BAB 2
RUANG LINGKUP PERUSAHAAN

Bagian ini berisi mengenai profil perusahaan beserta teori-teori yang menjadi pendukung dalam proses analisis dan implementasi Pembangunan *Cloud Computing* dan *Firebase Realtime* untuk Transaksi Pembayaran di Kedai Kopi Coger.

2.1 Profil Kedai Kopi Coger

Berikut ini merupakan profil dari Kedai Kopi Coger yang menjadi tempat penelitian penulis.

2.1.1 Sejarah Kedai Kopi Coger

Kedai Kopi Coger merupakan perusahaan yang bergerak di bidang kuliner, namun berfokus pada pengembangan kopi. Kedai Kopi Coger berdiri pada tahun 2016, di prakarsai oleh Steven Indra Wibowo yang sampai saat ini menjabat sebagai *Chief Excutive Officer* (CEO) sekaligus *Owner* Kedai Kopi Coger.

Kedai Kopi Coger merupakan perusahaan waralaba, saat ini terhitung ada sekitar 14 cabang yang tersebar ke berbagai kota di seluruh Indonesia. Selain itu, Kedai Kopi Coger memiliki konsep bisnis yang unik yaitu "Minum Sepuasnya Bayar Seikhlasnya", dimana pelanggan bisa memesan beberapa varian menu *Coffee Base* atau *Non-Milk* sepuasnya lalu bayar seikhlasnya sedangkan menu kopi dengan campuran susu ditarif dengan harga yang sangat terjangkau.

2.1.2 Tempat dan Kedudukan Perusahaan

Lokasi yang menjadi tempat penelitian penulis beralamatkan di Jl. Cisitu Indah VI No.184 A, Dago, Kecamatan Coblong, Kota Bandung, Jawa Barat 40135.

2.1.3 Cabang Kedai Kopi Coger

Berikut merupakan cabang dari Kedai Kopi Coger:

1. **Cabang Kota Bandung**

   a. **Coger Dago**

   Cabang ini merupakan *Main* cabang dari Kedai Kopi Coger yang terletak di Jl. Cisitu Indah VI No.184 A, Dago, Kecamatan Coblong, Kota Bandung, Jawa Barat 40135.

   b. **Coger Buah Batu**

   Cabang ini terletak di Jl. Terusan Buah Batu No.312, Kujangsari, Kec. Bandung Kidul, Kota Bandung, Jawa Barat 40287.

   c. **Coger Blaz Bluz (Kiara Condong)**

   Cabang ini terletak di Jl. Babakan Sari 1 No.11, Babakan Sari, Kec. Kiaracondong, Kota Bandung, Jawa Barat 40283.

2. **Cabang Kota Depok**

   a. **Coger Depok 1**

   Cabang ini berlokasi di Jl. Raya Cipayung Jaya No.16, Cipayung Jaya, Kec. Cipayung, Kota Depok, Jawa Barat 16437.

   b. **Coger Palanta (Depok 2)**

   Cabang ini terletak di Perumahan Sawangan Permai, Kelurahan Pasir Putih Blok R2/A2, Sawangan Baru, Kec. Sawangan, Kota Depok, Jawa Barat 16519.

3. **Cabang Kota Bekasi**

   a. **Coger Bintara**

   Cabang ini berlokasi di Jl. Bintara Jaya No.21, RW.5, Pd. Kopi, Kec. Bekasi Baru, Kota Bekasi, Jawa Barat 17136.

b. Coger Café THI

Cabang ini terletak di Ruko Asia Tropis Blok AT12 No 10A, Pusaka Rakyat, Tarumajaya, Bekasi, West Java 17213.

c. Coger Tarumajaya

Cabang ini terletak di Ruko Green Victoria Blok VG15 No.35, Sagara Makmur, Tarumajaya, Bekasi, West Java 17213.

4. Cabang Kota Bogor

Cabang Kota Bogor terletak di Perumahan Ambar Telaga Residen Blok D5, No. 01, Kota Bogor, Jawa Barat.

5. Cabang Kota Garut

Cabang ini terletak di Jl. Ciledug No.73, Regol, Kec. Garut Kota, Kabupaten Garut, Jawa Barat 44114.

6. Cabang Kota Cikarang

Cabang ini terletak di Jalan Kampung Tegal Gede Pandan No.26, RT.13/RW.6, Mekarmukti, Cikarang Utara, Bekasi, Jawa Barat 17530.

7. Cabang Kota Palembang

Cabang ini terletak di Jl. May Zen No.19, Sei Selincah, Kec. Kalidoni, Kota Palembang, Sumatera Selatan 30161.

8. Cabang Kota Surabaya

Cabang ini terletak di Jl. Sidopurno, Sidopurno 2, Sidokepung, Kec. Buduran, Kabupaten Sidoarjo, Jawa Timur 61252.

9. Cabang Kota Malang

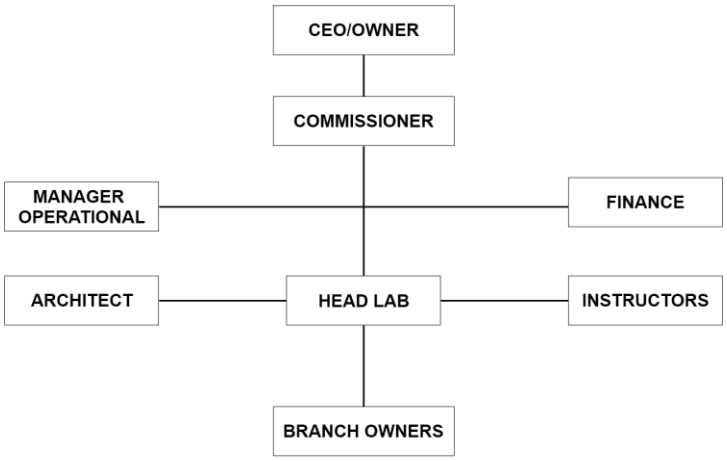
Cabang ini terletak di Jl. Kerto Rahayu No.67a, Ketawanggede, Kec. Lowokwaru, Kota Malang, Jawa Timur 65145.

2.1.4 Logo Perusahaan

Pada Gambar 2.1 merupakan logo dari Kedai Kopi Coger.

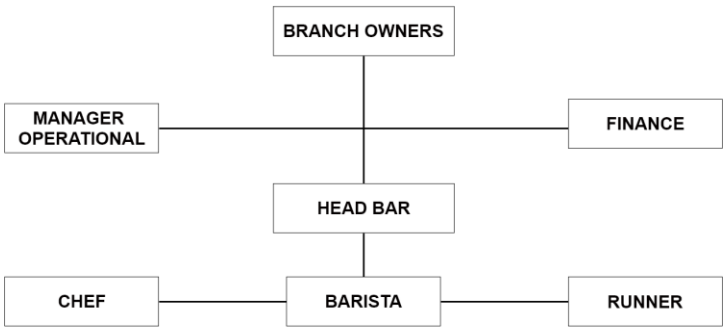
Gambar 2.1 Logo Kedai Kopi Coger

2.1.5 Struktur Organisasi Perusahaan



Gambar 2.2 Struktur Organisasi

2.1.6 Struktur Organisasi Cabang



Gambar 2.3 Struktur Organisasi Cabang

2.2 Tinjauan Pustaka

Untuk mendukung dalam pembuatan laporan ini, maka perlu dikemukakan teori-teori yang berkaitan dengan sistem operasi yang dibuat. Hal ini mencakup

Android, Java, Firebase, JSON, XML, Internet, UML, *Usecase*, *Usecase Scenario*, *Activity Diagram, Class Diagram, Sequence Diagram*.

2.2.1 Teknologi Informasi

Teknologi Informasi banyak digunakan dalam melakukan pengelolaan serta menejemen di suatu organisasi atau institusi. Teknologi Informasi digunakan untuk mencapai suatu tujuan tertentu [2].

2.2.2 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis Linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Android merupakan generasi baru *platform mobile*, *platform* yang memberikan pengembang untuk melakukan pengembangan sesuai dengan yang diharapkannya.

Sistem operasi yang mendasari Android dilisensikan dibawah GNU, *General Public Lisence* Versi 2 (GPLv2), yang sering dikenal dengan istilah "copyleft" lisensi dimana setiap perbaikan pihak ketiga harus terus jatuh di bawah terms. Android didistribusikan di bawah Lisensi *Apache Software* (ASL/Apache2), yang memungkinkan untuk distribusi kedua dan seterusnya. Komersialisasi pengembang (produsen handset khususnya) dapat memilih untuk meningkatkan *platform* tanpa harus memberikan perbaikan mereka ke masyarakat *open source* .

Sebaliknya, pengembang dapat keuntungan dari perangkat tambahan seperti perbaikan dan mendistribusikan ulang pekerjaan mereka di bawah lisensi apapun yang mereka inginkan. Pengembang aplikasi Android diperbolehkan untuk mendistribusikan aplikasi mereka di bawah skema lisensi apapun yang mereka inginkan [3].

2.2.3 Point Of Sale

Point Of Sale merupakan sebuah sistem yang berfungsi untuk membantu proses transaksi dengan *customer*. Sistem ini banyak dikembangkan guna

meningkatkan fleksibilitas dan efisiensi dalam proses transaksi dengan *customer* [4].

2.2.4 Java

Java adalah sebuah bahasa pemrograman yang diciptakan oleh James Gosling, seorang *developer* dari *Sun Microsystem* pada tahun 1991. Selanjutnya Java dikembangkan *Sun Microsystem* dan banyak digunakan untuk menciptakan *Executable Content* yang dapat didistribusikan melalui *network* [2].

Java adalah bahasa pemrograman *Object-Oriented* dengan unsur-unsur seperti bahasa C++ dan bahasa-bahasa lainnya yang memiliki *libraries* yang cocok untuk lingkungan internet. Java dapat melakukan banyak hal dalam melakukan pemrograman, seperti membuat animasi halaman web, pemrograman Java untuk Ponsel dan aplikasi interaktif. Java juga dapat digunakan untuk handphone, internet dan lain-lain [2].

2.2.5 *Firebase*

Firebase Realtime Database adalah *database* yang di-*host* di *cloud*. Data disimpan sebagai JSON dan disinkronkan secara *realtime* ke setiap klien yang terhubung. Ketika Anda membuat aplikasi lintas-platform dengan SDK Android, iOS, dan JavaScript, semua klien akan berbagi sebuah *instance Realtime Database* dan menerima *update data* terbaru secara otomatis [3].

2.2.6 *JavaScript Object Notation (*JSON)

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (generate) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 - Desember 1999 [4].

JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python

dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data [4].

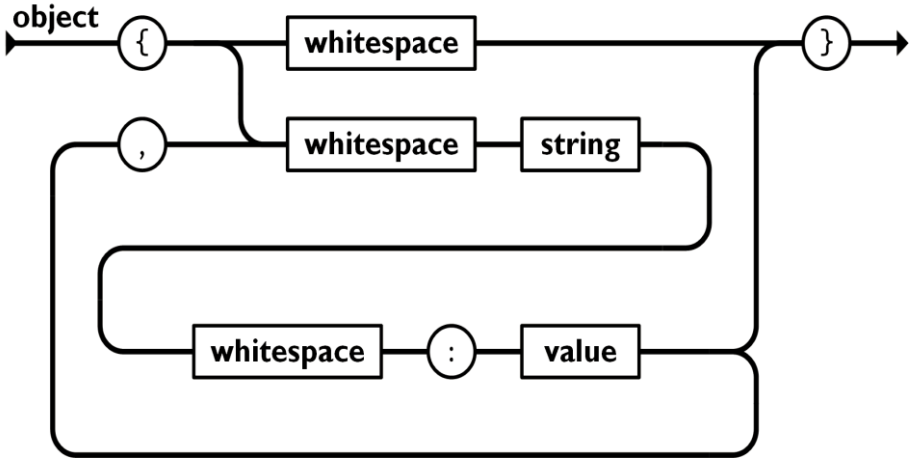
JSON terbuat dari 2 struktur, yaitu :

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (object), rekaman (record), struktur (struct), kamus (dictionary), tabel hash (hash table), daftar berkunci (keyed list), atau associative array [4].
2. Daftar nilai terurutkan (an ordered list of values). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (array), vektor (vector), daftar (list), atau urutan (sequence) [4].

Berikut merupakan bentuk diagram pada JSON :

1. *Object*

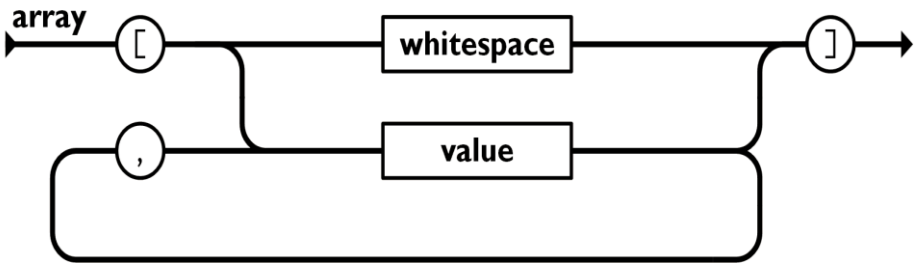
   Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan {kurung kurawal buka dan diakhiri dengan } kurung kurawal tutup. Setiap nama diikuti dengan :titik dua dan setiap pasangan nama/nilai dipisahkan oleh ,koma.



Gambar 2.4 Diagram Objek pada JSON

2. *Array*

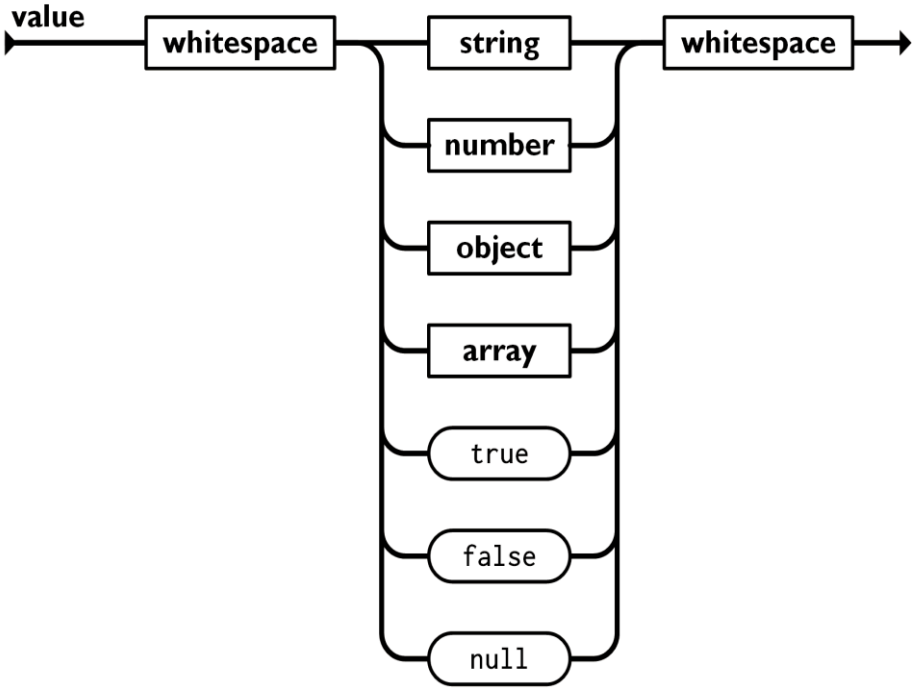
   Array adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [kurung kotak buka dan diakhiri dengan]kurung kotak tutup. Setiap nilai dipisahkan oleh ,koma.



Gambar 2.5 Diagram *Array* pada JSON

3. *Value*

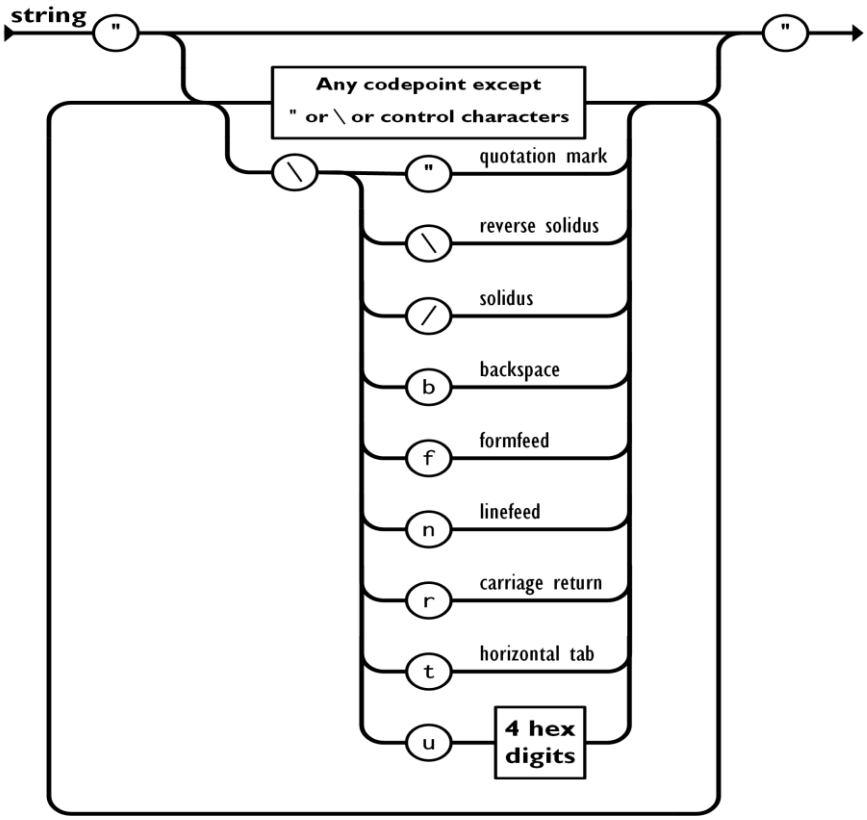
Value dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau *true* atau *false* atau *null*, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.



Gambar 2.6 Diagram *Value* pada JSON

4. *String*

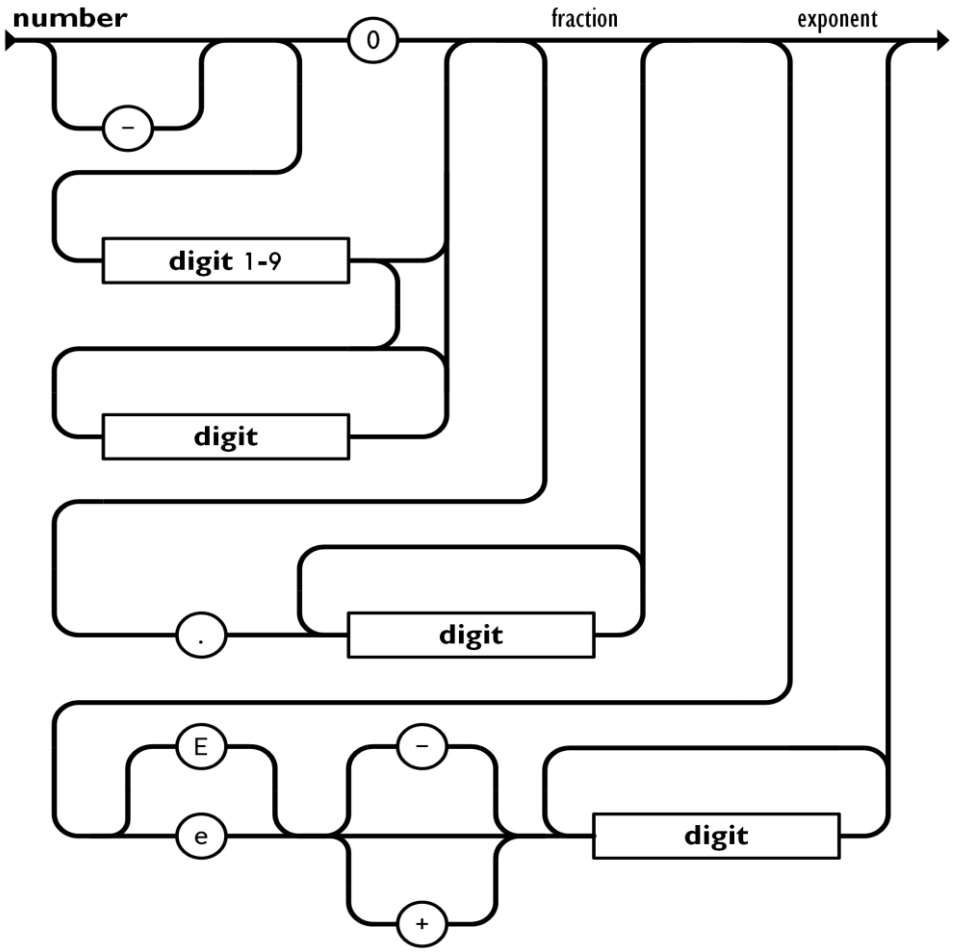
String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.



Gambar 2.7 Diagram *String* pada JSON

5. *Number*

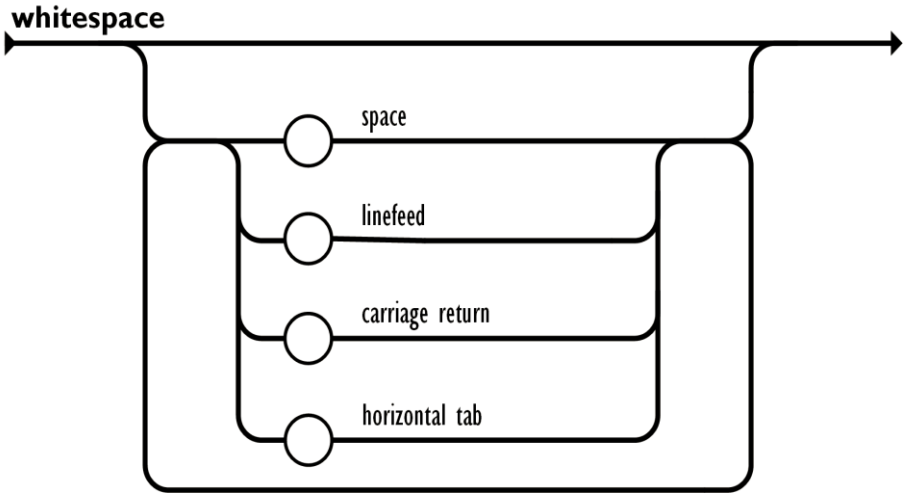
Number adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.



Gambar 2.8 Diagram *Number* pada JSON

6. *Whitespace*

Whitespace dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil encoding yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.



Gambar 2.9 Diagram *Whitespace* pada JSON

2.2.7 *Extensible Markup Language* (XML)

Extensible Markup Language (XML) termasuk dalam keluarga *Markup Language* seperti halnya HTML atau WML. Namun dengan XML, kita bisa mendefinisikan *custom tag* sendiri.

Pada XML, untuk menjelaskan apakah sebuah tag itu valid atau tidak, kita bisa menggunakan DTD (*Document Type Definition*). DTD juga menjelaskan struktur dari dokumen XML. Kemudian untuk menjelaskan arti-arti dari tag yang dibuat, dapat digunakan *stylesheet*, yaitu XSL (*Extensible Stylesheet Language*). XSL terdiri dari tiga bagian dan tiap bagian ada rekomendasinya sendiri-sendiri dari W3C, yaitu : Xpath (XML *Path Language*), bahasa untuk menunjukkan sebuah bagian dari dokumen XML. XSLT (XSL *Transformation*), bahasa untuk mentransformasikan sebuah dokumen XML ke dokumen lain. XSL, yaitu XSLT ditambah dengan kumpulan penjelasan mengenai *formatting objects* dan *formatting properties* [5].

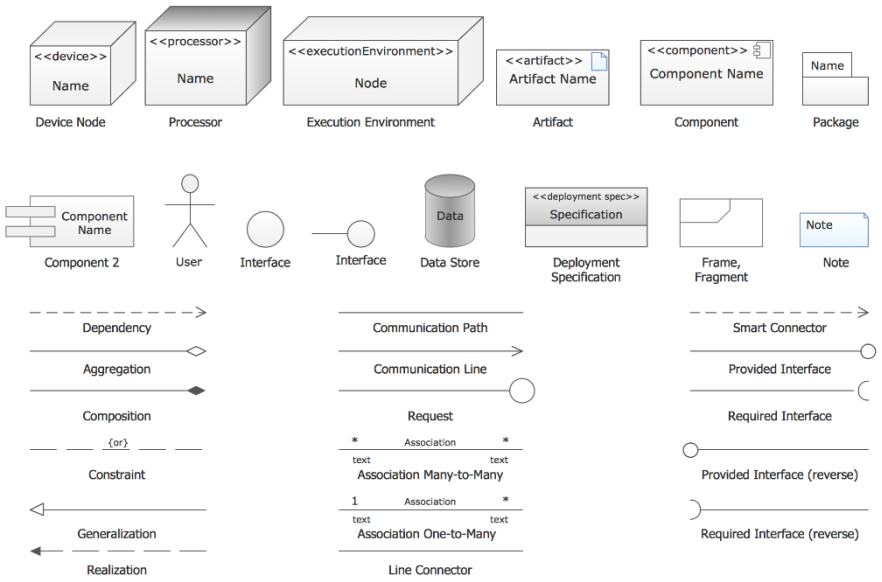
2.2.8 Internet

Internet adalah suatu jaringan komunikasi yang menghubungkan satu media elektonik dengan media yang lainnya. Standar teknologi pendukung yang dipakai secara global adalah Transmission Control Protocol atau Internet Protocol Suite (disingkat sebagai istilah TCP/IP). TCP/IP ini

merupakan protokol pertukaran paket (dalam istilah asingnya Switching Communication Protocol) yang bisa digunakan untuk miliaran lebih pengguna yang ada di dunia. Sementara itu, istilah "internetworking" berarti cara/prosesnya dalam menghubungkan rangkaian internet beserta penerapan aturannya yang telah disebutkan sebelumnya [6].

2.2.9 UML

Unified Modelling Language atau 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 [7].

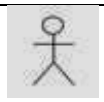
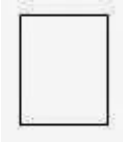


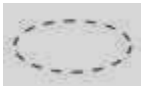
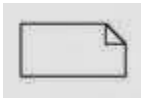
Gambar 2.10 UML Notations

2.2.10 Usecase

Usecase adalah sebuah teknik yang digunakan dalam pengembangan sebuah software atau sistem informasi untuk menangkap kebutuhan fungsional dari sistem yang bersangkutan, Use Case menjelaskan interaksi yang terjadi antara 'aktor'—inisiator dari interaksi sistem itu sendiri dengan sistem yang ada, sebuah Use Case direpresentasikan dengan urutan langkah yang sederhana [8].

Tabel 2.1 Usecase Notations

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
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.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang

			menghasilkan suatu hasil yang terukur bagi suatu actor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

2.2.11 *Usecase Scenario*

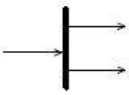
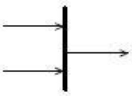
Usecase scenario adalah alur jalannya proses *use case* dari sisi aktor dan sistem. *Skenario use case* dibuat per use case terkecil, misalkan untuk generalisasi maka *scenario* yang dibuat adalah *use case* yang lebih khusus. Skenario normal adalah *scenario* bila sistem berjalan normal tanpa terjadi kesalahan atau error. Sedangkan skenario alternatif adalah *scenario* bila sistem tidak berjalan normal atau mengalami error. Skenario normal dan skenario alternatif dapat berjumlah lebih dari satu. Alur skenario inilah yang nantinya menjadi landasan pembuatan *sequence diagram* / diagram sekuen [9].

Use case description detail	Apa rincian maksud dan mengapa itu berguna
Related Requirements	Beberapa indikasi mengenai persyaratan apa yang perlu dipenuhi berdasarkan kasus kegunaan ini baik sebagian atau seluruhnya.
Goal In Context	Keberadaan use case di dalam sistem dan mengapa use case ini penting.
Preconditions	Keadaan sebelum use case dapat dieksekusi.
Successful End Condition	Bagaimana kondisi sistem seharusnya jika use case dieksekusi dengan sukses.
Failed End Condition	Bagaimana kondisi sistem jika use case gagal dieksekusi dengan sukses.
Primary	Aktor-aktor utama yang berpartisipasi dalam use case. Seringkali termasuk aktor yang memicu atau langsung menerima informasi dari hasil eksekusi use case.
Secondary Actors	Aktor yang ikut berpartisipasi tetapi bukan pemain utama dalam eksekusi use case.
Trigger	Peristiwa yang dipicu oleh aktor yang menyebabkan use case dieksekusi.
Main Flow	Tempat untuk menggambarkan masing-masing langkah penting dalam eksekusi normal use case.
Extentions	Deskripsi langkah-langkah alternatif dari yang dijelaskan dalam aliran utama.

Gambar 2.11 Usecase Scenario

2.2.12 *Activity Diagram*

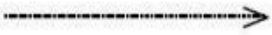
Activivity diagram adalah diagram yang menggambarkan *worlflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem [10].

Simbol	Keterangan
	Start Point
	End Point
	Activities
	Fork (Percabangan)
	Join (Penggabungan)
	Decision
Swimlane	Sebuah cara untuk mengelompokkan activity berdasarkan Actor (mengelompokkan activity dalam sebuah urutan yang sama)

Gambar 2.12 Activity Diagram

2.2.13 *Class Diagram*

Class diagram adalah diagram yang menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. kelas memiliki 3 bagian utama yaitu *attribute*, *operation*, dan *name*. kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem [11].

asosiasi / *association*	relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
asosiasi berarah / *directed association* 	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 / *dependency* 	relasi antar kelas dengan makna kebergantungan antar kelas
agregasi / *aggregation* 	relasi antar kelas dengan makna semua-bagian (*whole-part*)

Gambar 2.13 Class Diagram

2.2.14 *Sequence Diagram*

Sequence diagram adalah diagram yang menggambarkan kelakuan objek pada usecase dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *usecase* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Banyaknya diagram sekuen yang harus digambar sebanyak *usecase* yang memiliki proses sendiri atau yang penting semua *usecase* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *usecase* yang ada maka diagram sekuen yang dibuat semakin banyak [12].

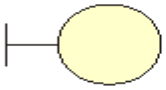
a. An Actor

Menggambarkan orang yang sedang berinteraksi dengan sistem

b. Entity Class

Menggambarkan hubungan kegiatan yang akan dilakukan

c. Boundary Class



Mengganbarkan sebuah penggambaran dari form

d. Control Class

Menggambarkan penghubung antara boundary dengan tabel

e. A focus Of Control & A life line

Menggambarkan tempat mulai dan berakhirnya sebuah massage

f. A massage

Menggambarkan Pengiriman Pesan

Gambar 2.14 Sequence Diagram