BAB II
TINJAUAN PUSTAKA

2.1.Teori Umum

2.1.1. Pengertian Aplikasi

Sutabri (2012:147), aplikasi adalah alat terapan yang difungsikan secara khusus dan terpadu sesuai kemampuan yang dimilikinya. Asropudin (2013:6), *application* adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya *Ms-Word*, *Ms-Excel*.

Dari kedua pendapat di atas, maka dapat disimpulkan bahwa aplikasi adalah sebuah *software* yang dimiliki oleh sebuah komputer yang berguna untuk mengerjakan tugas tertentu sesuai dengan kemampuan yang dimilikinya.

2.1.2. Pengertian Informasi

Sutabri (2012:22), informasi adalah data yang telah diklasifikasikan atau diolah atau diinterpretasikan untuk digunakan dalam proses pengambilan keputusan. Pratama (2014:9), informasi merupakan hasil pengolahan data dari satu atau berbagai sumber, yang kemudian diolah, sehingga memberikan nilai, arti, dan manfaat.

Dari kedua pengertian di atas dapat disimpulkan bahwa informasi adalah suatu data yang telah diklasifikasikan dan diolah sehingga memiliki suatu manfaat yang dapat digunakan bagi penggunanya dan dapat digunakan dalam mengambil suatu keputusan.

2.1.3. Pengertian Dakwah

Kamus Besar Bahasa Indonesia (2008), dakwah adalah penyiaran agama dan pengembangannya dikalangan masyarakat. Tohyar (2011:207), dakwah dapat diartikan dengan semua kegiatan yang bersifat mengajak, menyeru, atau memanggil orang lain untuk menjalankan perintah Allah dan rasul-Nya sebagaimana tercantum dalam Al-Qur'an dan hadis untuk menggapai kebahagiaan hidup di dunia dan akhirat. Fakhruroji (2017:8), dakwah merupakan upaya

mengajak orang lain kepada keadaan yang lebih baik dalam ukuran-ukuran yang sesuai dengan ajaran Islam.

Berdasarkan pendapat di atas, dapat disimpulkan bahwa dakwah adalah kegiatan penyiaran agama yang bersifat mengajak, menyeru, atau memanggil orang lain untuk beriman dan taat kepada Allah SWT sesuai dengan aqidah, syariat dan akhlak Islam.

2.1.4. Pengertian *Android*

Wahana Komputer (20013:2), *Android* adalah sebuah sistem operasi *mobile* yang berbasiskan pada versi modifikasi *linux*. Sujatmiko (2012:19), *Android* adalah sistem operasi berbasis *linux* untuk telepon seluler seperti telepon pintar dan komputer tablet. *Android* menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh berbagai macam piranti bergerak.

Berdasarkan pengertian di atas, maka dapat disimpulkan bahwa *Android* merupakan sistem operasi *mobile* berbasis *linux* yang bersifat terbuka (*open source*) dan dirancang untuk perangkat seluler seperti *smartphone* dan komputer tablet.

2.1.4.1. Sekilas Tentang *Android*

Wahana Komputer (20013:2), pertama kali *Android* dikembangkan oleh perusahaan *Android.Inc*. Nama perusahaan inilah yang pada akhirnya digunakan sebagai nama proyek sistem operasi *mobile* tersebut, yaitu sistem operasi *Android*. Pada tahun 2005, sebagai bagian dari strategi untuk memasuki pasar *mobile*, *google* membeli *Android* dan mengambil alih proses pengembangannya sekaligus *team developer Android*. *Google* menginginkan *Android* untuk menjadi sistem operasi *open source* dan gratis, kebanyakan kode *Android* dirilis dibawah lisensi *Open Source Apache* yang berrarti setiap orang bebas untuk menggunakan dan mengunduh *source code Android* secara penuh. *Android* telah dikembangkan dan di-*update* beberapa kali semenjak dirilis pertamanya.

2.1.4.2. Generasi *Android*

Salbino (2014:17), ponsel pertama yang memakai Sistem Operasi *Android* adalah *HTC Dream* yang di rilis pada tanggal 22 Oktober 2008 dan pada awal tahun 2009 mulailah para pengembang ponsel menggunakan OS *Android* ini. Perkembangan *Android* adalah sebagai berikut:

1. *Android* Versi 1.0

   Android 1.0, merupakan versi komersial pertama *Android*, dirilis 23 September 2008 dengan kode nama *Apple Pie* serta ukuran layar 320x480 HVGA. Perangkat *Android* pertama yang tersedia secara komersial adalah HTC *Dream*.

2. *Android* Versi 1.1

   Android versi 1.1 di rilis pada 9 Maret 2009 oleh *Google*. *Android* ini telah di*support* oleh *Google Mail Service* dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan *gmail*, dan pemberitahuan *email*.

3. *Android* Versi 1.5 *Cup Cake*

   Android Cup Cake di rilis pada pertengahan Mei 2009, masih oleh *Google Inc*. *Android* ini telah dilengkapi *software development kit* dengan berbagai pembaharuan termasuk penambahan beberapa fitur lain, yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke *youtube*, *upload* gambar ke *Picasa* langsung dari telepon, serta mendapat dukungan *Bluetooth* A2DP.

4. *Android* Versi 1.6 *Donut*

   Android Donut di rilis pada September 2009 menampilkan proses pencarian yang lebih baik dari versi-versi sebelumnya. *Android Donut* juga memiliki fitur-fitur tambahan seperti galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus, kamera, *camcorder* dan galeri yang terintegrasikan, *Text-to-speech engine*, *dial* kontak, teknologi *text to change speech*, baterai indikator dan kontrol *applet* VPN.

5. *Android* Versi 2.0/2.1 *Eclair*

   Android Eclair dirilis pada 3 Desember 2009. Perubahannya antara lain : pengoptimalan *hardware*, peningkatan *Google Maps* 3.1.2, perubahan UI

dengan *browser* baru dan dukungan *HTML5*, daftar kontak yang baru, dukungan *flash* untuk kamera 3,2 MP, *digital Zoom*, dan *Bluetooth* 2.1. *Android Eclair* adalah *Android* pertama yang mulai dipakai oleh banyak *smartphone*, fitur utama *Eclair* yaitu perubahan total struktur dan tampilan *user interface*.

6. *Android* Versi 2.2 *Froyo* (*Frozen Yogurt)*

   Android Froyo dirilis pada 20 Mei 2012. Versi ini memiliki kecepatan kinerja dan aplikasi 2 sampai 5 kali dari versi-versi sebelumnya. Selain itu ada penambahan fitur-fitur baru seperti dukungan *Adobe Flash* 10.1, integrasi V8 *JavaScript engine* yang dipakai *Google Chrome* yang mempercepat kemampuan randering pada *browser*, pemasangan aplikasi dalam *SD Card*, kemampuan *Wifi Hotspot Portabel*, dan kemampuan *auto update* dalam aplikasi *Android Market*.

7. *Android* Versi 2.3 *Gingerbread*

   Android Gingerbread di rilis pada 6 Desember 2010. Perubahan umum yang didapat dari *Android* versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antar muka (*user interface*) didesain ulang, dukungan format video VP8 dan WebM, efek audio baru (*reverb, equalization, headphone virtualization,* dan *bass boost*), dukungan kemampuan *Near Field Communication* (NFC), dan dukungan jumlah kamera yang lebih dari satu.

8. *Android* Versi 3.0/3.1 *Honeycomb*

   Android Honeycomb di rilis pada awal 2012. Versi *Android* yang dirancang khusus untuk *device* dengan layar besar seperti *Tablet PC*. Fitur baru pada *Android Honeycomb* antara lain dukungan terhadap *processor multicore* dan grafis dengan *hardware acceleration*. *User interface* pada *Honeycomb* juga berbeda karena sudah didesain untuk tablet. Tablet pertama yang memakai *Honeycomb* adalah tablet *Motorola Xoom* yang dirilis bulan Februari 2011.

9. *Android* Versi 4.0 *Ice Cream Sandwich*

   Android Ice Cream Sandwich diumumkan secara resmi pada 10 Mei 2011 pada ajang *Google I/O Developer Conference* (*San Fransisco*), pihak *Google*

mengklaim *Android Ice Cream Sandwich* akan dapat digunakan baik di *smartphone* ataupun tablet. *Android Ice Cream Sandwich* membawa fitur *Honeycomb* untuk *smartphone* serta ada penambahan fitur baru seperti membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari *email* secara *offline*, dan berbagi informasi dengan menggunakan NFC. Ponsel pertama yang menggunakan sistem operasi ini adalah *Samsung Galaxy Nexus*.

10. *Android* Versi 4.1 *Jelly Bean*

    Android Jelly Bean juga diluncurkan pada acara *Google I/O* 10 Mei 2011 yang lalu. Keunggulan dan fitur baru versi ini, diantaranya peningkatan *input keyboard*, desain baru fitur pencarian, *UI* yang baru dan pencarian melalui *Voice Search* yang lebih cepat. Sistem operasi *Android Jelly Bean* 4.1 pertama kali digunakan dalam produk tablet *Asus*, yakni *Google Nexus* 7.

11. *Android* versi 4.2 *Jelly Bean*

    Fitur *photo sphere* untuk panaroma, *daydream* sebagai *screensaver*, *power control*, *lock screen widget*, menjalankan banyak *user* (dalam tablet saja), *widget* terbaru. *Android* 4.2 pertama kali dikenalkan melalui *LG Google Nexus* 4.

12. *Android* Versi 4.3 *Jelly Bean*

    Google merilis *Jelly Bean* 4.3 pada 24 Juli 2013 di *San Fransisco*. *Nexus* 7 generasi kedua adalah perangkat pertama yang menggunakan sistem operasi ini. Sebuah pembaruan minor dirilis pada tanggal 22 Agustus 2013.

13. *Android* Versi 4.4 *Kitkat*

    Google mengumumkan *Android* 4.4 *KitKat* (dinamai dengan izin dari *Nestle* dan *Hershey)* pada 3 September 2013, dirilis pada tanggal 31 Oktober 2013. Keunggulannya diantaranya pembaruan antarmuka dengan bar status dan navigasi transparan pada layar depan, optimasi kinerja dengan spesifikasi perangkat yang lebih rendah, *NFC Host Card Emulation* sebagai *emulator* kartu pintar, *WebViews* berbasis *Chromium*, *Sensor batching*, *Step Detector*, dan *Counter API*, peningkatan tampilan mode layar penuh, penyeimbang

audio, pemantauan audio, dan peningkatan suara audio, dukungan *Bluetooth Message Acces Profile (MAP).*

2.1.5. Pengertian Aplikasi Informasi Dakwah (*E-Text*) Berbasis *Android* pada Kantor Wilayah Kementerian Agama Provinsi Sumatera Selatan

Aplikasi Informasi Dakwah (e-*Text*) Berbasis *Android* pada Kantor Wilayah Kementerian Agama Provinsi Sumatera Selatan merupakan aplikasi yang dapat digunakan untuk mendapatkan informasi mengenai kegiatan dakwah yang terjadi di Sumatera Selatan khususnya Kota Palembang, informasi tersebut berupa naskah dakwah yang berbentuk e-*Text*, nama penceramah, dan juga lokasi tempat dakwah tersebut berlangsung serta jadwal sholat untuk Palembang dan sekitarnya.

2.2.Teori Khusus

2.2.1. Pemrograman Berorientasi Objek

2.2.1.1. Pengertian Pemrograman Berorientasi Objek

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*)

Asropudin (2013:102), *UML* singkatan dari *Unified Modelling Language* ini adalah bahasa pemrograman yang digunakan untuk perangkat lunak berorientasi objek. 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.

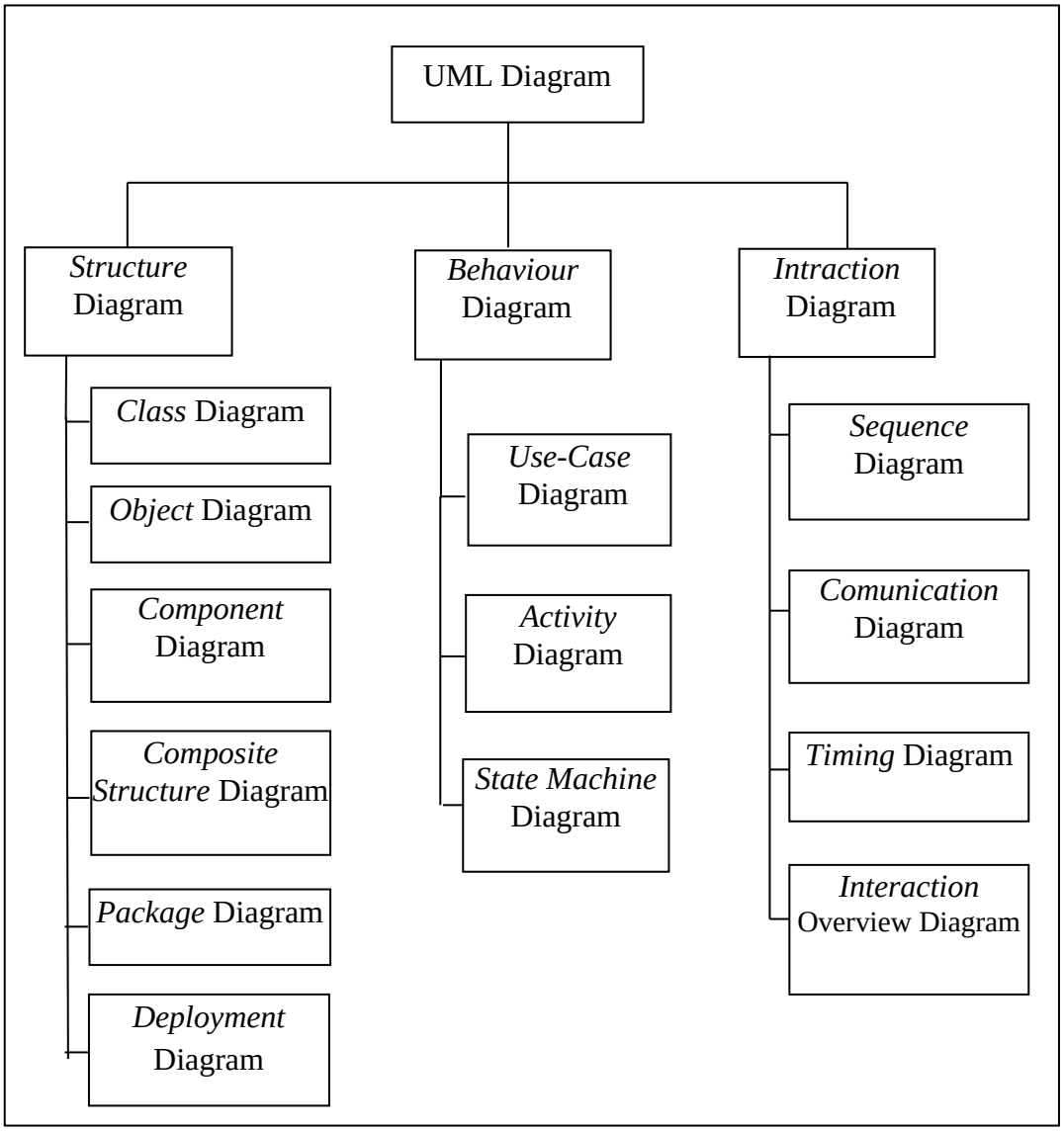
b. *Behavior* Diagram

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:



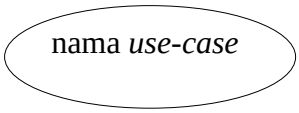
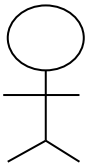
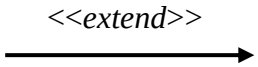
(*Sumber*: Sukamto dan Shalahuddin (2016:140))

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

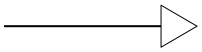
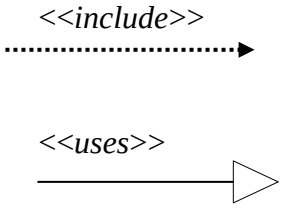
2.2.4. *Use-Case* Diagram

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

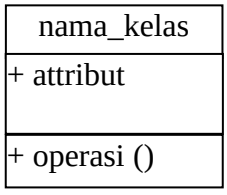
No	Simbol	Deskripsi
5.	Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use-case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
6.	Menggunakan / *include* /*uses* 	Relasi *use-case* tambahan ke sebuah *use-case* di mana *use-case* yang ditambahkan memerlukan *use-case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use-case*.

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

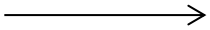
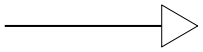
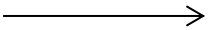
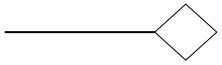
2.2.5. *Class* Diagram

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 	Kelas pada struktur sistem.
2.	Antarmuka / *interface*  nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.

Lanjutan Tabel 2.2. Simbol-simbol *Class* Diagram

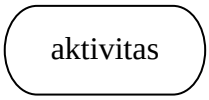
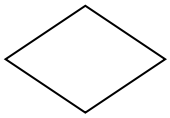
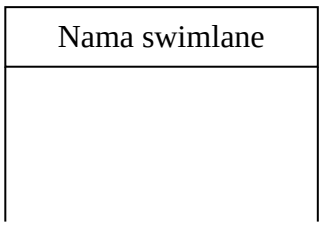
No	Simbol	Deskripsi
3.	Asosiasi / *association*	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
4.	Asosiasi berarah / *directed association* 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity.*
5.	Generalisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus).
6.	Kebergantungan/ d*ependency* 	Relasi antarkelas dengan makna kebergantungan antarkelas.
7.	Agregasi / *aggregation* 	Relasi antarkelas dengan makna semua-bagian *(whole-part)*.

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

2.2.6. *Activity Diagram*

Sukamto dan Salahuddin (2016:161), *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

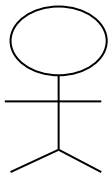
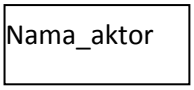
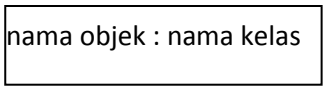
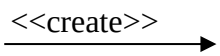
No	Simbol	Deskripsi
1.	Status Awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih 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))

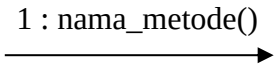
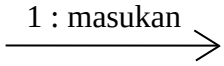
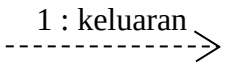
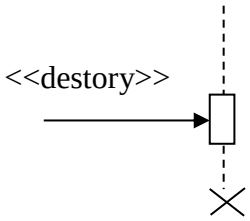
2.2.7. *Sequence Diagram*

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.

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.
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.

Lanjutan Tabel 2.4. Simbol-simbol *Sequence Diagram*

No	Simbol	Deskripsi
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.
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 (2013:272), 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. Sihab (2011), metode pengujian adalah cara atau teknik untuk menguji perangkat lunak, mempunyai mekanisme untuk menentukan data uji yang dapat menguji perangkat lunak secara lengkap dan mempunyai kemungkinan tinggi untuk menemukan kesalahan. Pengujian perangkat lunak perlu dilakukan untuk mengevaluasi baik secara manual maupun otomatis untuk menguji apakah

perangkat lunak sudah memenuhi persyaratan atau belum, dan untuk menentukan perbedaan antara hasil yang diharapkan dengan hasil sebenarnya.

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)

Shihabuddin (2014), *black-box testing* merupakan pengujian yang berfokus pada fungsional dari perangkat lunak, tester dapat mendefinisikan kumpulan kondisi input dan melakukan pengetesan pada spesifikasi fungsional program. 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 Program

2.3.1. Pengertian Basis Data (*Database*)

Sujatmiko (2012:40), basis data (*database*) adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data. Pratama (2014:17), elemen basis data pada sistem informasi berfungsi sebagai media untuk penyimpanan data dan informasi yang dimiliki oleh sistem informasi bersangkutan.

Berdasarkan beberapa pendapat di atas dapat disimpulkan bahwa basis data merupakan kumpulan data yang disimpan secara sistematis di dalam komputer yang dapat diolah untuk menghasilkan informasi.

2.3.2. Pengertian *MySQL*

Nugroho (2013:94), *MySQL* adalah *software* atau program *database server*. Madcoms (2012:282), *MySQL* adalah salah satu *database* untuk *server* yang menggunakan *SQL* yang bersifat *free* (gratis atau tidak perlu membayar untuk menggunakannya), selain itu dapat berjalan di berbagai *platform* antara lain *Linux* dan *Windows*.

Berdasarkan kedua pendapat di atas dapat disimpulkan bahwa MySQL adalah sebuah *software* (perangkat lunak) yang digunakan untuk mengelola sebuah *database*.

2.3.3. Java

Kadir (2013:4), ekstensi .java digunakan pada kode sumber bahasa java. Contoh kode sumber bahasa java ditunjukkan di bawah ini:

```
package com.example.aplikasipertama;
import Android.os.Bundle;
```

```
import Android.app.Activity;
import Android.view.Menu;
public class MainActivity extends Activity {
        @Override
        protected void onCreate (Bundle savedInstanceState) {
                super.onCreate (savedInstanceState);
                setContentView (R.layout.activity_main);
        }
        @Override
        public boolean onCreateOptionsMenu (Menu menu) {
                getMenuInflater().inflate (R.menu.main, menu);
                return true;
        }
}
```

2.3.4. XML

Kadir (2013:5), ekstensi .xml digunakan pada data yang menggunakan format *eXtended Markup Language* (XML). Contoh data degan format XML ditunjukkan di bawah ini.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="app_name"> Aplikasi Pertamaku </string>
        <string name="action_settings"> Settings </string>
        <string name="hello_world"> Selamat Belajar Pemrograman Android!
        </string>
</resources>
```

File .xml diawali dengan baris semacam berikut:

```
<?xml version="1.0" encoding="utf-8"?>
```

Data yang terkandung dalam *file* .xml sering kali dinyatakan dalam pasangan tag. Dalam hal ini, suatu tag (menyatakan suatu elemen) dapat mengandung banyak tag. Pada contoh di atas, pasangan tag <string> dan </string> menyatakan sebuah data. Tiga pasangan <string> dan </string> terdapat pada pasangan tag <resource> dan </resource>.

Adakalanya, elemen XML tidak mempunyai pasangan tag penutup. Pada keadaan seperti ni, elemen akan diakhiri dengan />. Pada contoh berikut, TextView diakhiri dengan />:

```
<TextView
    Android:id="@+id/textViewNegara"
    Android:layout_width="@+id/textViewNegara"
    Android: layout_width ="@+id/textViewNegara"
    Android:text="Negara:" />
```

Suatu tag dapat mengandung satu atau beberapa atribut. Pada tag <TextView> di atas, *Android*:id dan *Android*:layout_width adalah nama atribut, sedangkan "@+id/textViewNegara" dan "wrap_content" adalah nilai atribut.