BAB II

TINJAUAN PUSTAKA

2.1 Teori Umum

2.1.1 Pengertian Komputer

Kadir (2018:2) mengemukakan bahwa, "Komputer merupakan peralatan elektronik yang bermanfaat ,melaksanakan berbagai pekerjaan yang dilakukan oleh manusia ".

Selain pengertian diatas, adapun pengertian yang dikemukakan oleh Ikhsan dan Kurniawan (2015:13), "Komputer adalah sebuah mesin hitung elektronik yang secara cepat menerima informasi masukan digital dan mengolah informasi tersebut menurut seperangkat instruksi yang tersimpan dalam komputer tersebut dan menghasilkan keluaran informasi yang dihasilkan setelah diolah."

Berdasarkan pendapat di atas, dapat disimpulkan bahwa definisi dari komputer adalah sebuah alat elektronik yang dapat memproses data digital dan informasi yang bermanfaat bagi *user* sehingga dapat menghasilkan keluaran informasi yang dihasilkan setelah diolah.

2.1.2 Pengertian Perangkat Lunak

Menurut Sukamto dan Shalahuddin (2018:2), "Perangkat Lunak (*Software)* adalah program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaan (*user manual*)".

Sedangkan Swara dan Febriadi (2018:8), "Perangkat lunak merupakan seluruh perintah yang digunakan untuk memproses informasi".

Berdasarkan pendapat diatas, dapat disimpulkan bahwa perangkat lunak merupakan cara penggunakan yang ditujukan kepada komputer sesuai kebutuhan pemakai yang digunakan untuk memproses informasi.

2.1.3 Pengertian Data

Menurut Kristanto (2018:8) menyatakan bahwa, "Data merupakan bentuk yang belum dapat memberikan manfaat yang besar bagi penerimanya, sehingga perlu suatu model yang nantinya akan dikelompokkan dan diproses untuk menghasilkan informasi".

Selain itu menurut Rusdiana dan Irfan (2016:68), "Data adalah fakta-fakta mentah yang harus dikelola untuk menghasilkan informasi yang memiliki arti bagi suatu organisasi atau perusahaan.

Berdasarkan pendapat di atas, dapat disimpulkan bahwa definisi dari data adalah fakta yang belum dapat memberikan manfaat yang besar bagi penerimanya, sehingga perlu suatu model untuk menghasilkan informasi yang memiliki arti bagi suatu organisasi atau perusahaan.

2.1.4. Metode Pengembangan Sistem

Penelitian ini menggunakan metode pengembangan perangkat lunak dengan RUP (*Rational Unified Process*). Menurut Sukamto dan Shalahuddin (2016:125), "RUP (*Rational Unified Process*) adalah pendekatan pengembangan perangkat lunak yang dilakukan berulang-ulang (*iterative*), fokus pada arsitektur (*architecture-centric*), lebih diarahkan berdasarkan penggunaan kasus (*use case driven*)". Adapun tahap-tahap (*fase*) dalam metode pengembangan RUP menurut Sukamto dan Shalahuddin (2018:128-131) adalah sebagai berikut:

1. *Inception* (permulaan)

   Tahap ini lebih pada memodelkan proses bisnis yang dibutuhkan (*bussiness modeling*) dan mendefinisikan kebutuhan akan sistem yang akan dibuat (*requirements*).

2. *Elaboration* (perluasan/perencanaan)

   Tahap ini lebih difokuskan pada perencanaan arsitektur sistem. Tahap ini juga dapat mendeteksi apakah arsitektur sistem yang diinginkan dapat dibuat atau tidak. Mendeteksi resiko yang mungkin terjadi dari arsitektur yang dibuat.

Tahap ini lebih pada analisis dan desain sistem serta implementasi sistem yang fokus pada purwarupa sistem (*prototype*).

3. *Construction* (kontruksi)

   Tahap ini fokus pada pengembangan komponen dan fitur-fitur sistem. Tahap ini lebih pada implementasi dan pengujian sistem yang fokus pada implementasi perangkat lunak pada kode program. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/tonggak kemampuan operasional awal.

4. *Transition* (transisi)

   Tahap ini lebih pada deployment atau instalasi sistem agar dapat dimengerti oleh user. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/tongggak kemampuan operasional awal. Aktifitas pada tahap ini termasuk pada pelatihan user, pemeliharaan dan pengujian sistem apakah sudah memenuhi harapan user.

2.2 Teori Khusus

2.2.1 Pengertian Kamus Data

Sukamto dan Shalahuddin (2018:73) menjelaskan, “Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukkan (input) dan keluaran (ouput) dapat dipahami secara umum (memiliki standar cara penulisan).” Kamus data memiliki beberapa simbol sebagai berikut :

Tabel 2.1 Simbol-simbol Kamus Data

No	Simbol	Keterangan
1.	=	Disusun atau terdiri dari
2.	+	Dan
3.	[|]	Baik...atau...
4.	{ }^{n}	n kali diulang/bernilai banyak
5.	( )	Data opsional
6.	*..*	Batas komentar

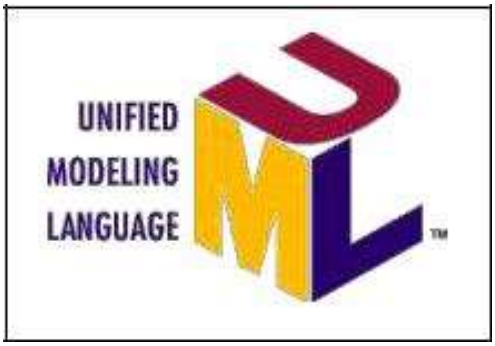
Sumber: Sukamto dan Shalahuddin (2016:74)

2.2.2 Pengertian *Unified Modeling Language* (UML)

Sukamto dan Shalahuddin (2018:133), menjelaskan tentang pengertian *Unified Modeling Language* sebagai berikut :

"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 menyediakan serangkaian gambar dan diagram yang sangat baik. Beberapa diagram memfokuskan diri pada ketangguhan teori *object-oriented* dan sebagian lagi memfokuskan pada detail rancangan dan konstruksi. Semua dimaksudkan sebagai sarana komunikasi antar *team programmer* maupun dengan pengguna.

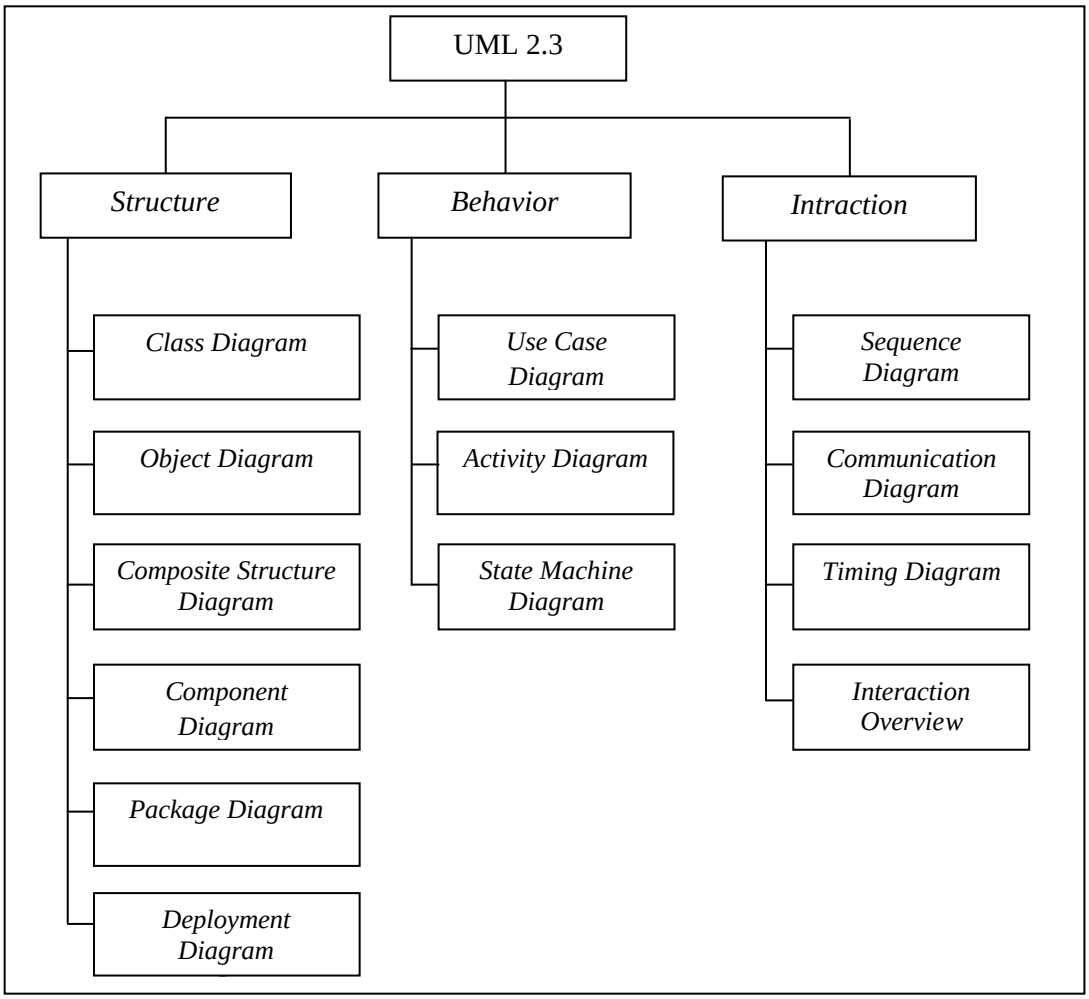


Sumber:wikipedia.org/undified_modeling_language

Gambar 2.1. Tampilan Logo UML

2.2.3 Kategori *Unified Modeling Language* (UML)

Menurut Sukamto dan Shalahuddin (2018:140), "Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori". Pembagian kategori dan macam-macam diagram Menurut Sukamto dan Shalahuddin tersebut dapat dilihat pada gambar dibawah:



Gambar 2.2 Kategori dan Macam-macam Diagram UML

Penjelasan singkat dari pembagian kategori pada diagram UML menurut Sukamto dan Shalahuddin (2018:141) :

1) *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2) *Behavior diagrams* yaitu kumpulam diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahana yang terjadi pada sebuah sistem.

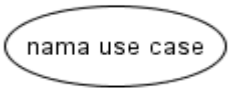
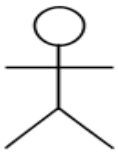
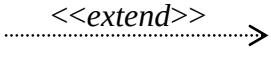
3) *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

2.2.4 Jenis-Jenis Diagram UML

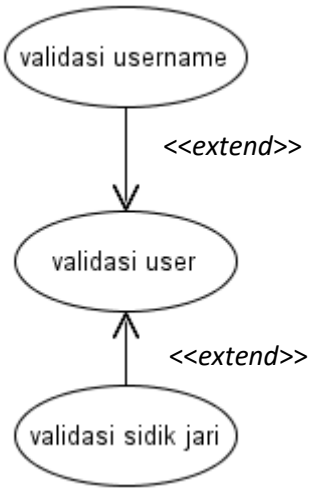
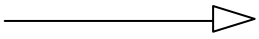
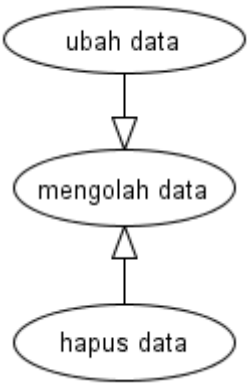
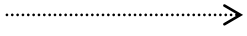
2.2.4.1 Pengertian *Use case* Diagram

Menurut Sukamto dan Shalahuddin (2016:155), *"Use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem." Adapun simbol-simbol yang digunakan dalam *use case* adalah sebagai berikut:

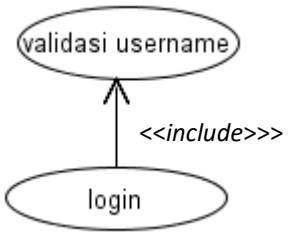
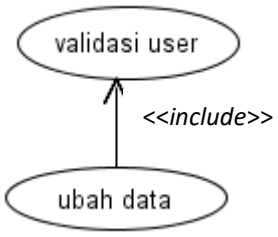
Tabel 2.2 Simbol-simbol *Use case* Diagram

No	Simbol	Deskripsi
1.	*Use case* 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal-awal frase nama *use case.*
2.	aktor / *actor* 	orang, proses, atau siatem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama actor.
3.	asosiasi / *association* _____	komunikasi antar aktor dan *use case* yang berpartisioasi pada *use case* atau *use case* memiliki ingeraksi dengan actor.
4.	ekstensi / *extend* 	relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang di tambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misalnya

Tabel 2.2 (lanjutan)

No	Simbol	Deskripsi
		 arah panah mengarah pada *use case* yang ditambahkan; biasanya *use case* yang menjadi *extend*-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya
5.	Generalisasi / *generalization* 	hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya,  misalnya: arah panah mengarah pada *use case* yang menjadi generalisasinya (umum).
6.	menggunakan / include / uses  <<*include*>>	relasi tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini ada dua sudut pandang yang cukup besar mengenai include di *use case*: • *Include* berarti *use case* yang ditambahkan

Tabel 2.2 (lanjutan)

No	Simbol	Deskripsi
	<<uses>>	akan selalu di panggil saat *use case* tambahan dijalankan, misalnya pada kasus berikut:  • *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang di tambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut:  kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

Sumber: Sukamto dan Shalahuddin (2016:156)

Ada dua hal utama pada *use case* yaitu:

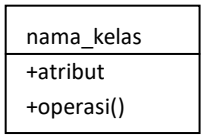
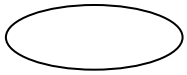
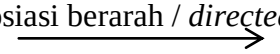
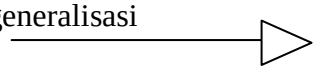
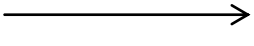
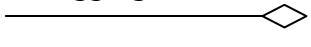
1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, Jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang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

2.2.4.2 Pengertian *Class Diagram*

Menurut Sukamto dan Shalahuddin (2018:141), *"Class Diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron."

Adapun simbol-simbol yang digunakan dalam *class* diagram adalah sebagai berikut:

Tabel 2.3 Simbol-simbol *Class* Diagram

No	Simbol	Deskripsi
1.	kelas 	Kelas pada struktur sistem
2	antarmuka / interface  nama_interface	Sama dengan konsep interface dalam pemrograman berorientasi objek
3.	asosiasi / association 	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai *multiplicity*
4.	asosiasi berarah / *directed assosiation* 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
5.	generalisasi 	Relasi antarkelas dengan makna generalisasi – spesialisasi (umum khusus)
6.	kebergantungan / *dependency* 	Relasi antarkelas dengan makna kebergantungan antar kelas
7.	agregasi / *aggregation* 	Relasi antarkelas dengan makna semua-bagian (*whole-part*)

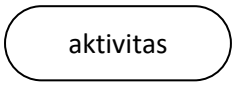
Sumber: Sukamto dan Shalahuddin

2.2.4.3 Pengertian *Activity Diagram*

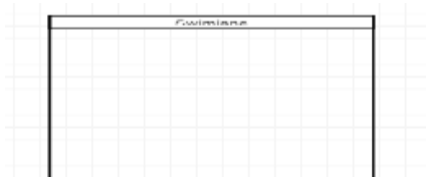
Menurut Sukamto dan Shalahuddin (2018:161), "A*ctivity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem."

Adapun simbol-simbol yang digunakan dalam *activity diagram* adalah sebagai berikut:

Tabel 2.4 Simbol-simbol *Activity Diagram*

No	Simbol	Deskripsi
1.	Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
3.	Percabangan / *decision* 	Asosiasi percabangan di mana jika ada pilihan aktivitas lebih dari satu
4.	Penggabungan / *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
5.	Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir

Tabel 2.4 (lanjutan)

No	Simbol	Deskripsi
6.	Swimlane  atau 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber: Rosa dan Shalahuddin (2016:162)

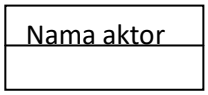
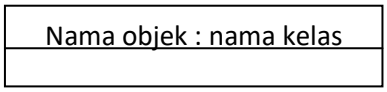
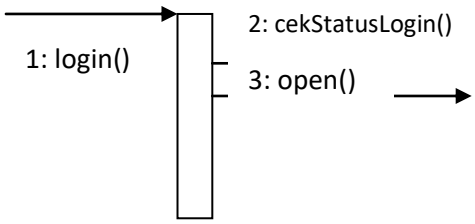
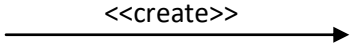
2.2.4.4 Pengertian *Sequence Diagram*

"Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan 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 beserta metode-metode yang dimiliki kelas yang diinstansikan menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada use case" (Sukamto dan Shalahuddin, 2016:165). "Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu" (Safaat, 2015:33-34).

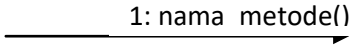
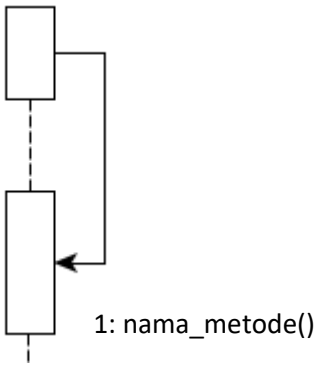
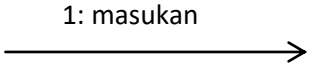
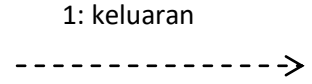
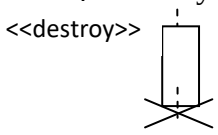
Dapat penulis simpulkan bahwa Sequence diagram adalah penggambaran skenario dari sebuah objek yang ada pada use case yang meliputi rangkaian langkah-langkah aktivitas dari objek berdasarkan waktu hidup objek dan pesan-pesan yang diterima maupun yang dikirimkan objek kepada objek lainnya.

Berikut simbol-simbol pada Sequence Diagram :

Tabel 2.5 Simbol-simbol pada Sequence Diagram

No	Simbol	Deskripsi
1.	Actor  atau  tanpa waktu aktif	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama actor
2.	Garis hidup / *lifeline* 	menyatakan kehidupan suatu objek
3.	Objek  	menyatakan objek yang berinteraksi pesan
4.	Waktu aktif 	menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  maka cek Status Login () dan open() dilakukan di dalam metode login() aktor tidak memiliki waktu aktif
5.	Pesan tipe create  	menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat

Tabel 2.5 (lanjutan)

No	Simbol	Deskripsi
6.	Pesan tipe call 	menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi
7.	Pesan tipe send 	menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi
8.	Pesan tipe return 	menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian
9.	Pesan tipe destroy 	menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy

Sumber: Rosa dan Shalahuddin (2013:165-167)6

2.3 Teori Judul

2.3.1 Pengertian Aplikasi

Budiharto dalam Ghazali (2016:18), "Aplikasi merupakan program yang dapat berjalan di komputer tersendiri (stand alone computer), dari mulai program yang simple sampai dengan program besar dan rumit".

2.3.2 Pengertian E-Commerce

Menurut Jony Wong (2010 : 33) pengertian dari electronic commerce adalah pembelian, penjualan dan pemasaran barang serta jasa melalui system elektronik. Seperti radio, televisi dan jaringan computer atau internet.

2.3.3 Pengertian Warung Kelontong

Pengetian warung kelontong dalam Kamus Besar Bahasa Indonesia adalah

Warung : Toko kecil tempat menjual barang kelontong atau makanan.

Kelontong : alat kelentungan yang selalu dibunyikan oleh penjaja barang dagangan untuk menarik perhatian pembeli dan barang-barang untuk keperluan sehari-hari.

Warung kelontong yaitu warung yang menyediakan kebutuhan rumah tangga seperti sembilan bahan pokok (sembako), makanan dan barang rumah tangga. Warung ini ditemukan berdampingan dengan pemilik rumah yang tidak jauh dengan masayarakat seperti perkamapungan, perumahan dan yang sering ditemui didalam gang.

2.3.4 Pengertian Wilayah

Wilayah dalam kamus besar bahasa indonesia adalah daerah (kekuasaan, pemerintahan, pengawasan, dan sebagainya); lingkungan daerah (provinsi, kabupaten, kecamatan);

2.3.5 Pengertian Android

Menurut Safaat (2015:1), "Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencankup sistem operasi, *middleware* dan aplikasi".

2.4 Teori Program

2.4.1 Pengertian Basis Data

Sukamto dan Shalahuddin (2013:43), "Basis data adalah sistem komputerisasi yang tujuan utamanya adalah memelihara data yang sudah ada yang diolah atau informasi dan membuat informasi tersedia saat dibutuhkan."

Kristanto (2018:79), "Basis data adalah kumpulan data, yang dapat digambarkan sebagai aktifitas dari satu atau lebih organisasi yang berelasi."

Kesimpulannya, Basis data adalah sistem komputerisasi sebagai aktifitas dari satu atau lebih organisasi.

2.4.2 Pengertian Firebase

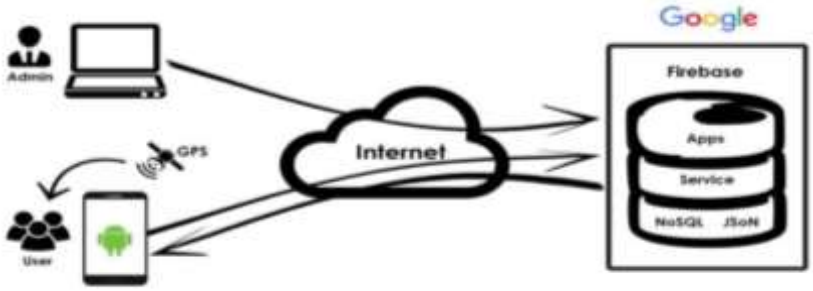
Sumber:firebase.google.com

Gambar 2.2 Logo Firebase

Firebase memiliki produk utama, yaitu menyediakan database realtime dan backend sebagai layanan (Backend as a Service). Layanan ini menyediakan pengembang aplikasi API yang memungkinkan aplikasi data yang akan disinkronisasi di klien dan disimpan di cloud Firebase ini. Firebase menyediakan library untuk berbagai client platform yang memungkinkan integrasi dengan Android, iOS, JavaScript, Java, Objective-C dan Node aplikasi Js dan dapat juga disebut sebagai layanan DbaaS (Database as a Service) dengan konsep realtime.

Firebase digunakan untuk mempermudah dalam penambahan fitur-fitur yang akan dibangun oleh developer.

Dalam Gambar 2.3 ditunjukkan contoh arsitektur sistem Firebase dengan Android.



Sumber:firebase.google.com

Gambar 2.3 Arsitektur Sistem Firebase

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. Meskipun basis data menggunakan JSON tree, data yang tersimpan dalam basis data bisa diwakili sebagai tipe bawaan tertentu yang sesuai dengan tipe JSON yang tersedia untuk membantu Anda menulis lebih banyak kode yang bisa dipertahankan.

Ada empat metode untuk menulis data ke Firebase Realtime Database:

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.

Sumber:firebase.google.com

Gambar 2.4 Metode Menulis Data ke Firebase

Untuk operasi tulis dasar, Anda bisa menggunakan setValue() untuk menyimpan data ke referensi yang ditetapkan, menggantikan data yang ada di jalur tersebut.

Fungsi dalam pengambilan data melalui 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() danonChildRemoved() untuk memantau perubahan daftar.
	onChildRemoved()	Mendengarkan item yang dibuang dari daftar. Gunakan dengan onChildAdded() danonChildChanged() untuk memantau perubahan daftar.
	onChildMoved()	Gunakan dengan data diurutkan untuk mendengarkan perubahan dalam prioritas item.

Sumber:firebase.google.com

Gambar 2.5 Callback Kejadian dalam Pengambilan Data Firebase

Untuk menambahkan listener kejadian, gunakan metode addValueEventListener() atauaddListenerForSingleValueEvent(). Untuk menambahkan listener kejadian anak, gunakan metode addChildEventListener(). Metode onDataChange() untuk membaca cuplikan statis konten pada jalur tertentu, seperti yang telah ada pada saat kejadian. Metode ini terpicu satu kali 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).

2.4.3 Pengertian Android

Menurut Safaat (2015:1), "Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencankup sistem operasi, *middleware* dan aplikasi".

Menurut Firly (2018:2), "Android yaitu dalam Bahasa inggris istilah Android berarti "Robot yang menyerupai manusia", hal tersebut dapat terlihat jelas pada icon android yang menggambarkan sebuah robot berwarna hijau yang memiliki sepasang tangan dan kaki".

Jadi kesimpulannya yaitu Android adalah sebuah sistem operasi perangkat *mobile* berbasis linux yang mencankup sistem operasi yang menggambarkan sebuah robot menyerupai manusia.

2.4.4 Pengertian Node.js

Node.Js adalah sebuah platform baru yang dikembangkan oleh Ryan Dahl, dimana memungkinkan programer JavaScript untuk membuat sebuah server dengan kinerja yang sangat tinggi memanfaatkan google v8 JavaScript engine dan asynchronous I/O. Menurut (Iqbal, Husni, & Studiawan, 2012, p. 242) adalah sistem perangkat lunak yang didesain untuk pengembangan aplikasi web. Aplikasi ini ditulis dalam Bahasa JavaScript, menggunakan basis event dan asynchrounous I/O. Tidak seperti kebanyakan bahasa JavaScript yang dijalankan pada peramban, Node.Js dieksekusi sebagai aplikasi server. Aplikasi ini terdiri dari V8 JavaScript Engine buatan Google dan beberapa modul bawaan yang terintegrasi.

Node.Js sebagian besar diimplementasikan dalam C dan C++, berfokus pada kinerja dan penggunaan memori yang rendah. Node.Js bertujuan untuk menyokong lama berjalanya proses server. Proses node tidak bergantung pada multithreadding untuk mendukung pelaksaan business logic, hal ini berdasar pada model event asynchronous I/O.

JavaScript sendiri sangat cocok untuk dengan node karena menudukung event callback. Sifat dasar JavaScript ini membuatnya sangat mudah untuk membuat objek yang dijadikan sebagai event handler (Tilkov & Vinoski, 2010).

2.4.5 Pengertian Java

Enterprise (2016:1), “Java merupakan Bahasa pemrograman yang sangat popular karena rentang aplikasi yang bisa dibuat menggunakan bahsa ini sangatlah luas, mulai dari komputer hingga smartphone”.

Sedangkan menurut Hariyanto (2014:3), “ Java merupakan bahasa orientasi objek untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, aplikasi untuk perangkat cerdas yang dapat berkomunikasi lewat internet/jaringan komuniakasi”.

Berdasarkan kedua pendapat diatas, maka dapat disimpulkan bahwa java adalah Bahasa pemrograman yang sangat popular untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, aplikasi untuk perangkat cerdas yang dapat berkomunikasi lewat internet/jaringan komunikasi.