BAB II
TINJAUAN PUSTAKA

2.1 Teori Umum

2.1.1 Pengertian Komputer

Menurut Daulay (2007:17), "Komputer adalah sekumpulan alat logik yang dapat menerima data, mengolah data dan menyimpan data dengan menggunakan program yang terdapat pada memori sistem komputer kemudian memberikan hasil pengolahan tersebut dalam bentuk *output.* Dengan kata lain komputer juga bisa diartikan sebagai sekumpulan perangkat elektronika yang terdiri dari unit *input,* proses dan *output."*. Pernyataan senada dikemukakan oleh Mulyono (2010:1), "Komputer adalah seperangkat alat elektronik yang terdiri atas peralatan *input*, dan peralatan *output* yang memberikan informasi, serta bekerja secara otomatis". Selain itu menurut Siallagan (2009:1), "Komputer adalah sebagai sekumpulan alat elektronik yang saling bekerja sama, dapat menerima data (*input*), mengolah data (*proces*), memberikan informasi (*output*), dan terkoordinasi di bawah kontrol program yang tersimpan dalam memorinya".

Sehingga dapat disimpulkan bahwa definisi dari komputer adalah sekumpulan alat elektronik yang membentuk sistem dan bekerja di bawah kontrol memori sehingga dapat mengolah *input* berupa data menjadi *output* berupa informasi.

2.1.2 Pengertian Perangkat Lunak

Mulyono (2010:97) menyatakan bahwa, "*Software* adalah rangkaian instruksi elektronik yang memerintahkan komputer untuk melakukan tugas tertentu sesuai dengan perintah yang diberikan oleh seorang pengguna komputer". Menurut Sukamto dan Shalahuddin (2016:2), "Perangkat lunak *(software)* adalah program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain, dan cara penggunaan *(user manual)"*. Sedangkan menurut Ladjamudin (2013:20), "*Software* merupakan kumpulan dari

perintah/fungsi yang ditulis dengan aturan tertentu untuk memerintahkan komputer melaksanakan tugas tertentu"

Dapat disimpulkan bahwa definisi perangkat lunak merupakan rangkaian instruksi dari program komputer yang berfungsi mengeksekusi dan menyelesaikan persoalan tertentu.

2.1.3 Pengertian Android

Menurut Safaat (2015: 15), "*Android* adalah sebuah sistem operasi untuk perangkat *mobile* berbasis *linux* yang mencakup sistem operasi, *middleware,* dan aplikasi". Sedangkan menurut Boedjiono, *et al* (2015:2), "*Software stack* Google untuk perangkat mobile".

Dari dua pernyataan di atas, dapat disimpulkan bahwa *Android* adalah sistem operasi berbasis *linux* yang ditujukan untuk perangkat *mobile.*

2.2 Teori Khusus

2.2.1 Pemrograman Berorientasi Objek

2.2.1.1 Pengertian Pemrograman Berorientasi Objek

Menurut Sukamto dan Shalahuddin (2016:99), "Pemrograman berorientasi objek adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya."

Dari definisi di atas dapat disimpulkan bahwa pemrograman berorientasi objek adalah program komputer dari berbagai objek yang melakukan suatu tindakan terhadap masing-masing objek.

2.2.2 *Rational Unified Process* (RUP)

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*). RUP merupakan proses rekayasa perangkat lunak dengan pendefinisian yang baik (*well defined*) dan penstrukturan yang baik (*well structured*). RUP menyediakan pendefinisian struktur yang baik untuk alur hidup proyek perangkat lunak.

Proses pengulangan/iteratif pada RUP secara global dapat dilihat pada gambar berikut:

(*Sumber*: Sukamto dan Shalahuddin (2016:125))

Gambar 2.1 Proses iteratif *RUP (Rational Unified Process)*

2.2.2.1 Fase *Rational Unified Process (RUP)*

Sukamto dan Salahuddin (2016:128-131), "RUP (*Rational Unified Process*) memiliki empat buah tahap atau fase yang dapat dilakukan secara iteratif." Berikut ini penjelasan untuk setiap fase pada RUP (*Rational Unified Process*)."

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

b. *Elaboration* (perluasan/perencanaan) Tahap ini lebih difokuskan pada perencanaan arsitektur sistem. Tahap ini juga dapat mendeteksi apakah sistem yang diinginkan dapat dibuat atau tidak. Tahap ini lebih pada analisis dan desain sistem serta implementasi sistem yang fokus pada purwarupa sistem (*prototype*).

c. *Construction* (konstruksi) Tahap ini fokus pada pengembangan komponen dan fitur-fitur sistem. Tahap ini lebih 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.

d. *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/tonggak kemampuan operasional awal. Aktivitas pada tahap ini termasuk pada pelatihan *user*, pemeliharaan dan pengujian sistem apakah sudah memenuhi harapan *user*.

Akhir dari keempat fase ini adalah produk perangkat lunak yang sudah lengkap. Keempat fase pada RUP (*Rational Unified Process*) dijalankan secara berurutan dan iteratif dimana setiap iterasi dapat digunakan untuk memperbaiki iterasi berikutnya.

2.2.3 *Unified Modelling Language* (*UML*)

Menurut Safaat (2015:33) dikutip dari Widodo, "*UML* adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. *UML* menawarkan sebuah standar untuk merancang model sebuah sistem.", Sukamto dan Salahuddin (2016:133), "*UML* (*Unified Modelling Language*) adalah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek."

Dari beberapa definisi di atas penulis menyimpulkan bahwa *UML* (*Unified Modelling Language*) adalah sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasikan sebuah sistem pengembangan perangkat lunak berorientasi objek.

2.2.3.1 Klasifikasi Diagram *UML* (*Unified Modelling Language*)

Sukamto dan Salahuddin (2016:140), "*UML* (*Unified Modelling Language*) terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori". Berikut ini penjelasan singkat dari pembagian kategori tersebut.

a. *Structure* Diagram

Yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

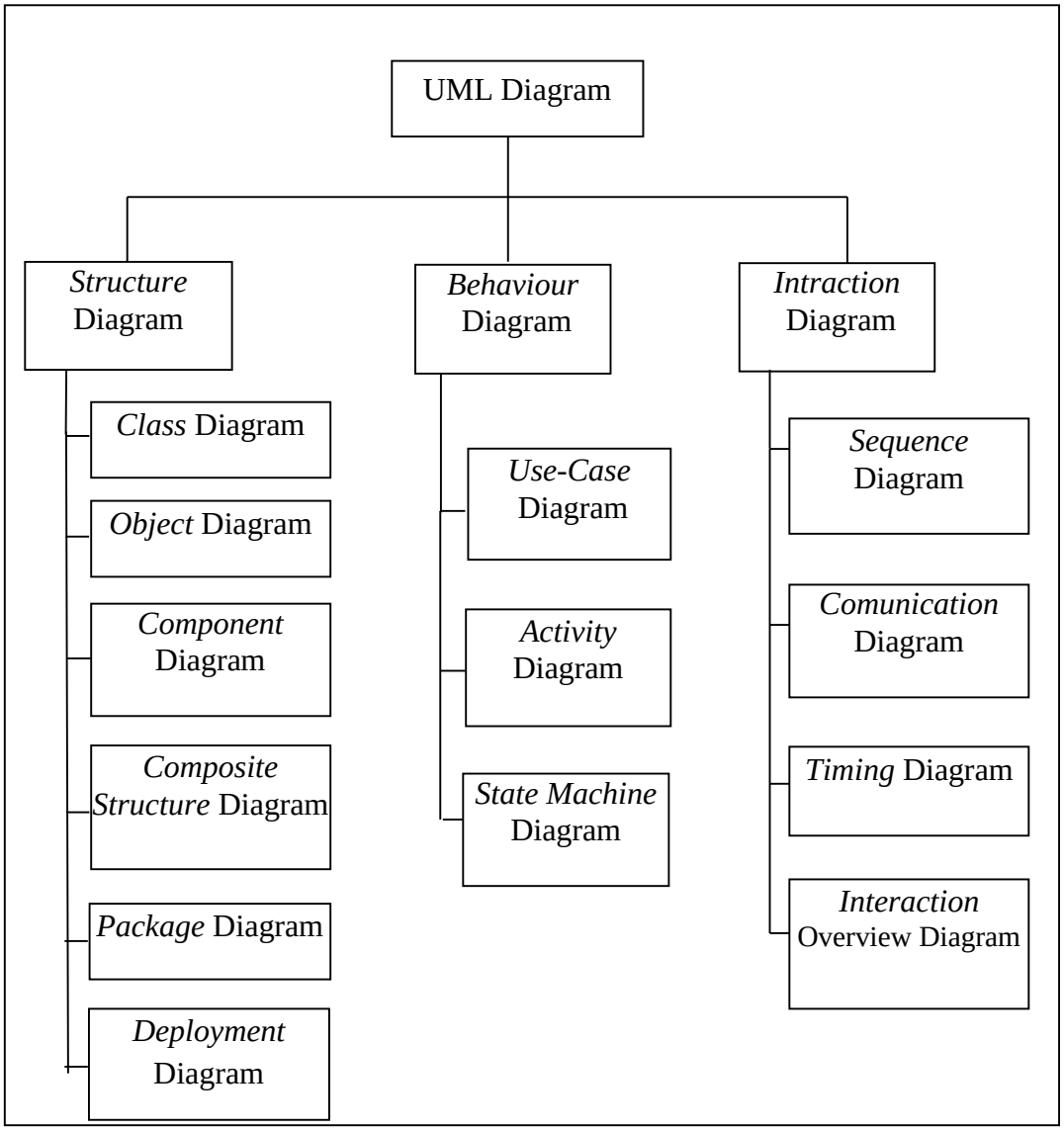
b. *Behavior* Diagram

Yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

c. *Interaction* Diagram

Yaitu kumpulan diagram yang digunakan untuk menggambarkan interkasi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar dibawah ini:



(*Sumber*: Sukamto dan Shalahuddin (2016:140))

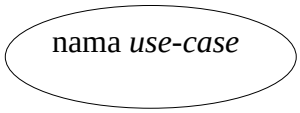
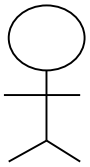
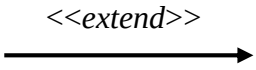
Gambar 2.2 Klasifikasi Diagram UML (*Unified Modelling Language*)

2.2.4 *Use-Case* Diagram

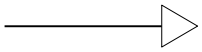
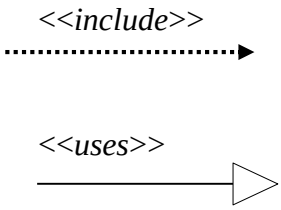
Sukamto dan Salahuddin (2016:155), *"Use-case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use-case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat". Adapun simbol-simbol yang digunakan dalam *use*

case adalah sebagai berikut:

Tabel 2.1. 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 frase nama *use-case*.
2.	Aktor / a*ctor*  nama aktor nama_*interface*	Orang, proses, atau sistem 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 berpartisipasi pada *use-case* atau *use-case* yang memiliki interaksi dengan actor.
4.	Ekstensi / *extend* 	Relasi *use-case* tambahan ke sebuah *use-case* dimana *use-case* yang ditambahkan dapat berdiri sendiri walau tanpa *use-case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; ditambahkan, missal 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.

Lanjutan Tabel 2.1. Simbol-simbol *Use-Case* Diagram

No	Simbol	Deskripsi
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.
6.	Menggunakan / *include* /*uses* 	Relasi *use-case* tambahan ke sebuah *use-case* di mana *use-case* yang ditambahkan memerlukan *use-case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use-case*.

(*Sumber*: Sukamto dan Shalahuddin (2016:156-158))

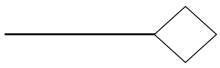
2.2.5 *Class* Diagram

Menurut Safaat (2015:33*), "Class Diagram* merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode/fungsi) *class diagram* menggambarkan struktur dan deskripsi class, *package*, dan objek yang berhubungan satu sama lain seperti *containment*, asosiasi, dan lain-lain". Sedangkan menurut Sukamto dan Salahuddin (2016:141), "*Class* diagram atau diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi".

Tabel 2.2 Simbol-simbol *Class* Diagram

No	Simbol	Deskripsi
1.	Kelas nama_kelas + attribut + operasi ()	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 dengan *multiplicity*.
4.	Asosiasi berarah / *directed association*	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/ d*ependency*	Relasi antarkelas dengan makna kebergantungan antarkelas.

Lanjutan Tabel 2.2 Simbol-simbol *Class* Diagram

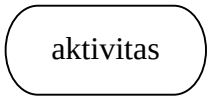
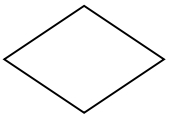
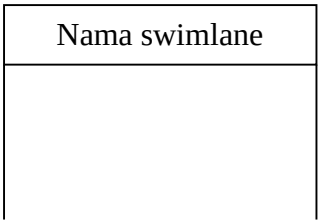
No	Simbol	Deskripsi
7.	Agregasi / *aggregation* 	Relasi antarkelas dengan makna semua-bagian *(whole-part).*

(*Sumber*: Sukamto dan Shalahuddin (2016:144-147)

2.2.6 *Activity Diagram*

Menurut Safaat (2015:34), "*Activity diagram* menggambarkan berbagai alir aktivitas dalam sistem yang sedang diracang, bagaimana masing-masing alir berawal, *decision* yang terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi". Sedangkan, Sukamto dan Salahuddin (2016:161) menyatakan, "*Activity* diagram adalah diagram yang menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis yang ada pada perangkat lunak".

Tabel 2.3 Simbol-simbol *Activity* Diagram

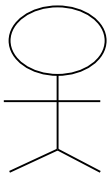
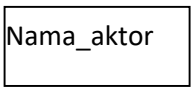
No	Simbol	Deskripsi
1.	Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih 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.
6.	*Swimlane* 	*Swimlane* memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

(*Sumber*: Sukamto dan Shalahuddin (2016:162-163)

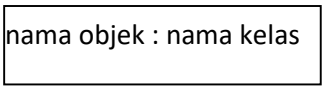
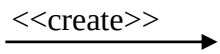
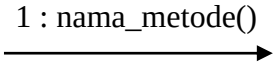
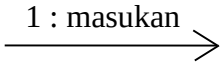
2.2.7 *Sequence Diagram*

Sukamto dan Salahuddin (2016:165), "*Sequence diagram* adalah diagram yang menggambarkan kelakuan objek pada *use-case* dengan mendeskripsikan waktu daur hidup objek dan *message* yang dikirimkan dan diterima antar objek". Sedangkan Menurut Safaat (2015:33-34), "*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu."

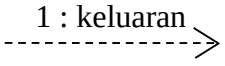
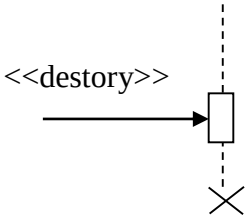
Tabel 2.4 Simbol-simbol *Sequence Diagram*

No	Simbol	Deskripsi
1.	Aktor  Aktor Atau  tanpa waktu aktif	Orang, proses, atau sistem 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*.
2.	Garis hidup/ *lifeline* 	Menyatakan kehidupan suatu objek.

Lanjutan Tabel 2.4. Simbol-simbol *Sequence Diagram*

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.
5.	pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat.
6.	Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri.
7.	Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data/ masukan/ informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi.

Lanjutan Tabel 2.4 Simbol-simbol *Sequence Diagram*

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 *destrory* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada *create* maka ada *destroy*.

(*Sumber*: Sukamto dan Shalahuddin (2016:165-167))

2.2.8 Metode Pengujian Perangkat Lunak

2.2.8.1 Pengertian Metode Pengujian

Sukamto dan Shalahuddin (2016:272) menyatakan, "Pengujian adalah satu set aktifitas yang direncanakan dan sistematis untuk menguji atau mengevaluasi kebenaran yang diinginkan. Aktifitas pengujian terdiri dari satu set atau sekumpulan langkah dimana dapat menempatkan desain kasus uji yang spesifik dan metode pengujian."

2.2.8.2 Metode Pengujian

Secara umum pola pengujian perangkat lunak adalah sebagai berikut:

a. Pengujian dimulai dari level komponen hingga integrasi antar komponen menjadi sebuah sistem.
b. Teknik pengujian berbeda-beda sesuai dengan berbagai isi atau unit uji dalam waktu yang berbeda-beda pula bergantung pada pengujian pada bagian mana yang dibutuhkan.

c. Pengujian dilakukan oleh pengembang perangkat lunak, dan jika untuk proyek besar, pengujian bisa dilakukan oleh tim uji yang tidak terkait dengan tim pengembang perangkat lunak (*independent test group* (ITG)).
d. Pengujian dan penirkutuan (*debugging*) merupakan aktivitas yang berbeda tetapi penirkutuan (*debugging*) harus diakomodasikan pada berbagai strategi pengujian.

2.2.8.3 *Back-Box Testing* (Pengujian Kotak Hitam)

Sukamto dan Shalahuddin (2016:275), "*Black-box testing* (pengujian kotak hitam) yaitu 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."

Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login maka kasus uji coba yang dilakukan adalah:

a. Jika user memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
b. Jika user memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalkan nama pemakai benar tapi kata sandi salah, atau sebaliknya atau keduanya salah.

2.3 Teori Judul

2.3.1 Pengertian Aplikasi

Budiharto (2013:5) menyatakan bahwa "*Applications* merupakan program yang dapat berjalan di komputer tersendiri (*stand alone computer*), dari mulai program yang simpel sampai dengan program besar dan rumit". Sedangkan Mulyono (2010:109) mengemukakan bahwa, "Perangkat lunak yang khusus

ditambahkan dalam sistem operasi yang ada untuk melakukan pekerjaan khusus sesua dengan kebutuhan pengguna komputernya disebut software aplikasi".

Berdasarkan pendapat di atas, maka dapat disimpulkan bahwa aplikasi adalah perangkat lunak berupa program yang berfungsi untuk membantu *user* mengerjakan tugas tertentu.

2.3.2 Pengertian *e-Voting*

Menurut Suharsono dan Priyatna (2017:2), *"e-Voting* adalah suatu sistem pemilihan dimana data dicatat, disimpan, dan diproses dalam bentuk informasi digital". Dikutip dari Purwati (2015:2) menurut Shalahuddin (2009), "Sebuah perangkat pemberian suara secara elektronik, sehingga memiliki kemampuan untuk mempercepat tabulasi data, menekan biaya pemilihan, dan memiliki kontribusi untuk mencegah pemilih yang tidak berhak".

2.3.3 Pengertian Pemilu Raya

Menurut Undang-Undang Dasar KM Politeknik Negeri Sriwijaya Bab VIII tentang Pemilihan Umum, Pasal 41 menyatakan bahwa, "Pemilihan Umum adalah sarana suksesi kepemimpinan lembaga Keluarga Mahasiswa Politeknik Negeri Sriwijaya".

2.3.4 Pengertian Keluarga Mahasiswa Politeknik Negeri Sriwijaya

Menurut Undang-Undang Dasar KM Politeknik Negeri Sriwijaya Bab I tentang Bentuk dan Kedaulatan, Pasal 1-3 menyatakan bahwa:

1. Keluarga Mahasiswa Politeknik Negeri Sriwijaya adalah sebuah organisasi mahasiswa yang legal dan formal yang mewadahi seluruh aktivitas kemahasiswaan di Politeknik Negeri Sriwijaya.
2. Keluarga Mahasiswa Politeknik Negeri Sriwijaya terdiri atas lembaga kemahasiswaan ditingkat Politeknik, Jurusan, dan Badan Semi Otonom yang

merepresentasikan Negara Kesatuan Republik Indonesia sesuai dengan kebutuhan dan kultur dunia kemahasiswaan.

3. Kedaulatan berada ditangan Mahasiswa dan dilaksanakan menurut UUD KM-POLSRI.

2.3.5 Pengertian Aplikasi *e-Voting* Pemilihan Umum Raya Keluarga Mahasiswa Politeknik Negeri Sriwijaya

Aplikasi *e-Voting* Pemilihan Umum Raya Keluarga Mahasiswa Politeknik Negeri Sriwijaya adalah aplikasi *e-Voting* yang ditujukan untuk penggunaan pada pemilihan umum pimpinan lembaga mahasiswa Politeknik Negeri Sriwijaya.

2.4 Teori Program

2.4.1 Pengertian Basis Data *(Database)*

Dikutip dari Simarmata dan Paryudi (2006:1), menurut Stephens dan Plew (2000), "Basisdata adalah mekanisme yang digunakan untuk menyimpan informasi atau data. Informasi adalah sesuatu yang kita gunakan sehari-hari untuk berbagai alasan." Penjelasan lain dikemukakan oleh Fathansyah (2012:2-3), "Basis Data terdiri atas 2 kata, yaitu basis dan data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya."

2.4.1.1 Pengertian *MySQL*

Menurut Kadir (2008:2), *"MySQL* (baca: mi-se-kyu-el) merupakan *software* yang tergolong sebagai DBMS (*Database Management System*) yang bersifat *open source*." Menurut Raharjo, *et al* (2012:210), *"MySQL* merupakan sistem *database* yang banyak digunakan untuk pengembangan aplikasi *web*."

Dapat disimpulkan bahwa *MySQL* merupakan *software* DBMS yang bersifat *open source* sehingga marak digunakan dalam pengembangan aplikasi.

2.4.2 Pengertian *Ionic Framework*

Menurut Boedjiono, *et al* (2015:2), "*Ionic Framework* adalah kerangka pembangunan aplikasi mobile HTML5 yang ditargetkan untuk membangun aplikasi *mobile hybrid.* Aplikasi *hybrid* pada dasarnya adalah *website* yang berjalan dalam *browser* sebuah aplikasi yang memiliki akses ke lapisan *platform native.* ". Anditya dan Ilhami (2015: 2-3), "*Framework Ionic* merupakan kerangka kerja yang menyediakan sejumlah kontrol antarmuka pengguna yang umum digunakan di aplikasi *mobile.*

Dari dua pernyataan di atas, dapat disimpulkan bahwa *Ionic Framework* adalah yang digunakan dalam pengembangan aplikasi *hybrid.*

2.4.2.1 Kelebihan *Ionic*

Menurut Anditya dan Ilhami (2015:6-7) berikut adalah kelebihan *Ionic:*

1. Ionic untuk Pengembang

Ionic mampu memenuhi kebutuhan pengembang yang menginginkan aplikasi yang cepat.

1. Dapat dibangun dengan *platform web.*
2. Dibangun dengan *AngularJS.* Bagi yang sudah *familiar* dengan *Angular,* tentu *Ionic* adalah pilihan terbaik.
3. Teknologi terkini, *Ionic* didesain agar dapat bekerja dengan fitur-fitur modern CSS, seperti animasi.
4. CLI tools. Dengan adanya *Command Line Tool,* anda dapat dengan cepat mengembangkan aplikasi melalui *browser.*

2. Ionic untuk Pemilik Bisnis

Ini dikarenakan *framework* ini juga disebuat *hybrid mobile app.* Artinya, satu aplikasi yang dibangun dengan *Ionic* mampu bekerja di *Android* dan *iOs.*

1. Hemat biaya. Tidak perlu menghabiskan biaya untuk dua platform, karena satu aplikasi yang dibangun dengan *Ionic* dapat bekerja di *Android* dan *iOs.*
2. *Open-source*. Tidak perlu membayar royalti untuk hak cipta, karena sepenuhnya *Ionic* ini berlisensi untuk *open source.*
3. Mudah dikembangkan. Membuat aplikasi dengan *Ionic* tidak butuh waktu yang lama. Dengan waktu singkat, aplikasi sudah bisa langsung *running.*

3. Ionic untuk Pengguna

1. Desain yang menarik. *Ionic* didesain sehingga dapat bekerja dan secara desain dapat digunakan pada hampir semua *mobile platform* sekarang ini.
2. Performansi. Dengan mengurangi manipulasi DOM, tidak ada *JQuery* dan *hardware accelerated transitions,* membuat *Ionic* menjadi salah satu *framework* yang relatif cepat sekarang ini.
3. Didukung di *Android* dan *iOs.* Saat ini *Ionic* telah dapat digunakan pada *platform Android* dan *iOs.*

2.4.3 Pengertian *CodeIgniter*

Menurut Hidayatullah dan Kawistara (2014:345), "*CodeIgniter* adalah suatu *framework* dengan metode *pattern Model View Controller* (MVC)". Hidayatullah dan Kawistara (2014:283-284) menjelaskan bahwa, "MVC adalah metode yang memisahkan *data logic (model)* dari *presentation logic (view) dan process logic (controller).*"

Sehingga dapat disimpulkan *CodeIgniter* adalah *framework* yang memisahkan antara logika data dengan logika tampilan dan logika proses.

2.4.4 Bahasa Pemrograman

2.4.4.1 Pengertian *AngularJS*

Boedjiono, *et al* (2015:2), *"AngularJS* adalah kerangka struktural untuk aplikasi web dinamis. AngularJS memungkinkan penggunanya menggunakan HTML sebagai bahasa pemrograman yang dipakai dan memungkinkan pengguna memperluas sintaks HTML untuk mengekspresikan komponen aplikasi yang dibuat oleh pengguna dengan jelas dan ringkas. Data *binding AngularJS* dan *dependency injection AngularJS* dapat memperingkas proses *coding.* Dan semua proses tersebut terjadi dalam *browser*, sehingga *AngularJS* mampu menjadi pasangan yang ideal dengan teknologi *server."*

(*Sumber:* Anditya dan Ilhami (2015:4))

Gambar 2.4 AngularJS by Google

2.4.1.1 Pengertian *Apache Cordova*

Menurut Boedjiono, *et al* (2015:2), *"Apache Cordova* adalah satu *set* perangkat API yang memungkinkan pengembang aplikasi mobile untuk mengakses fungsi perangkat *native."* Anditya dan Ilhami (2015:5), "*Apache Cordova* merupakan *layer* yang mengatur komunikasi antara jendela *browser* dan *native API. Cordova* adalah proyek *open source Apache* yang memiliki komunitas yang cukup besar."

(*Sumber:* Anditya dan Ilhami (2015:5))

Gambar 2.4 Apache Cordova

2.4.4.2 Pengertian HTML *(Hypertext Markup Language)*

Menurut Sulhan (2006:23), "*Hypertext Markup Language* (HTML) adalah suatu sistem untuk menambahkan dokumen dengan *table* yang menandakan bagaimana teks di dokumen harus disajikan dan bagaimana dokumen dihubungkan bersama-sama.". Budiharto (2013:27-28) menyatakan bahwa, "Dokumen HTML merupakan dokumen web yang statis, artinya hanya mampu menampilkan teks dan gambar yang statis sehingga untuk membuatnya lebih dinamis (misal: teks dan gambar bergerak, menampilkan database suatu perusahaan, dan lainnya) harus menggunakan bahasa lainnya seperti JavaScript, Java, PHP, dll."

2.4.4.3 Pengertian PHP *(Hypertext Preprocessor)*

Madcoms (2012:206) menjelaskan bahwa, "PHP (*Hypertext Preprocessor*) adalah salah satu bahasa pemrograman yang berjalan dalam sebuah webserver dan berfungsi sebagai pengolah data pada sebuah server". Sedangkan, Badiyanto (2013:32) mengemukakan bahwa, "PHP: *Hypertext Preprocessor* adalah bahasa skrip yang dapat ditanamkan atau disisipkan ke dalam HTML/PHP banyak dipakai untuk membuat situs web dinamis".

2.4.5 Webserver

Untuk bisa mengakses aplikasi web yang akan dibuat oleh penulis, maka diperlukan yang namanya *webserver*. Penulis menggunakan *Xampp* sebagai *webserver* dalam membangun Aplikasi *e-Voting* Pemilu Raya KM Politeknik Negeri Sriwijaya.

2.4.5.1 Pengertian *XAMPP*

"*XAMPP* adalah salah satu paket software web server yang terdiri dari *Apache, MySQL, PHP* dan *phpMyAdmin*" (Madcoms, 2009:1). Hal yang senada juga dikemukakan oleh Nugroho (2013:1) bahwa, "*Xampp* adalah paket program web lengkap yang dapat anda pakai untuk belajar pemrograman web, khhususnya *PHP* dan *MySQL*, paket ini dapat didownload secara gratis dan legal".