BAB 2
TINJAUAN PUSTAKA

2.1 Profil Dinas Pemuda dan Olahraga Kota Bandung

Dinas Pemuda dan Olahraga Kota Bandung dibentuk berdasarkan Peraturan Daerah Nomor 13 Tahun 2009 pasal 17C mempunyai tugas melaksanakan sebagian urusan Pemerintahan Daerah di bidang pemuda dan olahraga berdasarkan asas otonomi dan pembantuan. Mengacu pada Peraturan Presiden Nomor 29 Tahun 2014, tentang Akuntabilitas Kinerja Instansi Pemerintah, Dinas Pemuda dan Olahraga sebagai instansi pemerintah dan unsur penyelenggara negara diwajibkan menetapkan target kinerja dan melakukan pengukuran kinerja yang telah dicapai serta menyampaikan Laporan Kinerja Instansi Pemerintah (LKIP) [8].

2.1.1 Tugas Pokok dan Fungsi

Berdasarkan Peraturan Daerah Nomor 13 Tahun 2009 pasal 17C Dinas Pemuda dan Olahraga Kota Bandung mempunyai tugas melaksanakan sebagian urusan Pemerintahan Daerah di bidang pemuda dan olahraga berdasarkan asas otonomi dan pembantuan.

Untuk melaksanakan tugas tersebut Dinas Pemuda dan Olahraga Kota Bandung mempunyai fungsi :

1. Perumusan kebijakan teknis bidang Pemuda dan Olahraga;
2. Penyelenggaraan sebagaian urusan pemerintahan daerah dan pelayanan umum di bidang pemuda dan olahraga;
3. Pembinaan dan Pelaksanaan tugas di bidang pemuda dan olahraga yang meliputi kepemudaan, keolahragaan serta sarana dan prasarana;
4. Pelaksanaan teknis administratif Dinas; dan

5. Pelaksanaan tugas lain yang diberikan Walikota sesuai dengan tugas dan fungsinya.

Struktur Organisasi Dinas Pemuda dan Olahraga Kota Bandung adalah sebagai berikut:

1. Kepala Dinas
2. Sekretariat, membawahkan:
    a. Sub Bagian Umum dan Kepegawaian;
    b. Sub Bagian Keuangan dan Program
3. Bidang Kepemudaan, membawahkan:
    a. Seksi Bina Prestasi dan Pemberdayaan Pemuda;
    b. Seksi Bina Organisasi Kepemudaan dan Kemitraan.
4. Bidang Keolahragaan, membawahkan:
    a. Seksi Bina Olahraga Masyarakat;
    b. Seksi Bina Profesi, Organisasi dan Kemitraan.
5. Bidang Sarana dan Prasarana, membawahkan:
    a. Seksi Sarana dan Prasarana Kepemudaan;
    b. Seksi Sarana dan Prasarana Olahraga.
6. Unit Pelaksana Teknis Dinas;
7. Kelompok Jabatan Fungsional.

2.1.2 Visi dan Misi Dinas Pemuda dan Olahraga Kota Bandung

Visi misi adalah suatu konsep perencanaan yang disertai dengan tindakan sesuai dengan apa yang direncanakan untuk mencapai suatu tujuan.

Visi dari Dinas Pemuda dan Olahraga Kota Bandung adalah:
"Mewujudkan Pemuda dan Olahraga yang Berprestasi, Inovatif dan Sehat" [9].

Selanjutnya untuk mewujudkan visi tersebut, Dinas Pemuda dan Olahraga Kota Bandung memiliki misi yaitu:

1. Meningkatkan kemandirian Pemuda

2. Meningkatkan Olahraga yang Berkualitas, Berprestasi dan Memasyarakat;
3. Meningkatkan Pelayanan Publik dan Sarana Prasarana Pemuda dan Olahraga;
4. Meningkatkan kinerja aparatur dan e-goverment;

2.2 Landasan Teori

Menurut Kamus Besar Bahasa Indonesia Landasan adalah dasar atau tumpuan, sedangkan Teori adalah pendapat yang didasarkan pada penelitian dan penemuan yang didukung oleh data dan argumentasi. Selain itu Teori juga berarti generalisasi atau kumpulan generalisasi yang dapat digunakan untuk menjelaskan berbagai fenomena secara sistematik. (Wiliam Wiersma dalam Sugiyono, 2010:52).

Suatu landasan teori dari suatu penelitian disini bisa disimpulkan sebagai studi literatur atau tinjauan pustaka. Hasil dari landasan teori atau kajian teori ini diperoleh kesimpulan-kesimpulan atau pendapat-pendapat para ahli, kemudian dirumuskan pada pendapat baru.

2.2.1 Stadion

Stadion merupakan salah satu sarana dan prasarana untuk melakukan kegiatan keolahragaan khususnya sepak bola. Sebuah stadion yang baik selalu dikelola secara baik juga. Mulai dari manajemen kepengurusannya sampai pengelolaan sarana dan prasarananya. Pengelolaan stadion merupakan bagian dari proses pembangunan nasional khususnya pada upaya peningkatan kualitas sumber daya manusia yang mengarah pada: (1) peningkatan kesehatan jasmani masyarakat, (2) kualitas mental rohani masyarakat, (3) pembentukan watak dan kepribadian bangsa, (4) disiplin dan sportivitas, serta (5) peningkatan prestasi yang dapat membangkitkan rasa kebanggaan nasional (Kristiyanto, 2012: 3). Maka dengan demikian tujuan olahraga akan dapat tercapai secara efektif jika terpenuhinya sebuah standarisasi sarana-prasarana keolahragaan [10].

2.2.1.1 Stadion Gelora Bandung Lautan Api

Stadion Gelora Bandung Lautan Api (GBLA) merupakan sebuah stadion olahraga yang dibangun tahun 2009 dan diresmikan pada tahun 2013 (Gambar 2.1). Sempat direnovasi pada Juli–Agustus 2016, Stadion Gelora Bandung Lautan Api terletak di Desa Rancanumpang, Kecamatan Gedebage, Kota Bandung, Jawa Barat dan dimiliki oleh Pemerintahan Kota Bandung. Mampu menampung kapasitas hingga 38.000 penonton, Stadion Gelora Bandung Lautan Api menjadi markas dari kesebelasan Persib Bandung selain Stadion Si Jalak Harupat yang didirikan pada tahun 2003 (dibuka pada tahun 2005) dan berlokasi di Kabupaten Bandung.

Gambar 2.1 Stadion Gelora Bandung Lautan Api

Sosok Gelora Bandung Lautan Api sendiri dapat dilihat dari sepanjang jalan Tol Cileunyi–Padalarang, kira-kira pada KM 149–151, begitu juga bila menggunakan transportasi pesawat terbang ketika take-off atau landing di Bandar Udara Husein Sastranegara, Bandung. Stadion Gelora Bandung Lautan Api terlihat mencolok pada lingkungan sekitarnya karena bentuknya yang modern dan menyerupai pesawat UFO (*Unidentified Flying Object*) [11].

2.2.2 *Radio Frequency Identification* (RFID)

Sensor Radio Frequency Identification (RFID) adalah teknologi yang mampu mengidentifikasi berbagai objek menggunakan gelombang radio (AkintolaKolawole dan Olutayo Kehinde,2011:37). Sistem RFID terdiri dari 4 komponen yaituRFID tag (transponder), antena, *reader*, dan interface software(Miguel, et all., 2011:339) [12].

1. RFID tag (transponder) memiliki chip yang dapat menyimpan data berupa nomer ID unik dan memiliki antena yang berfungsi untuk mentransmisikan data ke RFID *reader* melalui gelombang radio yang dipancarkan RFID *reader*.
2. Antena terdapat pada RFID tag (tag-antena) dan RFID *reader* (*reader* antena) atau (interogator)yang berfungsi mentransmisikan data dari chip RFID tag ke RFID *reader* melalui gelombang radio.
3. RFID *reader* adalah perangkat yang kompatibel dengan RFID tag. RFID *reader* akan memancarkan gelombang radio dan menginduksi RFID tag, kemudian RFID tag akan mengirim data ID dari antena yang terdapat pada rangkaian RFID tag melalui gelombang radio yang dipancarkan RFID *reader*.
4. Interface Softwareyang berfungsi untuk membaca data ID dari RFID *reader* dan mengolah data tersebut sehingga dapat digunakan menjadi password.

2.2.2.1 RFID Tag

RFID tag memiliki chip yang didalamnya dapat menyimpan data berupa nomor ID, transponder atau tag-antena yang berfungsi untuk mengirim data melalui gelombang radio yang dipancarkan RFID *reader* dan encapsulation atau bungkus yang berfungsi untuk melindungi chip agartidak mudah rusak (Ho Tien Dang, 2013:16). Berdasarkan catu dayanya, tag RFID dapat digolongkan menjadi dua :

1. Tag Aktif yaitu tag yang catu dayanya diperoleh dari baterai, sehingga akan mengurangi daya yang diperlukan oleh pembaca RFID dan tag dapat mengirimkan informasi dalam jarak yang lebih jauh. Kelemahan dari tipe

tag ini adalah harganya yang mahal dan ukurannya yang lebih besar karena lebih komplek. Semakin banyak fungsi yang dapat dilakukan oleh tag RFID maka rangkaiannya akan semakin komplek dan ukurannya akan semakin besar.

2. Tag Pasif yaitu tag yang catu dayanya diperoleh dari medan yang dihasilkan oleh pembaca RFID. Rangkaiannya lebih sederhana, harganya jauh lebih murah, ukurannya kecil, dan lebih ringan. Kelemahannya adalah tag hanya dapat mengirimkan informasi dalam jarak yang dekat dan pembaca RFID harus menyediakan daya tambahan untuk tag RFID. Tag RFID telah sering dipertimbangkan untuk digunakan sebagai barcode pada masa yang akan datang. Pembacaan informasi pada tag RFID tidak memerlukan kontak sama sekali. Karena kemampuan rangkaian terintegrasi yang modern, maka tag RFID dapat menyimpan jauh lebih banyak informasi dibandingkan dengan barcode (Bakhtiar, B. dan Susanti, R. Elektron, Vol.1: 63-64).

RFID tag terdiri dari 3 bagian yaitu:

1. Mikroprosesor

   Mikroprosesoradalah chip yang terletak dalam sebuahRFID tag yang berfungsi sebagai penyimpan data.

2. Metal Coil

   Metal Coil terbuat dari kawat alumunium yang berfungsi sebagai antena yang dapat beroperasi pada frekuensi 13,56 MHz. Apabila sebuah RFID tag masuk kedalam jangkauan *reader* maka antena akan mengirimkan data yang ada pada tag kepada *reader* terdekat.

3. Encapsulating

   Encapsulating adalah bahan yang berfungsi untuk melindungiRFID tag dan antena terbuat dari bahan plastik atau kaca [12].

2.2.3 OOP

Object Oriented Programing (OOP) merupakan paradigma pemrograman yang populer saat ini yang telah menggantikan teknik pemrograman berbasis prosedur. *Object Oriented Programing* yang berarti pula Pemrograman Beorientasi Objek sudah ditemukan sekitar tahun 1960 dan dikembangkan pada permulaan tahun 1970 [13].

Pemrograman Berorientasi Objek (*Object Oriented Programming*/OOP) merupakan pemrograman yang berorientasikan kepada objek, dimana semua data dan fungsi dibungkus dalam *class-class* atau *object-object*. Setiap objek dapat menerima pesan, memproses data, mengirim, menyimpan dan memanipulasi data. Beberapa object berinteraksi dengan saling memberikan informasi satu terhadap yang lainnya.

Pemrograman adalah mengharuskan pemahaman. Bahasa pemograman merupakan akses untuk perancangan dan pemrograman. Hanya bila pemograman dapat memahami gagasan-gagasan keberadaan fasilitas-fasilitas Dwibahasa pemrograman memungkinkan pemrogram menjadi mahir menggunakan bahasa pemrograman. Bahasa berorientasi objek adalah akas yang dapat dipergunakan oleh pemrogram secara bagus atau jelek.

Masing-masing objek harus berisikan informasi mengenai dirinya sendiri dan dapat dihubungkan dengan objek yang lain. Pemrograman berorientasi objek berbeda dengan pemrograman prosedural yang hanya menggunakan satu halaman kebawah untuk mengerjakan banyak perintah atau *statement*. Penggunaan pemrograman berorientasi objek sangat benyak sekali, contoh : java, php, perl, c#, cobol, dan lainnya.

Konsep Dasar Pemrograman Berorientasi Objek

Berikut ini adalah konsep-konsep dalam pemrograman berorientasi objek :

1. *Class*

Kelas (*Class*) merupakan penggambaran satu set objek yang memiliki atribut yang sama. Kelas mirip dengan tipe data ada pemrograman non objek, akan tetapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya. Kelas baru dapat dibentuk lebih spesifik dari kelas ada umumnya.kelas merupakan jantung dalam pemrograman berorientasi objek.

2. *Object*

   Objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan teknik yang efektif dalam menemukan cara yang tepat dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak. Orientasi objek merupakan teknik pemodelan sistem riil yang berbasis objek.

3. *Abstraction*

   Kemampuan sebuah program untuk melewati aspek informasi yang diolah adalah kemampuan untuk fokus pada inti permasalahan. Setiap objek dalam sistem melayani berbagai model dari pelaku abstrak yang dapat melakukan kerja, laporan dan perubahan serta berkomunikasi dengan objek lain dalam sistem, tanpa harus menampakkan kelebihan diterapkan.

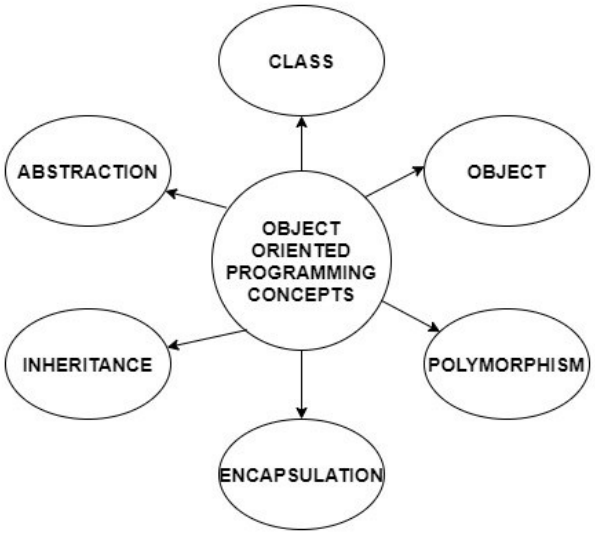
4. *Enkapsulasi*

   Enkapsulasi adalah proses memastikan pengguna sebuah objek tidak dapat menggantikan keadaan dari sebuah objek dengan cara yang tidak sesuai prosedur. Artinya, hanya metode yang terdapat dalam objek tersebut yang diberi izin untuk mengakses keadaan yang diinginkan. Setiap objek mengakses *interface* yang menyabutkan bagaimana objek lainnya dapat berintegrasi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

5. *Polimorfisme*

Polimorfisme merupakan suatu fungsionalitas yang diimplikasikan dengan berbagai cara yang berbeda. Pada program berorientasi objek, pembuat program dapat memiliki berbagai implementasi untuk sebagian fungsi tertentu.*Inheritas*

Konsep *inheritas* mempunyai fungsi mengatur polimorfise dan enkapsulasi dengan mengizinkan objek didefinisikan dan diciptakan dengan jenis khusus dari objek yang sudah ada. Objek-objek ini dapat membagi dan memperluas perilaku mereka tanpa mengimplementasikan perilaku tersebut.



Gambar 2.2 Konsep Dasar Pemrograman Berorientasi Objek

2.2.3.1 Uml

Unified Modelling Language (UML) yaitu suatu metode permodelan secara visual untuk sarana perancangan sistem berorientasi objek, atau definisi UML yaitu sebagai suatu bahasa yang sudah menjadi standar pada visualisasi, perancangan dan juga pendokumentasian sistem *software*. Saat ini UML sudah menjadi bahasa standar dalam penulisan blue print software [14].

UML dibuat oleh Grady Booch, James Rumbaugh, dan Ivar Jacobson di bawah bendera *Rational Software Corps*. UML menyediakan notasi-notasi yang membantu memodelkan sistem dari berbagai prespetktif. UML tidak hanya digunakan dalam

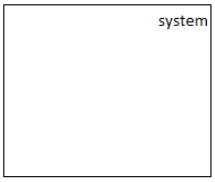
pemodelan perangkat lunak, namun hampir dalam semua bidang yang membutuhkan pemodelan. Tujuan dan fungsi dari penggunaan UML di antaranya:

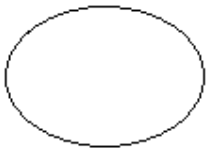
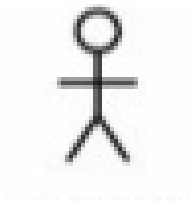
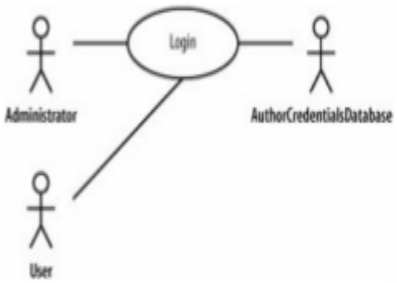
1. Dapat memberikan bahasa permodelan visual kepada pengguna dari berbagai macam pemerograman maupun proses rekayasa.
2. Dapat menyatukan praktek-praktek terbaik yang ada dalam permodelan.
3. Dapat memberikan model yang siap untuk digunakan, merupakan bahasa permodelan visual yang ekspresif untuk mengembangkan sistem dan untuk saling menukar model secara mudah.
4. Dapat berguna sebagai *blue print*, sebab sangat lengkap dan detail dalam perancangannya yang nantinya akan diketahui informasi yang detail mengenai koding suatu program.
5. Dapat memodelkan sistem yang berkonsep berorientasi objek, jadi tidak hanya digunakan untuk memodelkan perangkat lunak *(software)* saja.
6. Dapat menciptakan suatu bahasa permodelan yang nantinya dapat dipergunakan oleh manusia maupun oleh mesin.

2.2.3.2 Usecase

Use case diagram yaitu salah satu jenis diagram pada UML yang menggambarkan interaksi antara sistem dan aktor, *use case* diagram juga dapat mendeskripsikan tipe interaksi antara pemakai sistem dengan sistemnya.

Tabel 2.1 Simbol yang digunakan dalam use case

Nama Simbol	Gambar	Keterangan
Sistem		merupakan batasan (boundary) yang isinya merupakan berbagai use case yang digunakan.

Use Case		menunjukkan kegiatan/aktivitas yg bisa dilakukan.
Aktor		Entitas luar yang bisa memicu use case yang terhubung ke sistem, atau bisa dikatakan aktor adalah pengguna yang harus berinteraksi dengan sistem yang dibuat.
Relasi		Garis penghubung antara aktor dan use case maupun use case dengan use case yang lain, tanda include diantara garis yang terdapat antar use case menunjukkan bahwa use case tersebut membutuhkan use case lain yang sudah dihubungkan kepadanya, sedangkan tanda extend merupakan opsi alternatif bila ingin digunakan.

2.2.3.3 *Usecase Scenario*

Setiap *use case* diagram dilengkapi dengan skenario, skenario *use case* / *use case* skenario adalah alur jalannya proses *use case* dari sisi aktor dan *system*. Berikut adalah format tabel skenario *use case*.

Tabel 2.2 Keterangan dalam *use case scenario*

Deskripsi	Keterangan
Nama use case	menunjukkan kegiatan yang dilakukan.

Goal in context	hal yang bisa didapatkan oleh actor.
Description	penjelasan use case berdasarkan fungsionalitas.
Relasi use case	jika ada hubungan yang dibutuhkandengan use case lain.
precondition	kondisi yang diperlukan sebelum menjalankan use case.
post-condition	kondisi yang ada setelah use case dijalankan, ada dua kondisi yaitu berhasil/sukses dan gagal.
Aktor	memicu use case untuk dijalankan.
Trigger	digunakan oleh aktor agar bisa menjalankan use case yang dipilih.
Main flow	alur interaksi antara aktor dan sistem setelah menjalankan use case yang menunjukkan jika kondisinya benar.
extension	kebalikan dari main flow, yaitu alur yang menunjukkan jika kondisinya salah.

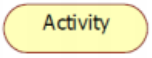
2.2.3.4 Activity Diagram

Activity diagram atau diagram aktivitas yaitu salah satu jenis diagram pada UML yang dapat memodelkan proses-proses apa saja yang terjadi pada sistem. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem [14]. 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 didefinisikan

2. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.

Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.

SIMBOL	SEMANTIC	FUNGSI
	Initial State	Penanda awal dari aktivitas
	Final State	Penanda akhir dari aktivitas
	Transisi	Menggambarkan alur antar aksi
	Aksi	Menggambarkan aksi yang ada pada satu aktivitas

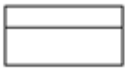
Gambar 2.3 Simbol *Activity* Diagram

2.2.3.5 Class Diagram

Class diagram yaitu salah satu jenis diagram pada UML yang digunakan untuk menampilkan kelas-kelas maupun paket-paket yang ada pada suatu sistem yang nantinya akan digunakan. Jadi diagram ini dapat memberikan sebuah gambaran mengenai sistem maupun relasi-relasi yang terdapat pada sistem tersebut [14]. Sususan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut :

1. Kelas Main yaitu kelas yang memiliki fungsi awal di eksekusi ketika sistem dijalankan
2. Kelas Interface kelas yang mendefinisikan dan mengatur tampilan ke pemakai, biasanya disebut kelas *boundaries.*

3. Kelas yang di ambil dari pendefinisian *use case* merupakan kelas yang menangani fungsi-fungsi yang harus diambil dari pendefinisian *use case*
4. Kelas Entitas merupakan kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

NO	GAMBAR	NAMA	KETERANGAN
1		*Generalization*	Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
2		*N-Ary Association*	Upaya untuk menghindari asosiasi dengan lebih dari 2 objek.
3		*Class*	Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama.
4		*Collaboration*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
5		*Realization*	Operasi yang benar-benar dilakukan oleh suatu objek.
6		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri
7		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya

Gambar 2.4 Simbol *Class Diagram*

2.2.3.6 Sequence Diagram

Sequence diagram yaitu salah satu jenis diagram pada UML yang menjelaskan interaksi objek yang berdasarkan urutan waktu, *sequence* diagram juga dapat

menggambarkan urutan atau tahapan yang harus dilakukan untuk dapat menghasilkan sesuatu seperti pada *use case diagram* [14].

Tabel 2.3 Simbol *Sequence Diagram*

Nama	Simbol	Keterangan
Actor		Menggambarkan seseorang atau sesuatu yang berinteraksi dengan sistem.
Boundary		Menggambarkan interaksi antara satu atau lebih yang menjadi penghubung antara actor dengan sistem.
Control		Menggambarkan "perilaku mengatur", mengkoordinasikan perilaku sistem dan dinamika dari suatu sistem, menangani tugas utama dan mengontrol alur kerja suatu sistem.
Entity		Menggambarkan informasi yang harus disimpan oleh sistem, entity juga memperlihatkan struktur data dari sebuah sistem.
Activation		Menggambarkan eksekusi terhadap obyek. Panjang kotak ini berbanding lurus dengan durasi aktivasi sebuah operasi.
Message		Menggambarkan message antar obyek, yang menunjukan urutan kejadian yang terjadi.
Message to Self		Looping terhadap obyek itu sendiri. Menggambarkan pesan atau hubungan obyek itu sendiri, yang menunjukan urutan kejadian yang terjadi.

Loop		Menunjukkan perilaku perulangan atau menggambarkan dari suatu kegiatan yang berulang-ulang

2.2.4 Sistem Terintegrasi

Integrasi adalah adanya saling keterkaitan antar sub sistem sehingga data dari satu sistem secara rutin dapat melintas, menuju atau diambil oleh satu atau lebih sistem yang lain [15].

Sistem integrasi (integrated system) merupakan sebuah rangkaian proses untuk menghubungkan beberapa sistem komputerisasi dan software aplikasi, baik secara fisik maupun secara fungsional. Sistem terintegrasi akan menggabungkan komponen sub-sub sistem ke dalam satu sistem dan menjamin fungsi-fungsi dari sub sistem tersebut sebagai satu kesatuan sistem.

Pengintegrasian sistem informasi merupakan salah satu konsep kunci dari sistem Informasi Manajemen. Berbagi sistem dapat saling berhubungan satu dengan yang lain dengan berbagai cara yang sesuai dengan keperluannya. Aliran informasi diantara sistem sangat bermanfaat bila data dalam file suatu sistem diperlukan juga oleh sistem yang lainnya, atau output suatu sistem menjadi *input* bagi sistem lainnya. Secara manual juga dapat dicapai suatu integrasi tertentu, misalnya data dari satu bagian dibawa kebagian lain, dan oleh petugas administrasi data tersebut digabung dengan data dari sistem yang lain. Jadi kalau secara manual maka derajat integrasinya menjadi tinggi.

Konsep Integrasi sistem adalah yaitu suatu konsep sistem yang dapat saling berhubungan satu dengan yang lain dengan berbagai cara yang sesuai dengan keperluan. Hal ini sangat bermanfaat bila suatu data dalam file suatu sistem diperlukan juga oleh sistem yang lainnya atau output sustu sistem menjadi *input* sistem lainnya [14].

2.2.5 Aplikasi

Aplikasi adalah program yang dibuat oleh pemakai yang ditujukan untuk melakukan suatu tugas khusus (Kadir, 2003).

Menurut Kadir (2008:3) program aplikasi adalah program siap pakai atau program yang direka untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Aplikasi juga diartikan sebagai penggunaan atau penerapan suatu konsep yang menjadi pokok pembahasan atau sebagai program komputer yang dibuat untuk menolong manusia dalam melaksanakan tugas tertentu. Aplikasi software yang dirancang untuk penggunaan praktisi khusus, klasifikasi luas ini dapat dibagi menjadi 2 (dua) yaitu:

a. Aplikasi software spesialis, program dengan dokumentasi tergabung yang dirancang untuk menjalankan tugas tertentu.
b. Aplikasi paket, suatu program dengan dokumentasi tergabung yang dirancang untuk jenis masalah tertentu.

Dari kedua pengertian di atas dapat disimpulkan bahwa aplikasi adalah sekumpulan perintah atau kode yang disusun secara sistematik untuk menjalankan suatu perintah yang diberikan oleh manusia melalui komponen atau hardware komputer yang digunakan oleh manusia dalam menjalankan program aplikasi, dengan demikian bisa membantu manusia untuk memberikan solusi dari apa yang diinginkan.

2.2.6 *Mobile Development*

Android Studio adalah sebuah *Integrated Development Environment* (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 [16].

Gambar 2.5 Logo Android Studio

Sebagai pengembangan dari Eclipse, Android Studio mempunyai banyak fitur-fitur baru dibandingkan dengan Eclipse IDE. Berbeda dengan Eclipse yang menggunakan Ant, Android Studio menggunakan Gradle sebagai build environment. Fitur-fitur lainnya adalah sebagai berikut :

1. Sistem Pembuatan berbasis Gradle yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat android
4. Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
6. Alat penguji dan kerangka kerja yang ekstensif
7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatbilitas ver si dan masalah-masalah lain
8. Dukungan C++ dan NDK
9. Dukungan bawaan untuk Google Cloud Platform, mempermudah pengintegrasian Google Cloud Messaging dan App Engine.

2.2.7 Firebase

Firebase adalah BaaS (*Backend as a Service*) yang saat ini dimiliki oleh Google. Firebase ini merupakan solusi yang ditawarkan oleh Google untuk mempermudah pekerjaan *Mobile Apps Developer*. Dengan adanya Firebase, apps developer bisa fokus mengembangkan aplikasi tanpa harus memberikan effort yang besar untuk urusan backend [17].

Beberapa fitur yang dimiliki oleh Firebase adalah sebagai berikut :

1. Firebase Analytics.
2. Firebase Cloud Messaging dan Notifications.
3. Firebase Authentication.
4. Firebase Remote Config.
5. Firebase Real Time Database.
6. Firebase Crash Reporting.

Dua fitur yang menarik adalah Firebase Remote Config dan Firebase Real Time Database. Secara sederhananya, Remote Config adalah fitur yang memungkinkan developer mengganti / mengubah beberapa konfigurasi aplikasi Android / iOS tanpa harus memberikan update aplikasi via Play Store / App Store. Salah satu konfigurasi yang bisa dimanipulasi adalah seperti warna / tema aplikasi.

Sedangkan Firebase *Real Time Database* adalah fitur yang memberikan sebuah NoSQL database yang bisa diakses secara *Real Time* oleh pengguna aplikasi. Dan hebatnya adalah aplikasi bisa menyimpan data secara lokal ketika tidak ada akses internet, kemudian melakukan sync data segera setelah mendapatkan akses internet.