BAB II
TINJAUAN PUSTAKA

2.1 Sistem

Sistem pada dasarnya adalah sekelompok unsur yang erat berhubungan satu dengan lainnya, yang berfungsi bersamasama untuk mencapai tujuan tertentu. (Mulyadi,2016)

2.2 Sistem Informasi

Sistem informasi adalah suatu sistem yang terdiri dari kumpulan komponen sistem, yaitu *software*, *hardware* dan *brainware* yang memproses informasi menjadi sebuah *output* yang berguna untuk mencapai suatu tujuan tertentu dalam suatu organisasi. (Mulyanto,dkk,2017)

2.3 Point Of Sales

Aplikasi *point of sale* (POS) merupakan aplikasi yang dirancang sesuai dengan kebutuhan dan dapat untuk diintegrasikan dengan alat pendukung supaya dapat membantu mempermudah dan mempercepat proses transaksi. Usaha pada bidang penjualan membuat POS menjadi sistem yang sangat penting, karena itu diharapkan aplikasi POS ini dapat menyajikan informasi–informasi transaksi yang dilakukan dan juga berbagai macam laporan penjualan yang diinginkan (Syarifudin dan Kosasih, 2015). Keuntungan yang diperoleh dengan POS yaitu dapat meingkatkan kualitas pelayanan dikarenakan proses transaksi akan menjadi lebih cepat dan lebih sistematis, sehingga dapat mendukung kualitas pelayanan terhadap pelanggan, meningkatnya citra usaha dikarenakan konsumen yang puas sehingga membuat perusahaan mempunyai nilai lebih, kemudian keuntungan dalam berkompetensi yang dikarenakan segi pengelolaan data dan efisiensi waktu lebih unggul dibanding dengan yang tidak menggunakan sistem informasi, dan kemudahan dalam pengambilan keputusan dan proses pengontrolan karena dengan adanya POS tersebut dapat dengan mudah melakukan pengontrolan dikarenakan semua laporan sudah tersedia dengan cepat dan juga mempermudah dalam proses pengambilan keputusan (Nugraha dan Rahayu, 2013)

2.4 Android

Android merupakan sistem operasi untuk telepon seluler yang berbasis *Linux,* Android menyediakan platform bagi para pengembangan yang menciptakan aplikasi sendriri untuk digunakan oleh bermacam piranti bergerak. Untuk mengembangkan android dibentuklah *Open Handset Alliance konsorsium* dari 34 perusahaan piranti perangkat keras dan piranti perangkat

lunak, dan telekomunikasi termasuk Google, HTC, *Intel, Motorola, Qualcomm, T-Mobile,* dan Nvidia (Arifianto, 2011).

2.5 Flutter

Dalam buku Pemrograman Android dengan Flutter, Flutter adalah *software development kit* (SDK) buatan google yang berfungsi untuk membuat aplikasi mobile menggunakan bahasa pemrograman Dart, baik untuk Android maupun iOS. Dengan fluter, aplikasi Android dan iOS dapat dibuat menggunakan basis kode dan Bahasa pemrograman yang sama, yaitu Dart, Bahasa pemrograman yang juga diproduksi oleh Google pada tahun 2011. Sebelumnya, aplikasi murni (*native*) untuk Android perlu dibuat menggunakan Bahasa pemrograman Java atau Kotlin. Sedangkan aplikasi iOS perlu dibuat menggunakan Bahasa pemrograman *Objective*-C atau *Swift.* Flutter ditujukan untuk mempermudah dan mempercepat proses pengembangan aplikasi mobile yang dapat berjalan di atas Android dan iOS, tanpa harus mempelajari dua Bahasa pemrograman secarah terpisah.(Raharjo Budi,2019)

2.6 Dart

Dart adalah bahasa pemrograman yang diproduksi oleh *Google*, dirancang oleh Lars Bak dan Kasper Lund. Dart pertama kali dikenalkan pada 10 Oktober 2011. (Raharjo Budi,2019) Dart dapat digunakan untuk membuat aplikasi *server* (berbentuk *command-line interface*), web, maupun mobile (Android dan iOS). Aplikasi Dart dieksekusi secara langsung melalui Dart *Virtual Machine* (VM) tanpa melalui proses penerjemahan ke kode objek (*bytecode*) terlebih dahulu. Gambar berikut menunjukan proses eksekusi aplikasi Dart

2.7 MongoDB

MongoDB adalah salah satu jenis *database NoSQL* berbasis dokumen dengan menggunakan format file berupa JSON (*JavaScript Object Notation*). Jika dikomparasikan dengan penggunaan *database* SQL, dimana setiap data tersimpan dalam bentuk tabel. Sedangkan pada *Mongo*DB, data akan disimpan ke dalam sebuah dokumen berformat JSON. Pada umumnya, penggunaan dari *NoSQL* sendiri lebih dikhususkan untuk menangani jumlah data yang sangat besar (*big data*). Sehingga, arsitektur dari kedua jenis basis data tentu sangat berbeda.(Adani,2021)

2.8 Metode Pengembangan Sistem

Dalam perancangan Tugas Akhir ini penulis menggunakan metode *Prototype. Prototype Model* adalah salah satu metode pengembangan perangkat lunak yang banyak

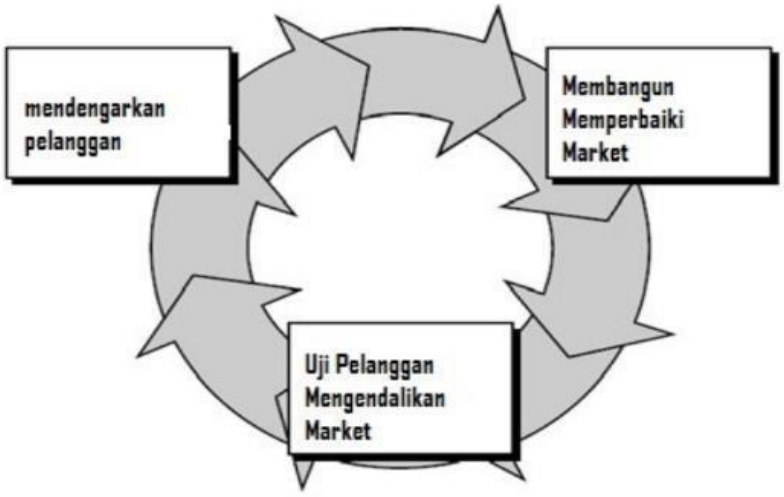
digunakan. Dengan Metode Prototyping ini pengembangan dan pelanggan dapat saling berinteraksi selama proses pembuatan sistem. Sering terjadi seorang pelanggan hanya mendefinisikan secara umum apa yang dibutuhkan, Pemrosesan dan data-data apa saja yang dibutuhkan. Sebaliknya disisi pengembang Kurang memperhatikan efesiensi Algoritma. Kemampuan sistem oprasi dan *interface* yang menghubungkan manusia dengan computer. (Roger S Pressman,2010)

Pada *Prototyping model* kadang – kadang klien hanya memberikan beberapa kebutuhan umum *software* tanpa detail input, proses atau detail output dilain waktu mungkin tim pembangun (developer) tidak yakin terhadap efesiensi dari algoritma yang digunakan, tingkat adaptasi terhadap sistem operasi atau rancangan form *user interface*. Ketika situasi seperti ini terjadi model *prototyping* sangat membantu proses pembangunan *software.* Proses pada *prototyping* bisa dijelaskan sebagai berikut :

a.Pengumpulan Kebutuhan : *developer* dan *klien* bertemu dan menentukan tujuan umum, kebutuhan yang diketahui dan gambaran bagian-bagian yang akan dibutuhkan berikutnya. Detail kebutuhan mungkin tidak dibicarakan disini, pada awal pengumpulan kebutuhan.

b.Perancangan : Perancangan dilakukan cepat dan rancangan mewakili aspek *software* yang diketahui. Dan rancangan ini menjadi dasar pembuatan *prototype.*

c.Evaluasi *Prototypen:* klien mengevaluasi *prototype* yang dibuat dan dipergunakan untuk memperjelas kebutuhan *software.*

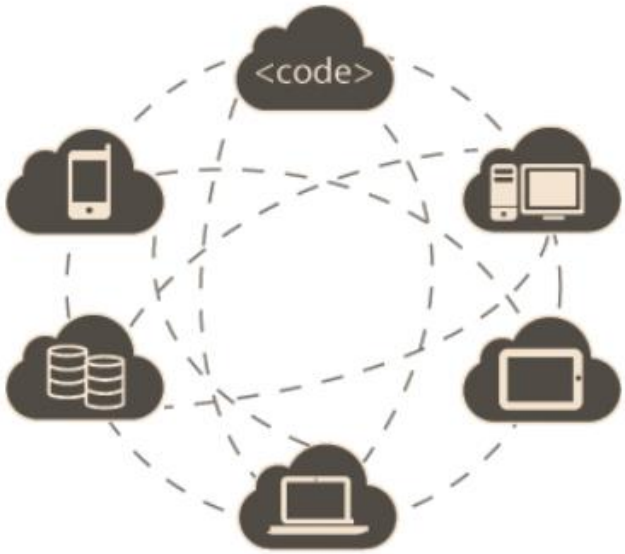


Gambar 2.1 Model Pengembangan *Prototype* (Roger S. Pressman Ph.D, 2010)

2.9 *Cloud Computing*

Cloud Computing adalah sebuah model komputasi dimana aktivitas pemrosesan. penyimpanan, perangkat lunak dan layanan lainnya disediakan layaknya sumber *virtual* terpadu pada suatu jaringan yang umumnya adalah internet. Sumber daya komputasi dari *cloud computing* tersebar dan dapat diakses berdasarkan kebutuhan dari perangkat apapun dan dimanapun terhubung.

Konsep Cloud computing biasanya dianggap sebagai internet. Karena internet sendiri digambarkan sebagai awan (*Cloud*) besar (biasanya dalam skema jaringan, internet dilambangkan sebagai awan) yang berisi sekumpulan komputer yang saling terhubung. *Cloud computing* datang sebagai sebuah evolusi yang mengacu pada konvergensi teknologi dan aplikasi lebih dinamis.



Gambar 2.2 *Cloud computing* (Sumber : DataART)

2.10 UML (*Unified Modeling Language*)

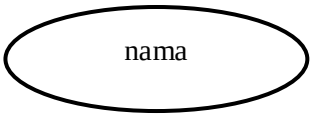
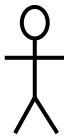
Bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan yang kompleks sehingga lebih mudah untuk dipelajari dan dipahami. (Rosa A.S , M. Shalaluddin, 2016).Tujuan pemodelan dalam kerangka pengembangan sistem adalah sebagai sarana analisis, pemahaman, *visualisasi*, dan komunikasi antar tim

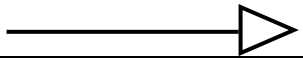
pengembang yang beranggotakan beberapa/banyak anggota. Beberapa diagram dalam UML yang akan digunakan dalam membantu pengembangan sistem yaitu :

2.10.1 Usecase Diagram

Usecase Diagram adalah sebuah pemodelan untuk kelakuan (behavior) sistem informasi yang akan diuat. *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Berikut merupakan simbol-simbol dari *Usecase Diagram*:

Tabel 2.1 Simbol *Usecase Diagram* (Rosa, 2016)

Simbol	**Desripsi**
Usecase  	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *Usecase*
Aktor/ *actor*  nama aktor	Aktor merupakan 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 aktor
Asosiasi/ *asociation* 	Komunikasi antara aktor dan *Usecase* yang berpatisipasi pada *Usecase* atau *Usecase* memiliki Interaksi dengan aktor

Ekstensi/ *extend*  <<*extend*>>	Relasi *Usecase* tambahan ke sebuah *Usecase* dimana *Usecase* yang ditambahkan dapat berdiri sendiri walau tanpa *Usecase* tmbahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *Usecase* tambahan memiliki nama depan yang sama dengan *Usecase* yang ditambahkan.
Generalisasi/ *generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *Usecase* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya
Menggunakan/ *Include/ uses* <<*include*>> 	Relasi*Usecase*tambahan ke sebuah *Usecase* dimana *Usecase* yang ditambahkan memerlukan *Usecase* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *Usecase*ini

2.10.2 Activity Diagram

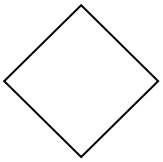
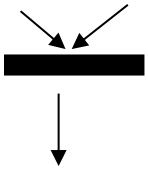
Diagram Aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas sebuah sistem atau operasi bisnis. Perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

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 didefinisikan.
b. Urutan atau pengelompokkan tampilan dari sistem/user *interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilannya.
c. Rancangan pengujian dimana setiap aktivitas dianggap memberikan sebuah pengujian yang perlu didefinisikan kasus ujinya.

Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Tabel 2.2 *Activity Diagram* (Rosa, 2016)

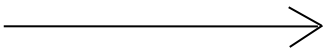
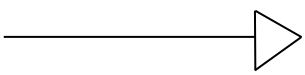
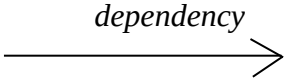
Simbol	**Deskkripsi**
Simbol Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasana diawali dengan kata kerja.
Percabangan / ***Decision*** 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan/ Join 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu status.
Status Akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Join 	Digunakan untuk menunjukan kegiatan yang digabungkan

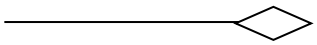
Sumber : Rossa A.S dan M.Shalahuddin, 2013

2.10.3 *Class Diagram*

Menurut Rosa & Salahuddin (2016) *class diagram* mengembangkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Berikut simbol-simbol yang akan digunakan dalam menggambarkan *Class Diagram* dapat dilihat pada tabel 2.3 berikut ini:

Tabel 2.3 *Class Diagram* (Rosa, 2016)

No.	Simbol	Deskripsi
1.	Nama_kelas +Atribute +Operasi	Kelas pada struktur sistem**.**
2.	Antar Muka/*Interface*  Nama_*Interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
3.	Asosiasi / *Asociation* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan symbol
4.	Asosiasi Berarah / *Directed Association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan symbol.
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)
6.	Ketergantungan / *dependency* 	Relasi antar kelas dengan makna ketergantungan antar kelas.

7.	Agregasi / *aggregation* 	Relasi antar kelas dengan maksna semua bagian *(whole-part)*

Sumber: (Rosa & Salahuddin, 2013)

2.11 Blackbox Testing

Blackbox testing adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian blackbox testing harus dibuat dengan kasus benar dan kasus salah**.**(Shalahuddin dan Rosa,2016),