# BAB II
# LANDASAN TEORI

## 2.1 Sistem

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran yang tertentu.(Antonio and Safriadi, 2012).

Adapun menurut refrensi (Mulyani, 2016) berpendapat "Sistem adalah kumpulan dari sub sistem / bagian / komponen apapun baik fisik ataupun nonfisik yang saling berhubungan satu sama lain dan bekerja secara harmonis untuk mencapai satu tujuan tertentu."

## 2.2 Pengertian Jejaring Sosial

Jejaring sosial menjadi fenomena yang semakin sering digunakan, keberadaannya semakin tidak bisa dipisahkan dari cara berkomunikasi antar manusia. Sebagai bentuk aplikasi dalam komunikasi secara virtual, jejaring sosial merupakan kemajuan Teknologi Informasi dan Komunikasi (TIK) atau *Information Communication Technology* (ICT).

Menurut (Herliani, 2015) mengemukakan bahwa "Jejaring sosial (social network) adalah bentuk struktur sosial yang terdiri dari simpul-simpul yang saling terkait dan terikat oleh satu atau lebih tipe hubungan yang spesifik. Simpul-simpul yang dimaksudkan disini dapat berupa individu maupun organisasi.".

## 2.3 *Android*

Menurut (Hartati *et al.*, 2017) "Android adalah sebuah sistem operasi pada handphone yang bersifat terbuka dan berbasis pada sistem operasi Linux . Android juga dapat di kembangkan oleh setiap progeremer yang memiliki kemampuan

dibaidang system oprasi Linux dan Bahas pemograman Java. Android juga menyediakan platform terbuka atau open score bagai para programmer untuk mengembangkan aplikasi yang akan digunakan untuk bermacam peranti bergerak"

## 2.4 *Android SDK*

Menurut (Nur Yati, 2018) "Android-SDK merupakan tools bagi para programmer yang ingin mengembangkan aplikasi berbasis google android. Android SDK mencakup seperangkat alat pengembangan yang komprehensif. Android SDK terdiri dari debugger, libraries, handset emulator,dokumentasi, contoh kode, dan tutorial. Saat ini Android sudah mendukung arsitektur x86 pada Linux (distribusi Linux apapun untuk desktop modern), Mac OS X 10.4.8 atau lebih, Windows XP atau Vista."

## 2.5 *Java Development Kit*

*Java Development Kit* (JDK) merupakan perangkat lunak yang digunakan untuk manajemen dan membangun berbagai aplikasi Java. JDK merupakan superset dari JRE, berisikan segala susuatu yang ada di JRE (*Java Runtime Envirotment*) ditambahkan *compiler* dan *debugger* yang diperlukan untuk membangun aplikasi.(Maiyana, 2018)

## 2.6 *Java*

Java adalah bahasa pemrograman tingkat tinggi yang berorientasi objek dan program java tersusun dari bagian yang disebut kelas. Kelas terdiri atas metode-metode yang melakukan pekerjaan dan mengembalikan informasi setelah melakukan tugasnya. Para pemrogram Java banyak mengambil keuntungan dari kumpulan kelas di pustaka kelas Java, yang disebut dengan Java Application Programming Interface (API). Kelas-kelas ini diorganisasikan menjadi sekelompok yang disebut paket (package). Java API telah menyediakan fungsionalitas yang memadai untuk menciptakan applet dan aplikasi canggih. Jadi ada dua hal yang harus dipelajari dalam Java, yaitu mempelajari bahasa Java dan

bagaimana mempergunakan kelas pada Java API. Kelas merupakan satusatunya cara menyatakan bagian eksekusi program.(Putri, 2014)

## 2.7 Aplikasi Pendukung

### *a. Android Studio*

Android Studio adalah sebuah IDE untuk *Android Development* yang diperkenalkan google pada acara Google I/O 2013. Android Studio merupakan pengembangkan dari Eclipse IDE, dan dibuat berdasarkan IDE Java populer, yaitu IntelliJ IDEA. Android Studio merupakan IDE resmi untuk pengembangan aplikasi Android. Sebagai pengembangan dari Eclipse, Android Studio mempunyai banyak fitur-fitur baru dibandingkan dengan Eclipse IDE.(Susanty, Astari and Thamrin, 2019)

### b. *Firebase*

Firebase adalah penyedia layanan *cloud* dengan *back-end* sebagai servis yang berbasis di San Fransisco, California. Firebase membuat sejumlah produk untuk pengembangan aplikasi *Mobile* ataupun web. Firebase didirikan oleh Andrew Lee dan James Tamplin pada tahun 2011 dan diluncurkan dengan *cloud database* secara *realtime* di tahun 2012. Produk utama dari Firebase yakni suatu database yang menyediakan API untuk memungkinkan pengembang menyimpan dan mensinkronisasi data lewat *multiple client*. Perusahaan ini diakusisi oleh Google pada Oktober 2014. Semua data Firebase Realtime *Database* disimpan sebagai objek JSON. Bisa dianggap basis data sebagai *JSON tree* yang di-*host* di awan. Tidak seperti basis data SQL, tidak ada tabel atau rekaman. Ketika ditambahkan ke JSON tree, data akan menjadi simpul dalam struktur JSON yang ada. (Sonita and Fardianitama, 2018)

Ada empat metode untuk menulis data ke *Firebase Realtime Database* Untuk operasi tulis dasar,Anda bisa menggunakan *setValue()* untuk menyimpan data ke referensi yang ditetapkan, menggantikan data yang ada di jalur tersebut.

| Metode | Penggunaan umum |
|---|---|
| `setValue()` | Menulis atau mengganti data ke jalur yang didefinisikan, seperti `users/<user-id>/<username>`. |
| `push()` | Tambahkan ke daftar data. Setiap kali Anda memanggil `push()`, Firebase akan menghasilkan ID unik, seperti `user-posts/<user-id>/<unique-post-id>`. |
| `updateChildren()` | Memperbarui beberapa kunci untuk jalur yang didefinisikan tanpa mengganti semua data. |
| `runTransaction()` | Memperbarui data kompleks yang bisa rusak karena pembaruan bersamaan. |

Gambar 2.1 Metode Menulis Data ke Firebase

| Listener | Callback kejadian | Penggunaan biasa |
|---|---|---|
| `ValueEventListener` | `onDataChange()` | Membaca dan mendengarkan perubahan untuk seluruh konten jalur. |
| `ChildEventListener` | `onChildAdded()` | Mengambil daftar item atau mendengarkan penambahan daftar item. Disarankan untuk digunakan dengan `onChildChanged()` dan `onChildRemoved()` untuk memantau perubahan daftar. |
| | `onChildChanged()` | Mendengarkan perubahan pada item dalam daftar. Gunakan dengan `onChildAdded()` dan`onChildRemoved()` untuk memantau perubahan daftar. |
| | `onChildRemoved()` | Mendengarkan item yang dibuang dari daftar. Gunakan dengan `onChildAdded()` dan`onChildChanged()` untuk memantau perubahan daftar. |
| | `onChildMoved()` | Gunakan dengan data diurutkan untuk mendengarkan perubahan dalam prioritas item. |

Gambar 2.2 Callback Kejadian dalam Pengambilan Data Firebase

1. untuk menambahkan listener kejadian, gunakan metode *addValueEventListener()* atau bisa juga mengunakan *addListenerForSingleValueEvent()*,
2. untuk menambahkan listener kejadian anak, gunakan metode *addChildEventListener()*,
3. metode *onDataChange()* untuk membaca cuplikan statis konten pada jalur tertentu, seperti yang telah ada pada saat kejadian. Metode ini terpicu satu kali 10 ketika listener terpasang dan terpicu lagi setiap kali terjadi perubahan data, termasuk anaknya. *Callback* kejadian meneruskan cuplikan yang berisi semua data di lokasi tersebut, termasuk data anak.

Jika tidak ada data, cuplikan yang dikembalikan adalah *null*. Metode *onDataChange()* dipanggil setiap kali terjadi perubahan data pada referensi database yang ditetapkan, termasuk perubahan ke anaknya. (*Firebase*, 2015).

## 2.8 JSON

*JavaScript Object Notation* (JSON) merupakan *lightweight* data *interchange* yang berbasis *JavaScript Programming Language*. JSON berbasis teks/tulisan dengan format yang dapat dibaca dan dikenali oleh manusia untuk merepresentasikan struktur data sederhana dan *array asosiatif*. JSON merupakan bahasa independen yang lengkap dan menggunakan konvensi yang familiar bagi para programmer bahasa C, antara lain bahasa pemrograman C, C++, C#, Java, JavaScript, Perl, Python, dan lainnya (Sudirman, 2016)

## 2.9 Alat Bantu Perancangan Sistem

### a. Diagram UML

#### *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, (Arifin, Luthfi Mahendra and Kurnia Handayani, 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*.

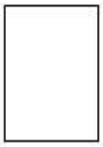
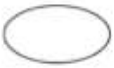
a) Aktor merupakan orang, proses, atau sistem laen yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi

yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.

b) *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 | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| 1 |  | *Actor* | Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*. |
| 2 | << include >>  | *Include* | Menspesifikasikan bahwa *use case* sumber secara *eksplisit*. |
| 3 | << extend >>  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
| 4 | | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 5 |  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 6 |  | *Use Case* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |

### *2. Activity Diagram*

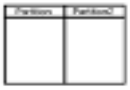
Diagram aktivitas atau *avtivity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor,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

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

a) Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didenfinisikan.
b) Uratan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antar muka tampilan.
c) Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
d) Rancangan menu yang ditampilkan pada perangkat lunak.

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

Tabel 2.2. Simbol *Activity Diagram*

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| 1 |  | *Swimlane* | Menunjukkan siapa yang bertanggung jawab dalam melakukan aktivitas dalam suatu diagram. |
| 2 |  | *Action* | Langkah-langkah dalam sebuah activity. Action bisa 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. |
| 5 |  | *Decision Node* | Menunjukkan suatu keputusan yang mempunyai satu atau lebih transisi dan dua atau lebih transisi sesuai dengan suatu kondisi. |
| 6 |  | *Control Flow* | Menunjukkan bagaimana kendali suatu aktivitas terjadi pada aliran kerja dalam tindakan tertentu. |

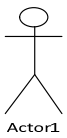
### 3. *Squence Diagram*

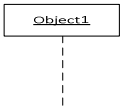
Diagram *Squence* menggambarkan kelakuan objek pada *use case* dengan mendefinisikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode 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, (Arifin, Luthfi Mahendra and Kurnia Handayani, 2016)

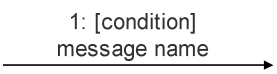
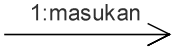
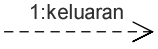
Berikut adalah tabel simbol-simbol yang ada pada Squence Diagram.

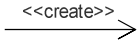
Tabel 2.3. Simbol *Squence Diagram*

| **Simbol** | **Deskripsi** |
|---|---|
| Aktor   | Orang, proses atau sistem lain yang berinteraksi dengan system informasi yang akan dibuat di luar system informasi yang dibuat itu sendiri |
| Garis hidup  | Menyatakan kehidupan suatu objek. |

| | |
|---|---|
| Objek   | Menyatakan objek yang berinteraksi pesan. |
| Waktu aktif  | Menyatakan objek dalam keadaan aktif dan berinteraksi pesan. |

Tabel 2.4 Simbol *Squence Diagram* (lanjutan)

| **Simbol** | **Deskripsi** |
|---|---|
| Pesan tipe *call*   | Orang, proses atau sistem lain yang berinteraksi dengan system informasi yang akan dibuat di luar system informasi yang dibuat itu sendiri |
| Pesan tipe *send*   | Menyatakan bahwa suatu objek mengirimkan data /masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim. |
| 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*<br>  | 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, (Arifin, Luthfi Mahendra and Kurnia Handayani, 2016)

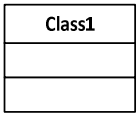
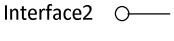
a) Atribut merupakan varibel-varibel yang dimiliki oleh suatu kelas.
b) Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

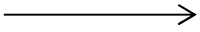
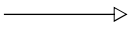
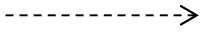
Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan didalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Banyak berbagai kasus, perancangan kelas yang dibuat tidak sesuai dengan kelas-kelas yang dibuat pada perangkat lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang dirancangan dan hasil jadinya tidak sesuai.

Kelas-kelas yang ada pada struktur system harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan system 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.

a) Kelas main, Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
b) Kelas yang menangani tampilan sistem (*view*), Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
c) 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.
d) 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 di jadikan 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 2.5 Simbol *Class Diagram*

| **Simbol** | **Deskripsi** |
|---|---|
| Kelas   | Kelas pada struktur sistem. |
| atarmuka/*interface*   | Sama dengan konsep *interface* dalam pemograman berorientasi objek. |
| Asosiasi  | Relasi antar kelas dalam makna umum, asosiasi biasanya juga disertai dengan *multiplicity*. |

| | |
|---|---|
| Asosiasi berarah  | Relasiantar kelas dengan makna kelas yang satudi gunakan 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 makna semua bagian (*whole-part*). |