesuai

dengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang dirancangan dan hasil jadinya tidak sesuai.

Kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak atau *programmer* dapat membuat kelas-kelas didalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada digram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1. Kelas main
   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2. Kelas yang menangani tampilan sistem (*view*)
   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3. Kelas yang diambil dari pendefinisian *use case* (*controller*)
   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use care*, kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.
4. Kelas yang diambil dari pendefinisian data (*model*)
   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua tabel yang dibuat dibasis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi atau atribut mutivalue pada ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggung jawabkan atau tetap ada didalam perancangan kelas.

Tabel 2.7 Simbol *Class Diagram*

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 Android

Menurut (Nadia Firly, 2018) Android adalah sistem operasi yang berfungsi sebagai penghubung (*device*) antara pengguna dan perangkat keras pada *smarthphone* atau alat elektronik tertentu. Sehingga, hal tersebut

memungkinkan pengguna dapat berinteraksi dengan *device* dan menjalankan berbagai macam aplikasi *mobile.*

2.7.1 Android SDK

Menurut (Nadia Firly, 2018) Android SDK adalah merupakan sebuah kit yang berfungsi untuk mengembangkan berbagai aplikasi berbasis Android oleh para *developer*. Didalam SDK telah terdapat berbagai tools yang bertujuan untuk proses pengembangan aplikasi seperti proses *debugger, emulator, software libraries,* dan dokumentasi.

2.8 PHP (*Hypertext Preprcessor)*

Menurut (Jubilee Enterprise, 2018) PHP merupakan Bahasa pemrograman yang digunakan untuk membuat website dinamis dan interaktif. Dinamis artinya, website tersebut bisa berubah-ubah tampilan dan kontennya sesuai kondisi tertentu. Sebagai contoh, PHP bisa menampilkan tanggal dan hari saat ini secara berganti-ganti di dalam sebuah website. Interaktif artinya, PHP dapat memberi *feedback* bagi *user* (misalnya menampilkan hasil pencarian produk).

2.9 Java

Java menurut definisi dari *Sun Microsystem* adalah nama untuk sekumpulan teknologi untuk membuat dan menajalankan perangkat lunak pada komputer *standalone* ataupun pada lingkungan jaringa. *Java* 2 adalah generasi kedua dari *Java Platform. Java* berdiri atas sebuah mesin *interpreter* yang diberi nama *Java Virtual Machine (*JVM). JVM inilah yang akan membaca *bytecode* dalam *file .class* dari suatu program sebagai representasi langsung program yang berisi bahasa mesin. Oleh karena itu bahasa *Java* disebut sebagai bahasa pemograman yang *portable* karena dapat dijalankan pada berbagai sistem operasi, asalkan pada sistem operasi tersebut terdapat JVM.

Java merupakan bahasa pemrograman objek murni karena semua kode programnya dibungkus dalam kelas. Saat ini *Sun Microsystem* sudah diakuisisi *Oracle Corporation* sehingga pengembangan *Java* diteruskan oleh *Oracle Corporation*(Rosa A.S, M.Shalahuddin, 2016).

2.10 *MySQL*

MySQL adalah sebuah program *database server* yang mampu menerima dan mengirimkan datanya dengan sangat cepat, *multi user* serta menggunakan perintah standar SQL (*Structured Query Language*). *MySQL* juga telah mendukung bahasa pemrograman berfitur API seperti Java sehingga memudahkan para programmer java untuk berkoneksi dengan menggunakan *MySQL*. (Rosa A.S, M.Shalahuddin, 2016).

2.11 Destinasi

Destinasi adalah segala sesuatu yang ada di daerah tujuan wisata yang merupakan daya tarik orang-orang mau *dating* berkunjunng ke tempat tersebut atau semua hal yang menarik untuk dilihat dan dan dirasakan wisatawan yang disediakan atau bersumber pada alam saja. (Fakhri Nur Haffiyan, 2018)

2.12 GIS (*Geographic Information Sistem*)

Menurut Budi Harto & Zaini Kurniawan (2017). Sistem Informasi Gegografis atau disingkat SIG merupakan suatu sistem berbasis komputer yang digunakan untuk mengumpulkan, menyimpan, mengatur, menggabungkan dan menganalisis data-data geografis. Pengertian informasi geografis adalah informasi mengenai tempat-tempat yang terletak di permukaan bumi, pengetahuan mengenai posisi dimana suatu objek terletak di permukaan bumi dan informasi mengenai keterangan-keterangan yang terdapat di permukaan bumi yang posisinya diketahui. Objek-objek dan fenomena-fenomena dimana lokasi geografis itu berada

penting dianalisis demi pengambilan keputusan-keputusan atau demi kepentingan-kepentingan tertentu.