BAB II

LANDASAN TEORI

2.1. Konsep Dasar Sistem

2.1.1 Pengertian Sistem

Sistem dapat didefinisikan sebagai serangkaian komponen yang yang dikoordinasikan untuk mencapai serangkaian tujuan. Suatu perusahaan sangat bergantung pada sistem sebagai dasar untuk melaksanakan aktivitas. Informasi dihasilkan oleh sistem informasi yang merupakan alat untuk memproses kebutuhan bahan baku pada setiap bagian perusahaan.“Sistem adalah kumpulan/group dari sub sistem/bagian/komponen apapun baik *phisik* ataupun *non phisik* yang saling berhubungan satu sama lain dan bekerja sama secara harmonis untuk mencapai tujuan tertentu”.Sistem didefinisikan sebagai sekumpulan prosedur yang saling berkaitan dan saling terhubung untuk melakukan suatu tugas bersama-sama (Dewi, Saryoko, & Sukmana, 2018).

2.1.2. Karakteristik Sistem

Bahwa “sebuah sistem memiliki karakteristik atau sifat-sifat tertentu yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu *system*” (Hamim, 2014:2). Adapun karakteristik yang dimaksud sebagai berikut :

1. Komponen Sistem (*Components System*) suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang bekerja sama membentuk satu kesatuan.

Setiap sub sistem memiliki sifat-sifat yang menjalankan suatu fungsi tertentu dan mempengaruhi sistem secara keseluruhan.

2. Batasan Sistem (*Boundary*) Ruang lingkup system merupakan daerah yang membatasi antara system dengan yang lainnya atau system dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkugan Luar Sistem (*Environtment*) Bentuk apapun yang ada di luar ruang lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut dengan lingkungan luar system. Lingkungan luar sistem ini dapat menguntungkan dan dapat juga merugikan sistem tersebut. Lingkungan luar tersebut harus selalu dijaga dan dipelihara. Sementara, lingkungan luar yang merugikan harus dikendalikan, kalau tidak akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Sistem (*Interface*) Media yang menghubungkan sistem dengan sub system yang lain disebut dengan penghubung system atau interface. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsystem yang lain. Keluaran suatu subsystem akan menjadi masukan untuk subsistem yang lain dengan melewati penghubung. Dengan demikian, terjadi suatu integritas sistem yang membentuk satu kesatuan.

5. Masukkan Sistem (*Input*) Energi yang dimasukan ke dalam sistem disebut masukan sistem, yang berupa pemeliharaan (*Maintenance Input*) dan sinyal (*Signal Input*). Sebagai contoh, di masukan suatu unit sistem komputer “program“ adalah maintenance input yang digunakan untuk

mengoprasikan komputer. Sementara “data“ adalah signal input yang akan diolah menjadi informasi.

6. Keluaran Sistem (*Output*) Hasil dari energy yang diolah dan di kalsifikasi menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain, sedangkan contoh informasi keluaran yang dihasilkan adalah informasi ini adalah informasi, di mana informasi ini dapat digunakan sebagai masukan untuk pengambilan keputusan atau hal-hal yang merupakan input bagi subsitem lainnya.

7. Pengolahan Sistem (*Procces System*) Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Sebagai contoh, sistem akuntansi sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem (*Objective System*) Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik, suatu sistem tidak memiliki sasaran maka operasi sistem yang tidak ada gunanya.Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan.

2.1.3. Teori Konsep Dasar Sistem

A. Android

Menurut (Safaat H, 2014:1) “Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi”. Android menyediakan *platform* terbuka bagi para pengembang untuk

menciptakan aplikasi mereka. Android sebagai "*platform* mobile pertama yang lengkap, terbuka, dan bebas" (Safaat H, 2014:3).

1. Lengkap *(Complete Platform)*

   Sistem operasi android merupakan sistem operasi yang aman dan banyak menyediakan *tools* dalam mengembakan aplikasi yang dapat digunakan kembali oleh *developer* untuk mengembangkan aplikasinya.

2. Terbuka *(Open Source Platform)*

   Platform Android yang bersifat *open source* (terbuka), hal ini yang membuat sistem operasi android mudah dikembangkan oleh *developer*.

3. Bebas *(Free Platform)*

   Developer tidak perlu membayar royalti untuk memperoleh license sehingga dapat dengan bebas mengembangkan, mendistribusikan dan memperdangankan sistem operasi android.

BINA SARANA INFORMATIKA
UNIVERSITAS

Sumber: (Safaat H, Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android, 2014:1)

Gambar II.1. Logo Android

B. SQLite

"SQLite adalah sebuah *open source* database yang telah ada cukup lama, cukup stabil, dan sangat terkenal pada perangkat kecil, termasuk Android". SQLite cocok digunakan sebagai database dalam Android, maupun Berbagai program yang membutuhkan kemudahan serta tidak membutuhkan *resource* besar. SQLite menggunakan konsep *Zero Configuration* dan *embedded* database, yaitu sebuah konsep dimana *developer* tidak perlu menginstall ataupun melakukan setup database kedalam sebuah server database sehingga tidak membutuhkan resource yang besar dan proses pembuatan database pun menjadi instan. Selain itu dengan menggunakan konsep *embedded* maka database pada SQLite digunakan sebagai *library* dan di *compile* bersamaan dengan program. SQLite saat ini banyak digunakan di dalam aplikasi dan program termasuk dalam beberapa *high profile project*. SQLite juga merupakan mesin database SQL *embedded* yang berbeda dengan kebanyakan database SQL lainnya. seperti yang dijelaskan sebelum nya SQLite tidak memiliki proses server yang terpisah sehinggaSQLite membaca dan menulis secara langsung ke disk (Hipp, 2018)

C. Android Studio

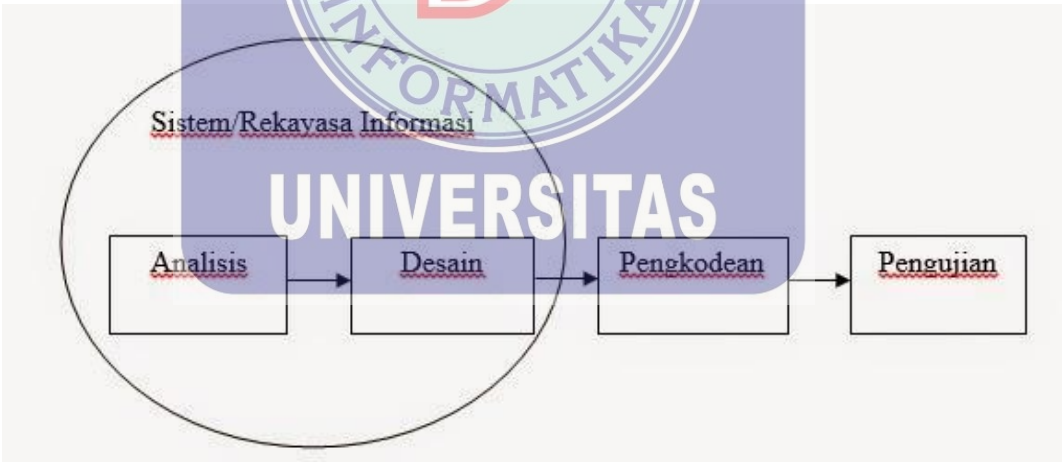
"Android Studio adalah sebuah IDE yang bisa digunakan untuk pengembangan aplikasi android, dan dikembangkan oleh google. Android Studio merupakan suatu pengembangan dari Eclipse IDE, dan dibuat berdasarkan IDE Java populer, yaitu IntelliJ IDEA" (Developers, 2018)

D. JDK *(Java Development Kit)*

JDK adalah *Sun Microsystem* produk ditujukan untuk pengembangan Java. Sejak diperkenalkannya Java, telah jauh SDK Java yang paling banyak digunakan. Pada Tanggal 17 November 2006, Sun mengumumkan bahwa akan dirilis dibawah GNU *General Public License* (GPL), sehingga membuat perangkat lunak bebas (Wikipedia, 2018).

E. Metode *Waterfall*

Model air terjun(*waterfall*) sering juga disebut model sekuensial linear(*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidupperangkat lunak secara sekuensial atau terurut dimulai dari analisis desain,pengkodean,pengujian,tahap pendukung*(support*). Berikut adalah gambar model air terjun:



Sumber : Sukamto & Shalahuddin (2015:29)

Gambar II.2 *Waterfall*

1. Analisis kebutuhan perangkat lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan 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. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitek perangkat lunak, representasi antarmuka, dan prosedur pengkodean. Tahap ini mentranslasikan kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Pembuatan kode program

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

4. Pengujian

   Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal inidilakukan untuk meminimalisir kesalahan (error) dan memastikan keluaran yang dihasilkan dengan yang diinginkan.

5. Pendukung (support) atau pemeliharaan (maintenance)

   Tidak menutupi kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke user, perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi

proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.2. Teori Pendukung

A. *Entity Relation Diagram* (ERD)

Entity Relationship Diagram (ERD) merupakan bentuk paling awal dalam melakukan perancangan basis data relasional. "ERD biasanya memiliki hubungan binary (satu relasi menghubungkan dua buah entitas). Beberapa metode perancangan ERD menoleransi hubungan relasi ternary (satu relasi menghubungkan tiga buah relasi) atau N-ary (satu relasi menghubungkan banyak entitas), tapi banyak metode perancangan ERD yang tidak mengizinkan ternary atau N-ary" Menurut (Sukamto & Shalahuddin, 2015:52).

B. LRS *(Logical Record Structure)*

LRS merupakan hasil dari dari transformasi dalam tahapan kardinalitas dari ERD ke LRS dan menghasilkan atribut-atribut yang saling berelasi.

1. Pengertian *Logical Record Structure* (LRS)

Logical Record Structure (LRS) "merupakan hasil transformasi ERD ke LRS yang melalui proses kardinalitas dan menghasilkan atribut-atribut yang saling berelasi*"* menurut (Al-Bahra 2013:159).

Aturan pokok dalam melakukan transformasi E-R Diagram ke logical record structure sangat dipengaruhi oleh elemen yang menjadi titik perhatian utama pada langkah transformasi dengan proses kardinalitas, yang terdiri dari tiga kardinalitas yaitu sebagai berikut (Al-Bahra 2013:159) :

a. *One to One*

Yaitu proses kardinalitas yang panahnya lebih diarahkan di entity dengan jumlah atribut yang lebih sedikit.

b. *One to Many*

Relasi harus diagbungkan dengan entity pada pihak *many*, dan tidak perlu melihat banyak sedikitnya pada entity tersebut.

c. *Many to Many*

Yaitu proses kardinalitas pada *relationship* berubah status menjadi file konektor, sehingga baik *entity* maupun relasi akan menjadi struktur *record* sendiri.

2. Derajat *Relationship*

Derajat *relationship* atau kardinalitas menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Bahwa "Derajat *Relationship* adalah jumlah entitas yang berpartisipasi dalam satu *Relationship*" menurut (Al-Bahra 2013:144).

Derajat *relationship* yang sering dipakai didalam ERD menurut (Al-Bahra 2013:145) adalah sebagai berikut:

a. *Unary Relationship*

Unary relationship adalah model yang terjadi diantara entity yang berasal dari *entity* set yang sama. Sering juga disebut sebagai *recursive relationship* atau *reflective relationship*.

b. Binary Relationship

Adalah model *relationship* antara *instance-instance* dari suatu tipe entitas (dua *entity* yang berasal dari *entity* yang sama). *Relationship* ini paling umum digunakan dalam pembuatan model data.

c. Ternary Relationship

Ternary Relationship merupakan *relationship* antara *instance-instance* dari tiga tipe entitas sepihak.

2.2.1. *Unified Modeling Language* (UML)

2.2.1.1. Pengenalan *Unified Modeling Language* (UML)

"Unified Modeling Language (UML) adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek". UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung (Sukamto & Shalahuddin, 2015:133).

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun, dan dokumentasi dari sistem perangkat lunak.

UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

2.2.1.2. Sejarah *UML*

"Bahasa pemrograman berorientasi objek yang pertama dikembangkan dikenal dengan nama Simula-67 yang dikembangkan pada tahun 1967.

Perkembangan aktif dari pemrograman berorientasi objek mulai menggeliat ketika berkembangnya bahasa pemrograman *Smalltalk* pada awal 1980-an yang kemudian diikuti dengan perkembangan bahasa pemrograman berorientasi objek yang lainnya seperti C objek, C++, *Eiffel*, dan CLOS.

Sekitar lima tahun setelah Smalltalk berkembang, maka berkembang pula metode pengembangan berorientasi objek. Karena banyaknya metodologi-metodologi yang berkembang pesat saat itu. Maka dibuat bahasa yang merupakan gabungan dari beberapa konsep, seperti konsep *Object Modeling 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 lainnya dimana *James R. Rumbaigh, Grady Booch,* dan *Ivar Jacobson* bergabung dalam sebuah perusahaan yang bernama *Rational Software Corporation* menghasilkan bahasa yang disebut dengan *Unified Modeling Language* (UML).

Pada tahun 1996, *Object Management Group* (OMG) mengajukan proposal agar adanya standarisasi pemodelan berorientasi objek dan pada bulan September 1997 UML diakomodasi oleh OMG sehingga sampai saat ini UML telah memberikan kontribusinya yang cukup besar di dalam metodologi berorientasi objek dan hal-hal yang terkait di dalamnya" (Sukamto & Shalahuddin, 2015:138).

2.2.1.3. Diagram *UML*

"UML terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori" (Sukamto & Shalahuddin, 2015:140). Berikut ini penjelasan singkat dari pembagian kategori tersebut.

1. *Structure diagram*, yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan. *Structure diagram* terdiri dari *class diagram, object diagram, component diagram, composite structure diagram, package diagram dan deployment diagram.*
2. *Behavior diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem. Behavior diagram terdiri dari *Use case diagram, Aktivity diagram, State Machine System.*
3. *Interaction diagram* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem. *Interaction diagram* terdiri dari *Sequence Diagram, Communication Diagram, Timing Diagram, Interaction Overview Diagram.*

2.2.2. *Use Case Diagram*

"Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan 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 di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu" (Sukamto & Shalahuddin, 2015:155).

2.2.3. *Activity Diagram*

"Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu di perhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh system" (Sukamto & Shalahuddin, 2015:161).

2.2.4. *Class Diagram*

"Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan *method* atau operasi" (Sukamto & Shalahuddin, 2015:141).

Berikut penjelasan atribut dan *method* :

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau *method* adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

2.2.5. *Sequence Diagram*

"Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dengan *massage* 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 diinstansiasi 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 jalannya pesan sudah dicakup dalam diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak (Sukamto & Shalahuddin, 2015:165).

BINA SARANA INFORMATIKA
UNIVERSITAS