BAB 2
LANDASAN TEORI

2.1 Aplikasi

Perangkat lunak atau piranti lunak adalah istilah khusus untuk data yang diformat, dan disimpan secara digital, termasuk program komputer, dokumentasinya, dan berbagai informasi yang bisa dibaca, dan ditulis oleh komputer. Dengan kata lain, bagian sistem komputer yang tidak berwujud. Istilah ini menonjolkan perbedaan dengan perangkat keras komputer. Pembuatan perangkat lunak itu sendiri memerlukan "bahasa pemrograman" yang ditulis oleh programmer untuk selanjutnya di kompilasi dengan kompiler sehingga menjadi kode yang bisa dikenali oleh mesin hardware. Dan salah satu contoh dari maca perangkat lunak yaitu perangkat lunak aplikasi (*application software*) seperti pengolah kata, lembar tabel hitung, pemutar media, dan paket aplikasi perkantoran seperti OppenOffice.org. [1]

Perangkat lunak aplikasi yaitu perangkat lunak yang digunakan untuk membantu pemakai komputer untuk melaksanakan pekerjaannya. Jika ingin mengembangkan program aplikasi sendiri, maka untuk menulis program aplikasi tersebut, dibutuhkan suatu bahasa pemrograman yang dapat berbentuk *assembler*, *compiler* ataupun *interpreter*. [2]

Perangkat lunak adalah instruksi langsung komputer untuk melakukan pekerjaan dan dapat ditemukan disetiap aspek kehudipuan modern dari aplikasi yang kritis untuk hidup (*life-critical*), seperti perangkat lunak pemantauan medis dan pembangkit tenaga listrik sampai perangkat hiburan, seperti video game. Banyak produk perangkat lunak berisi jutaan baris kode yang diharapkan dapat melakukan pekerjaan dengan baik dalam menghadapi perubahan kondisi. Semua perangkat lunak juga membutuhkan keandalan yang tinggi yang harus dihasilkan secara ekonomis.[3]

Teknik rekayasa perangkat lunak akan meningkatkan fungsionalitas dan efisiensi aplikasi dan juga kemudahan dan efisiensi dari pengembangan perangkat

lunak. Komunitas rekayasa perangkat lunak memiliki 630.000 praktisi dan para pendidik di Amerika Serikat, dan diperkirakan ada 1.400.000 praktisi di Uni Eropa, Asia, dan di tempat lain, jumlah tersebut sekitar 60% ukuran rekayasa tradisional. Pelopor rekayasa perangkat lunak adalah Barry Boehm, Fres Brooks, CAR Hoare, dan David Parnas.[4]

Sejalan dengan semakin luasnya PC dan jaringan komputer diera ini, perangkat lunak juga berkembang untuk memenuhi kebutuhan perseorangan. Perangkat lunak dapat dibedakan menjadi perangkat lunak sistem yang bertugas menangani sisi internal dan perangkat lunak aplikasi yang digunakan secara langsung oleh penggunanya untuk keperluan tertentu. Otomatisasi yang ada didalam perangkat lunak mengarah pada suatu jenis kecerdasan buatan. [1]

Saat ini perangkat lunak sudah terdapat dimana-mana, tidak hanya pada sebuah superkomputer dengan 25 prosesornya, sebuah komputer genggam pun telah dilengkapi dengan perangkat lunak yang dapat disinkronkan dengan PC. Tidak hanya komputer, bahkan peralatan, seperti telepon, TV, mesin cuci, AC, dan oven, telah ditanami perangkat lunak untuk mengatur operasi peralatan itu. Lebih hebat lagi, setiap peralatan itu mungkin suatu saat kelak akan dapat saling terhubung. Pembuatan sebuah perangkat lunak bukan lagi pekerjaan segelintir orang, tetapi telah menjadi pekerjaan banyak orang dengan beberapa tahapan proses yang melibatkan berbagai disiplin ilmu dalam perancangannya. Tingkat kecerdasan yang ditunjukkan oleh perangkat lunak pun semakin meningkat, selain permasalahan teknis, perangkat lunak sekarang mulai mengenal suara dan gambar.[1]

Beberapa buku referensi menyatakan bahwa perancangan aplikasi lebih diasumsikan sebagai perancangan yang melibatkan logika modul-modul yang akan dibuat dalam perangkat lunak. Sedangkan referensi lainnya melibatkan perancangan mengenai data didalam tahapan ini. Kedua pendapat tersebut tidaklah salah, karena untuk perangkat lunak berjenis sistem informasi perancangan mengenai data sangatlah vital untuk dilakukan, tetapi untuk jenis perangkat lunak non sistem informasi, perancangan data seringkali diabaikan. [4]

Perancangan aplikasi lebih diasumsikan sebagai perancangan yang melibatkan logika modul-modul yang akan dibuat dalam perangkat lunak. [4] Perancangan mengenai aplikasi melibatkan perancangan logika dan algoritma dari suatu perangkat lunak. Dengan adanya perancangan secara logika, maka notasi yang digunakan lebih banyak ke orrientasi untuk penggambaran diagram alir (*flowchart*). Secara detail pula, dalam perancangan ini dijabarkan mengenai kendala atau *constraint* yang harus diatasi didalam perangkat lunak. Apabila perangkat lunak yang dibuat merupakan jenis sistem informasi, maka *constraint* tersebut merupakan hasil dari analisa kebutuhan sistem yang sebelumnya telah dikerjakan dalam sebuah organisasi. [4]

2.2 Perancangan Antar Muka

Perancangan antar muka atau perancangan interface secara detail dibahas di disiplin ilmu HCI (*Human Computer Interaction*) atau IMK (Interaksi Manusia dan Komputer). Meski demikian, akan dibahas secara sekilas mengenai aspek *interface* dalam sebuah tahapan perancangan perangkat lunak. [4]

Interaksi manusia dan komputer bertujuan untuk mengembangkan keamanan, utilitas, efektivitas, efisiensi dan *usability* dari sistem yang memakai komputer serta memberikan pedoman bagi para desainer dalam mendesain sistem yang *usable*. Dalam kaitannya, interaksi manusia dan komputer merupakan pendukung utama dari perancangan antar muka. Sebab perancangan antar muka bertujuan untuk menjembatani antara kepentingan pengguna dengan perangkat lunak yang akan dibuat. [4]

Yang dimaksud dengan istilah *interface* atau antar muka sendiri adalah *the part of the system that you see, hear and feel.* Jadi antar muka adalah bagian dari perangkat lunak yang dapat dirasakan oleh panca indera pengguna baik dari sisi penglihatam pendengaran maupun diraba bahkan juga dapat dicium (untuk perangkat lunak tertentu). [4]

2.3 Smartphone

Menurut Williams dan Sawyer *smartphone* adalah telepon selular dengan mikroprosesor, memori, layar dan modem bawaan. *Smartphone* merupakan ponsel multimedia yang menggabungkan fungsionalitas *Personal Computer* (PC) dan handset sehingga menghasilkan *gadget* yang mewah dimana terdapat pesan teks, kamera, pemutar musik, video, *game*, akses email, tv digital, *search engine*, pengelola informasi pribadi, fitur GPS, jasa telepon internet, dan bahkan terdapat telepon yang juga berfungsi sebagai kartu kredit. [5]

Didalam penelitian ini smartphone digunakan untuk membantu aplikasi dalam mengakses perangkat keras melalui *Application Programming Interface* (API) yang terdapat pada sebuah sistem operasi Android untuk kemudian dapat memberikan *data monitoring*.

2.4 Android

Android adalah sebuah kumpulan perangkat lunak yang bersifat open source yang mencakup sistem operasi, *middleware*, dan aplikasi kunci bersama dengan satu set *Application Programming Interface* (API) untuk menulis aplikasi *mobile* yang dapat membentuk tampilan, rasa, dan fungsi dari sebuah perangkat *mobile* [6]. Awalnya, Google Inc. membeli Android Inc yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone*. Para pendiri Android Inc. bekerja pada Google untuk memulai membangun *platform* Android secara lebih intensif, di antaranya Andy Rubin, Rich Miner, Nick Sears, dan Chris White. Pada tanggal 12 November 2007 Google bersama *Open Handset Aliance* (OHA) merilis Google *Android Software Development Kit* (SDK). Paket SDK yang dirilis untuk mengembangkan aplikasi yaitu sistem operasi, *middleware,* dan aplikasi utama untuk perangkat mobile. Dengan dirilisnya SDK tersebut membuka kesempatan kepada pengembang untuk mengembangkan sebuah aplikasi berbasis Android [7].

Hingga awal tahun 2017 terdapat beberapa versi Android yang tersedia untuk digunakan, diantaranya adalah sebagai berikut:

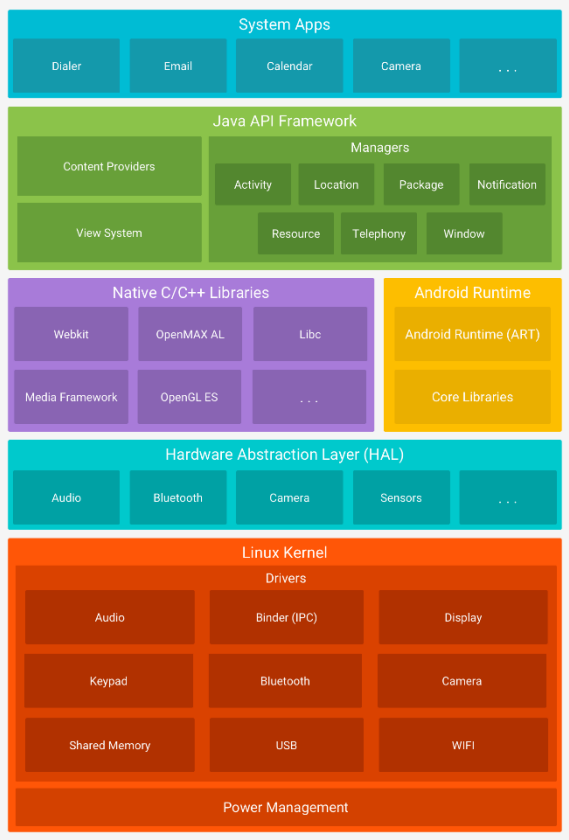
1. Android versi 1.1
2. Android versi 1.5 Cupcake

3. Android versi 1.6 Donut
4. Android versi 2.0/2.1 Éclair
5. Android versi 2.2 Froyo
6. Android versi 2.3 GingerBread
7. Android versi 3.0/3.1/3.2 Honeycomb
8. Android versi 4.0 Ice Cream Sandwich
9. Android versi 4.1.2/4.2.2/4.3.1 Jelly Bean
10. Android versi 4.4.2 Kitkat
11. Android versi 5.0.1/5.1.1 Lollipop
12. Android versi 6.0 Marshmallow
13. Android versi 7.0/7.1.1.1 Nougat

Didalam penelitian ini peneliti memilih versi sistem operasi Android Lollipop (API Level 22) ke atas karena pada sistem operasi tersebut telah terdapat interface untuk membaca ID unik pada banyak slot kartu SIM.

2.5 Arsitektur Android

Secara garis besar komponen utama yang terdapat pada arsitektur Android dapat dilihat pada Gambar 2.1.



Gambar 2.1 Arsitektur Android

1. Kernel Linux

kernel adalah *layer* dimana inti dari sistem operasi Android berada. Pada linux kernel terdapat *file-file* sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem-sistem operasi Android lainnya. Dengan menggunakan linux kernel akan membolehkan Android untuk memperoleh keuntungan fitur kunci keamanan dan membolehkan manufaktur perangkat untuk mengembangkan *driver* perangkat keras untuk kernel umum [7].

2. *Hardware Abstraction Layer* (HAL)

Hardware Abstraction Layer menyediakan *interface* standar yang mendukung kapabilitas hardware sebuah perangkat kedalam tingkatan *framework* API Java yang lebih tinggi. HAL berisi beberapa modul *library* yang mengimplementasikan *interface* masing-masing untuk jenis komponen *hardware* khusus seperti modul kamera atau *Bluetooth*. Ketika *framework* API membuat

pangilan untuk mengakses *hardware*, sistem Android memuat modul *library* untuk komponen *hardware* tersebut. [7]

3. Android Runtime

Android Runtime adalah *layer* yang membuat aplikasi Android dapat dijalankan dimana dalam prosesnya menggunakan implementasi Linux. Untuk perangkat yang berjalan pada versi Android 5.0 (API 21) atau keatas, setiap aplikasi berjalan didalam proses dan instansi *Android Runtime* (ART) masing-masing. ART ditulis untuk menjalankan beberapa *virtual machine* pada perangkat memori rendah dengan mengeksekusi *file* DEX. Berdasarkan Android versi 5.0 (API 21), *Dalvik* dulunya adalah *Android Runtime*. Apabila aplikasi berjalan baik pada ART, maka seharusnya berjalan baik pula pada *Dalvik* [8].

4. *Native* C/C++ *Libraries*

Beberapa komponen dan *service* dari sistem android seperti ART dan HAL dibuat dengan kode *native* yang membutuhkan *native library* yang ditulis dalam bahasa C dan C++. *Platform Android* menyediakan *framework* API Java untuk memaparkan fungsionalitas dari *native library* tersebut terhadap aplikasi [7].

5. Java API *Framework*

Seluruh set fitur pada sistem operasi Android tersedia untuk para pengembang melalui API yang ditulis dalam bahasa Java. Dalam hal ini, pengembang memiliki akses penuh terhadap *framework* API yang sama yang digunakan oleh aplikasi pada sistem Android. [9] API-API tersebut menyediakan kebutuhan dalam membangun sebuah aplikasi yang mencakup :

1. Sistem *View* yang kaya dan *extensible* yang dapat digunakan untuk membuat UI aplikasi, seperti *list*, *grid*, *text box*, *button*, dan sebagainya.
2. *Resource Manager* yang menyediakan akses terhadap sumbe rdaya *noncode* seperti *localized string*, *graphic*, dan file *layout*.
3. *Notification Manager* yang membolehkan aplikasi untuk menampilkan kustom *alert* didalam *status bar*.

4. *Activity Manager* yang mengatur *lifecycle* sebuah aplikasi dan menyediakan *navigasi back stack* umum.
5. *Content Provider* yang membolehkan aplikasi untuk mengakses data dari aplikasi lain seperti aplikasi kontak atau untuk membagikan data aplikasi itu sendiri.

6. System Apps

Android dibuat dengan satu set aplikasi bawaan seperti *email*, SMS *messaging*, *calendar*, *internet browsing*, *contacts*, dan lain lain. Aplikasi yang disertakan oleh *platform* tidak memiliki status spesial. Aplikasi pihak ketiga yang diinstal oleh pengguna dapat saja menjadi aplikasi *default* pengguna. Fungsi dari *system apps* adalah bertindak sebagai aplikasi untuk pengguna dan juga penyedia kapabilitas kunci untuk pengembang yang ingin mengakses melalui aplikasi miliknya sendiri [7].

Berdasarkan komponen arsitektur Android diatas diketahui bahwa sistem aplikasi yang dikembangkan oleh pengembang aplikasi dapat mengakses API dari lapisan *framework Java Android*. Dengan demikian aplikasi yang dikembangkan dapat digunakan mengakses *hardware* pada perangkat Android.

2.6 Komponen Aplikasi Android

Aplikasi Android ditulis dalam bahasa pemrograman Java. Kode Java dikompilasi bersama dengan data file yang dibutuhkan oleh aplikasi dimana prosesnya dikemas oleh tools yang dinamakan "*APT tools*" kedalam paket Android sehingga menghasilkan *file* dengan ekstensi apk. *File* apk itulah yang kita sebut dengan aplikasi dan nantinya dapat dipasang pada perangkat *mobile*. [7]

Terdapat empat jenis komponen pada aplikasi Android yaitu:

1. *Activity*

Activity akan menyajikan *User Interface* (UI) kepada pengguna sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi Android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak ac*t*ivity yang bergantung pada tujuan aplikasi dan desain aplikasi itu sendiri. Untuk berpindah

dari satu *activity* ke *activity* lain dapat dilakukan menggunakan sebuah *trigger* seperti klik tombol pada tampilan aplikasi.

2. *Service*

Service tidak memiliki *Graphic User Interface* (GUI), tetapi *service* berjalan secara *background*. Sebagai contoh dalam memutar musik, *service* mungkin memutar musik atau mengambil data dari jaringan, tetapi setiap *service* harus berada dalam kelas induknya. Apabila sebuah pemutar musik sedang memutar lagu dari list yang ada, aplikasi akan memiliki dua atau lebih *activity* yang memungkinkan pengguna untuk memutar sambil memilih lagu baru. Untuk menjaga musik tetap berjalan sebuah *activity* dapat menjalankan *service*.

3. *Broadcast Receiver*

Broadcast receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. Contoh broadcast seperti notifikasi zona waktu telah berubah, baterai lemah, gambar berhasil diambil oleh kamera, dan lain-lain. Aplikasi juga dapat menginisialiasi *broadcast* misalnya memberikan informasi pada aplikasi lain bahwa ada data yang telah diunduh ke perangkat dan siap untuk digunakan.

4. Content Provider

Content provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam *file* sistem seperti database SQLite. *Content provider* menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity.*

Didalam penelitian ini menggunakan seluruh komponen aplikasi diatas. *Activity* digunakan peneliti agar dapat memberikan Antar Muka bagi pengguna aplikasi, *service* digunakan untuk proses yang bergerak tanpa Antar Muka, *broadcast receiver* digunakan untuk membantu mendeteksi SMS masuk atau terjadi pergantian kartu SIM, dan *content provider* digunakan untuk membantu melakukan *query* terhadap data seperti kontak pada kartu SIM.

2.7 *Android* SDK (*Software Development Kit*)

Android SDK (*Software Development Kit*) adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi

pada *platform Android*. *Android* SDK disediakan sebagai alat bantu untuk mulai mengembangkan aplikasi pada platform *Android* menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi-netral, *Android* memberi kesempatan untuk membuat aplikasi yang dibutuhkan. [7]

Android SDK versi resmi dapat ditemukan di situs resmi Google. Namun terdapat juga SDK buatan pihak ketiga seperti App *Inventor* untuk *Android*, Basic4android, Corona SDK, Ruby Motion, dan Xamarin. SDK tersebut dapat digunakan untuk menulis program Android, namun tentunya bahasa pemrograman yang digunakan menyesuaikan dengan SDK masing-masing.

Setiap kali Google merilis versi baru dari *Android* maka sebuah SDK yang sesuai juga dirilis. Untuk dapat menulis program dengan fitur terbaru maka pengembang aplikasi *Android* harus mengunduh dan menginstal *Android* SDK sesuai versi masing-masing untuk *smartphone* tertentu. Didalam penelitian ini penulis menggunakan *Android* SDK resmi dari Google dengan versi minimum 25.

2.8 *Android Studio*

Android Studio adalah *Integrated Development Environment* (IDE) untuk pengembangan aplikasi *Android*, berdasarkan IntelliJ IDEA. Selain merupakan *editor* kode IntelliJ dan alat pengembang yang berdaya guna, *Android Studio* menawarkan fitur lebih banyak untuk meningkatkan produktivitas saat membuat aplikasi *Android*, misalnya:

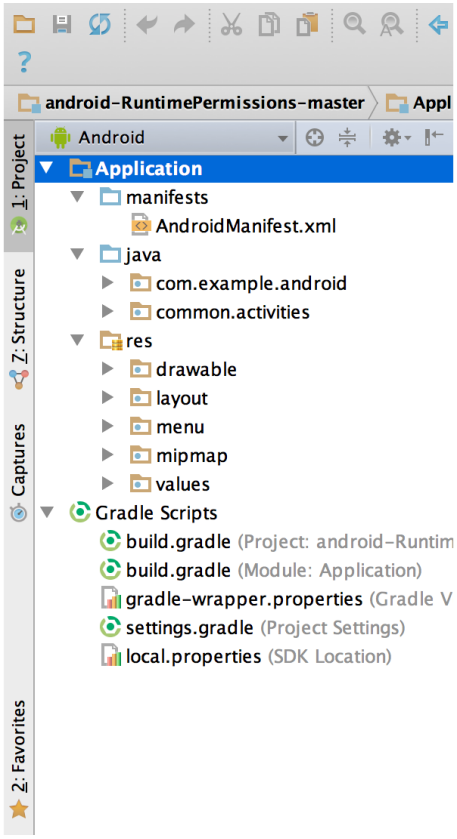
1. Sistem pembuatan berbasis *Gradle* yang fleksibel
2. *Emulator* yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat *Android*
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
6. Alat penguji dan kerangka kerja yang ekstensif

7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain
8. Dukungan C++ dan NDK
9. Dukungan bawaan untuk Google *Cloud Platform*, mempermudah pengintegrasian Google *Cloud Messaging* dan *App Engine*

Setiap proyek di *Android Studio* berisi satu atau beberapa modul dengan *file* kode sumber dan *file* sumber daya. Jenis-jenis modul mencakup:

1. Modul aplikasi *Android*
2. Modul perpustakaan
3. Modul Google *App* Engine

Secara default, *Android Studio* menampilkan *file* proyek Anda dalam tampilan proyek *Android* seperti yang ditunjukkan dalam gambar 2.2. Tampilan ini diatur menurut modul untuk memberi akses cepat ke *file* sumber kunci proyek Anda.



Sumber Gambar : https://developer.android.com/studio/intro/index.html

Gambar 2.2 File Proyek di Tampilan Android

Semua *file* versi terlihat di bagian atas di bawah *Gradle Scripts* dan masing-masing modul aplikasi berisi folder berikut:

1. *manifests* yang berisi file AndroidManifest.xml.
2. java yang berisi file kode sumber Java, termasuk kode pengujian JUnit.
3. res: Berisi semua sumber daya bukan kode, seperti tata letak XML, string UI, dan gambar *bitmap*.

Struktur proyek *Android* pada disk berbeda dari representasi rata ini. Untuk melihat struktur *file* sebenarnya dari proyek ini, pilih *Project* dari menu tarik turun *Project* (dalam gambar 2.2, struktur ditampilkan sebagai *Android*).

Android Studio digunakan peneliti dibandingkan IDE lain karena *Android Studio* telah memiliki dukungan penuh dari Google. Hal tersebut memudahkan pengembang dalam melakukan integrasi seperti penggunaan sebuah *library* yang melalui *file gradle* cukup dengan mendaftarkan satu baris program saja. Adapun versi *Android studio* yang digunakan peneliti yaitu versi 2.3.2.

2.9 Bahasa Pemograman Java

Java bermula dari proyek penelitian perusahaan Sun Microsystems dengan nama sandi *Green* pada tahun 1991. Terdapat prediksi bahwa *microprosesor* akan digunakan luas pada peralatan-peralatan elektronik. Karena adanya bermacam tipe mikroprosesor, maka dibutuhkan sebuah bahasa pemrograman yang dapat berjalan disemua mikroprosesor. Terciptalah sebuah bahasa pemrograman baru. Oleh James Gosling, yaitu salah satu orang yang berperan besar dalam proyek tersebut, program ini diberi nama Oak. Sesuai dengan pohon Oak yang tumbuh dan bisa dilihat melalui jendela kerjanya di Sun Microsystems. [9]

Selang beberapa tahun kemudian, ditemukan bahwa sudah ada bahasa pemrograman dengan nama Oak. Akhirnya setelah beberapa pegawai Sun mengunjungi sebuah kedai kopi, nama bahasa pemrograman ini diganti dengan Java. Java merupakan salah satu jenis kopi yang ada di kedai tersebut, yaitu biji kopi Jawa. [9]

Sun Microsystems mengumumkan kehadiran bahasa Java secara formal ditahun 1995. Bahasa ini mulai disambut hangat masyarakat luas seiring dengan meledaknya era internet. [9]

Java merupakan perangakat lunak yang cepat populer, karena perangkat lunak ini dapat berjalan diberbagai *platform*, mudah dipelajari, dan *powerful*. Aplikasi yang dapat dibuat dengan perangkat lunak ini adalah aplikasi berbasis PC, berbasis *web* dan selular. [10]

Perangkat lunak java dibangun atau dibuat dengan bahasa pemrograman C++, tetapi mengakomodasi teknologi terbaik dari bahasa pemrograman C dan memperbaiki kekurangan bahasa pemrograman C++. Java merupakan perangkat lunak *open source* yang andal dan portable untuk beberapa macam sistem operasi, seperti Window, Linux, Solaris, dan lainnya. Java juga termasuk bahasa pemrograman *Multithreading* (dapat mengerjakan beberapa proses dalam waktu yang bersamaan). Perangkat lunak Java juga secara oromatis menangani sisa memory yang tidak terpakai/sampah (*garbage*). [10]

Java telah berkembang dari semula ditujukan untuk pemrograman *applet* di *web browser* menjadi bahasa pemrograman pengembangan aneka ragam aplikasi, mulai dari yang berjalan di *handheld devices* seperti *handphone*, PDA, sampai aplikasi tersebar skala *enterprise* di beragam komputer *server*. Java merupakan bahasa orientasi objek untuk pengembangan aplikasi mandiri, aplikasi berbasis internet, aplikasi untuk perangkat cerdas yang dapat berkomunikasi lewat internet/jaringan komunikasi. Java tidak lagi hanya bahasa untuk membuat *applet* yang memperindah halaman web tapi Java telah menjadi bahasa untuk pengembangan aplikasi skala *enterprise* berbasis jaringan besar. [11]

Menurut Budi Raharjo java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. [12] Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis Java umumnya dikompilasi ke dalam p-code

(*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi Java mampu berjalan di beberapa *platform* sistem operasi yang berbeda, Java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Saat ini Java merupakan bahasa pemrograman yang paling populer digunakan, dan secaraluas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis *web*. [12]

Java bukan merupakan satu-satunya bahasa pemrograman yang dapat digunakan untuk mengembangkan aplikasi berbasis Android. Adapun bahasa lain seperti Xamarin yang menggunakan bahasa pemgrograman C# yang dapat digunakan sebagai opsi alternatif dalam membangun aplikasi berbasis *Android*. Namun disini peneliti menggunakan bahasa pemrograman Java karena lebih mudah dalam mencari contoh dokumentasi dan merupakan bahasa yang didukung oleh IDE *Android Studio* resmi dari Google.

Sebuah program java, khususnya aplikasi yang berinteraksi dengan pengguna (*user*), pasti memerlukan cara agar program yang bersangkutan dapat berkomunikasi dengan para penggunanya. Saat ini, tidak seperti program-program komputer di “abad batu”, program-program komputer masa kini umumnya menggunakan berbagai antarmuka pengguna berbasis grafis (GUI [*Graphical User Interface*]) sebagai sarana berkomunikasi dengan para penggunanya. Dalam hal ini, pemrograman berbasis grafis masa pada umumnya dikembangkan pada suatu teknik pemrograman canggih yang secara umum dinakaman sebagai pemrograman berbasis *event* (*event driven programming*)-perilaku program/aplikasi akan ditentukan oleh tindakan-tindakan yang dilakukan oleh pengguna program. [4]

2.10 Pengembangan Aplikasi *Mobile*

Perkembangan aplikasi *mobile* sangat bergantung terhadap perkembangan *smartphone*, baik dari sisi teknologi maupun jumlah produksi. Tentu saja, karena *smartphone* merupakan *platform* yang menjalankan aplikasi-aplikasi *mobile*. [13]

Dalam waktu yang sangat singkat, pertumbuhan smartphone menunjukkan angka yang signifikan. Menurut hasil survei lembaga penelitian Nielsen yang dilakukan pada pertengahan tahun 2012, sebanyak 67% dari total responden yang dilibatkan merupakan pengguna *smartphone*. Jumlah ini diprediksi akan terus bertambah seiring dengan antusiasme pasar. Bagaikan dua sisi mata uang, pesatnya pertumbuhan *smartphone* membawa angin segar bagi para pengembang aplikasi *mobile*. [13]

Secara garis besar, ada tiga pendekatan yang digunakan untuk mengembangan aplikasi *mobile*, yaitu aplikasi *native*, aplikasi *web*, dan aplikasi *hybrid*. [7] Aplikasi *native* adalah aplikasi yang secara khusus ditujukan untuk platform mobile tertentu dan menggunakan bahasa pemrograman serta perangkat lunak pengembang sesuai platfotm tersebut. aplikasi *web* merupakan aplikasi *website* yang secara spesifik dioptimalkan untuk penggunaan di lingkungan *smartphone*. Aplikasi ini dibangun menggunakan standar teknologi-teknologi *web*, seperti HTML5, CSS3, dan javaScript. Dan intuisi dari aplikasi *hybrid* adalah menanamkan *mobile* HTML5 kedalam kontainer *native*. Aplikasi ini berupaya mengombinasikan kelebihan-kelebihan pendekatan aplikasi *web mobile* HTML5 dan aplikasi *native*. Sederhananya, pendekatan ini mengonversi aplikasi *web* mobile HTML5 ke aplikasi *native smartphone* target. Untuk mengimplementasikan hal ini diperlukan dukungan perangkat lunak spesifik, yaitu *framework* pengembangan aplikasi *mobile*. [13]

2.11 *Web Service*

Web Service atau Layanan Web adalah serangkaian komponen piranti lunak yang bertukar informasi antara satu sama lain dengan bebas menggunakan standar komunikasi *Web* dan bahasa yang standar. Layanan *web* dapat bertukar informasi antara dua sistem yang berbeda, terlebpas dari sistem operasi atau bahasa pemrograman sistem tersebut. Layanan *web* dapat dipadukan untuk

membangun sistem penghubung aplikasi berbasis web standar terbuka dari dua organisasi yang berbeda, dan juga dapat digunakan untuk membuat aplikasi yang menghubungkan sistem terpisah dalam saru perusahaan. Layanan *Web* tidak terikat dengan sistem operasi atau bahasa pemrograman apapun, dan aplikasi-aplikasi yang berbeda dapat digunakan untuk berkomunikasi satu sama lain dengan cara standar tanpa penulisan kode khusus yang menghabiskan waktu.[14]

Teknologi dasar untuk layanan *web* adalah XML, singkatan dari *Extensible Markup Language.* Bahasa ini dikembangkan pada tahun 1996 oleh World Wide Web Consorsium (W3C, badan internasional yang mengatur perkembangan web) sebagai bahasa *markup* yang lebih tangguh dan *flexible* daripada *Hypertext Markup Language* (HTML) untuk halaman *web*. [14]

Layanan web berkomunikasi melalui pesan XML menggunakan protokol srandar untuk *web*. SOAP, singkatan dari *Simple Object Access Protocol* (protokol akses objek sederhana), adalah serangkaian protokol untuk menyusun pesan-pesan yang membuat aplikasi dapat menyampaikan data dan instruksi. WDSL, singkatan dari *Web Services Description language* (bahasa deskripsi layanan *web*), adalah sebuah kerangka kerja umum untuk menjelaskan perintah dan tugas yang dibuat oleh layanan *web* dan data yang akan diterimanya, sehingga dapat digunakan oleh aplikasi lainnya.[14]

Web service adalah layanan yang diidentifikasi dengan URI (*Uniform Resource Identifier*) yang mengekspos fiturnya melalui internet menggunakan protokol dan bahasa standar internet serta dapat diimplementasikan menggunakan standar internet seperti XML (*Extensible Markup Language*). [15][16]

Sampai dengan saat ini teknologi *web service* terus berkembang. Salah satu teknologi yang populer saat ini adalah *REST* (*Representational State Transfer*) atau terkadang disebut dengan *RESTful*. Beberapa contoh *RESTful web service* adalah *Amazon's Simple Storage Service* (S3), *Atom Publishing Protocol*, dan Google *Maps*. [16] Pada prinsipnya request ke suatu RESTful *web service* sebenarnya adalah suatu HTTP *Request*. [16]

Ada beberapa standar yang digunakan dalam *web service*. Beberapa di antaranya adalah SOAP (*Simple Object Access Protocol*), BPEL (*Business*

Process Execution Language), UDDI (*Universal Description, Discovery, and Integration Infrastructure*), WSDL (*Web Service Description Language*) [16]

PHP (PHP: *Hypertext Preprocessor*) adalah bahasa pemrograman web. PHP bisa digunakan untuk melakukan HTTP (*Hypertext Transfer Protocol*) *Request*.[16]

Maka dapat disimpulkan bahwa PHP *Web Service* bisa diimplementasikan dalam aplikasi *mobile* yang membutuhkan data dinamis. Pengujian atas *web service* bisa dilakukan dengan membuat file PHP secara manual ataupun menggunakan SOAP *web service*. Untuk memudahkan pemanggilan data bisa dilakukan modifikasi dengan memberikan *layer* tambahan berupa PHP *File* yang memanggil pada SOAP *web service*. [16]

Web service merupakan kumpulan logika aplikasi berisikan objek dan metode yang terletak didalam *web server* yang terhubung ke internet sehingga dapat diakses menggunakan protokol HTTP dan SOAP (*Simple Object Access Protocol*). *Web service* dapat digunakan untuk menyelesaikan masalah seperti aplikasi baru yang ingin memeriksa data transaksi keuangan dari sebuah *database* yang dikelola oleh aplikasi yang dibangun dengan bahasa pemrograman berbeda. Tujuan dari teknologi *web service* ini adalah untuk memudahkan beberapa aplikasi atau komponen aplikasi untuk saling berhubungan dengan aplikasi lain dalam sebuah organisasi dengan menggunakan standar yang tidak terikat *platform* dan tidak terikat akan bahasa pemrograman yang digunakan.

SOAP merupakan protokol standar yang ditujukan untuk pertukaran informasi. Protokol ini melakukan pemanggilan metode dalam bahasa XML (*Extensible Markup Language*). Selain XML, juga terdapat opsi lain yaitu menggunakan format pertukaran data JSON (*JavaScript Object Notation*). Dengan demikian SOAP adalah suatu mekanisme sederhana untuk melakukan pertukaran struktur dan tipe informasi dalam lingkungan yang tersebar dan terdistribusi menggunakan XML/JSON. Dokumen SOAP yang digunakan untuk melakukan *request* disebut SOAP request sedangkan dokumen SOAP yang diperoleh dari *web service* disebut dengan SOAP *response*. [17]

Untuk menjalankan fungsinya, *web service* memerlukan agen. Agen adalah bagian perangkat lunak atau perangkat keras yang mengirimkan dan menerima pesan. Agen dapat ditulis dengan berbagai bahasa pemrograman. Dan dapat berganti-ganti bahasa pemrograman dengan fungsi yang sama. Tujuan *web service* adalah untuk menyediakan beberapa fungsi atas nama pemiliknya, seseorang atau organisasi seperti bisnis atau perorangan. *Provider entity* adalah organisasi atau orang yang menyediakan agen yang sesuai untuk menerapkan service tertentu. *Requester entity* adalah seseorang atau organisasi yang berkeinginan untuk menggunakan *web service provider entity*. Itu akan menggunakan *requester agent* untuk menukar pesan dengan *provider agent* milik *provider entity*. [17]

Dalam pertukaran pesan agar berhasil, *requester entity* dan *provider entity* harus dulu sepakat menggunakan semantik dan mekanisme yang sama dalam pertukaran pesan. Semantik dalam *web service* adalah ekspektasi tentang perilaku (*behavior*) *service*. Ini semacam kontrak antara *requester entity* dan *provider entity* mengenai kegunaan dan konsekuensi dari interaksi tersebut. Mekanisme pertukaran pesan didokumentasikan di *Web Service Description* (WSD). WSD adalah spesifikasi mesin yang dapat berproses pada *interface web service*. menggambarkan format pesan, tipe data, protokol transpor, dan format serialisasi yang digunakan antara *requester agent* dan *provider agent*. Pada pembangunan aplikasi ini, *web service* berfungsi untuk menjembatani antara program *frontend* dan *backend*.

Peneliti menggunakan *web service* untuk ditempatkan di sebuah *server* sehingga aplikasi dapat menjalankan sistem sesuai yang diharapkan. Dengan menggunakan *web service* maka aplikasi yang dibangun dapat menyimpan data pengguna, data *monitoring*, dan sebagainya. Adapun metode yang digunakan untuk format pertukaran data adalah JSON.

Data yang didapatkan dari *web service* dikirimkan dalam format standar misalnya XML atau JSON (*Javascript Object Notation*). Dalam penelitian ini dipilih JSON. Kelebihan utama JSON dibandingkan dengan XML adalah dari sisi ukuran *file*, JSON memiliki ukuran yang lebih kecil dibandingkan dengan

XML.[18] Ukuran file yang kecil penting untuk *web service* yang akan dibuat karena nantinya data akan diakses oleh aplikasi *mobile* yang membutuhkan respon cepat. [19]

PHP mendukung untuk pengiriman maupun pengolahan data dalam format JSON. Untuk mengirimkan data dalam format JSON, PHP memiliki fungsi json_encode.[19]

Untuk menangani data JSON , PHP memiliki fungsi json_decode. SOAP merupakan layanan *web service* yang berbasis pada RPC (*Remote Procedure Call*). Kita bisa menggunakan SOAP untuk memanggil method atau fungsi yang berada di komputer lain melalui internet. Agar *client* bisa mengetahui *method* tersedia, port, format data input output, atau keterangan lain maka dideklarasikan oleh standar WSDL. [20] Kita bisa menggunakan PHP baik untuk membuat *web service server* maupun untuk membuat *web service client*. Namun untuk memudahkan pembuatan *web service server* beserta WSDL-nya maka kita bisa memanfaatkan *library* yang telah ada. Salah satu *library* yang bisa digunakan adalah NuSOAP. [19]

2.12 JSON (*JavaScript Object Notation*)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. [9]

JSON terbuat dari dua struktur yaitu pertama kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*. Dan kedua daftar nilai

terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini. [9]

Pada penelitian ini peneliti menggunakan metode pertukaran data format JSON karena strukturnya mudah dipahami bagi penulis sehingga mempercepat dalam membangun *web service* untuk aplikasi *Android*.

2.13 MySQL

MySQL merupakan perangkat lunak basis data *server* atau *smart*. Menurut *server*, karena perangkat lunak ini diletakkan di *server*, sedangkan menurut *smart*, karena dapat secara otomatis menentukan index terbaik, membagi hak akses *database* yang baik, dan mengolah transaksi dengan baik, seperti *commit*, *rollback*, dan lainnya. MySQL juga termasuk *software database* yang menggunakan bahasa standar SQL. [10]

MySQL adalah sistem manajemen *database* relasi yang bersifat ‘terbuka’. Terbuka maksudnya adalah MySQL boleh diunduh oleh siapa saja, baik versi kode program aslinya (*source code program)* maupun versi binernya (*executable program*) dan bisa digunakan secara (*relative*) gratis baik untuk dimodifikasi sesuai dengan kebutuhan seseorang maupun sebagai suatu program aplikasi komputer.

Sejarah MySQL yang merupakan hasil buah pikiran dari Michael ‘Monty’ Widenius, David Axmark, dan Allan Larson dimulai tahun 1995. Mereka bertiga kemudian mendirikan perusahaan bernama MySQL AB di Swedia. Tujuan awal ditulisnya program MySQL adalah untuk mengembangkan aplikasi web yang akan digunakan oleh salah satu klien MySQL AB. Pada saat itu MySQL AB adalahsebuah perusahaan konsultan *database* dan pengembang *software* (masih menggunakan nama perusahaan TcX DataKonsult AB). [21]

MySQL memiliki kinerja, kecepatan proses, dan ketangguhan yang tidak kalah dibanding database-database besar lainnya yang komersil seperti ORACLE, Sybase, Unify, dan sebagainya. Namun didalam penelitian ini peneliti menggunakan MySQL karena DBMS tersebut paling sering digunakan sehingga mudah dalam melakukan perbaikan saat terjadi kesalahan. [22]

2.14 Bahasa SQL (*Structure Query Language*)

SQL (*Structured Query Language*) pada dasarnya adalah bahasa komputer standar yang ditetapkan untuk mengakses dan memanipulasi sistem *database*. Sebuah *database* berisi satu tabel atau lebih dan memiliki nama yang berbeda untuk setiap tabel. Setiap tabel memiliki satu kolom (*field*) atau lebih dan memiliki baris (*record*). Query digunakan untuk mengakses dan mengolah *database*. [3] SQL terdiri dari empat bagian utama yaitu *Data Definition Language* (DDL), *Data Manipulation Language* (DML), *Data Control Language* (DCL), *Data Transaction Language* (DTL). Berikut adalah penjelasan dari setiap bagian tersebut :

1. *Data Definition Language* (DDL)

DDL merupakan perintah SQL yang digunakan untuk mendefinisikan, menciptakan, dan menghapus sebuah database ataupun tabel. Yang termasuk dalam perintah DDL meliputi *CREATE*, *DROP*, dan *ALTER*. Contoh dari perintah DDL adalah:

1. CREATE TABLE (digunakan untuk mendefinisikan sebuah tabel baru)
2. CREATE DATABASE (digunakan untuk mendefinisikan sebuah basis data yang baru)
3. CREATE INDEX (digunakan untuk membuat index)
4. CREATE VIEW (digunakan untuk membuat view)
5. DROP DATABASE (digunakan untuk menghapus basis data)
6. DROP TABLE (digunakan untuk menghapus tabel)
7. DROP INDEX (digunakan untuk menghapus index)
8. DROP VIEW (digunakan untuk menghapus view)
9. ALTER TABLE (digunakan untuk mengubah struktur suatu tabel)

2. *Data Manipulation Language* (DML)

DML merupakan perintah SQL yang digunakan untuk menambahkan, menyunting, menghapus, dan mengambil nilai dari data aktual yang didefinisikan oleh DDL. Contoh dari perintah DML adalah:

1. INSERT (digunakan untuk memasukkan nilai baru ke dalam tabel)
2. UPDATE (digunakan untuk mengubah suatu nilai pada satu atau lebih kolom)
3. DELETE (digunakan untuk menghapus suatu nilai pada tabel)
4. SELECT (digunakan untuk mengambil data dari tabel)

3. *Data Control Language* (DCL)

DCL digunakan untuk memberikan atau mencabut hak akses sumber daya tertentu didalam basis data. Pemberian hak akses dapat diterapkan pada DDL maupun DML. Contoh dari perintah TCL adalah:

1. GRANT (digunakan untuk memberikan izin)
2. REVOKE (digunakan untuk menghapus izin yang sudah ada)

4. *Data Transaction Language* (DTL)

DTL digunakan untuk mengontrol perintah dari transaksi DML hingga DDL. Contoh dari perintah DTL adalah:

1. BEGIN (digunakan untuk memulai transaksi multistatement)
2. COMMIT (digunakan untuk mengakhiri dan menerima transaksi)
3. ROLLBACK (digunakan untuk membatalkan perubahan yang dilakukan dalam *transaction* atau menghapus semua data yang dimodifikasi pada awal *transaction*)

2.15 Analisis dan Perancangan Berorientasi Objek

Analisis dan perancangan berorientasi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang didominasi oleh objek. Didalam membangun sistem berorientasi objek akan menjadi lebih baik apabila langkah awalnya didahului dengan proses analisis dan perancangan yang berorientasi objek. Tujuannya adalah mempermudah programmer didalam mendesain program dalam bentuk objek-objek beserta keterhubungan antar objek untuk kemudian dimodelkan dalam sistem nyata.

Unified Modelling Language (UML) adalah bahasa pemodelan standar untuk pengembangan perangkat lunak dan sistem. [23] Bahasa UML dapat digunakan untuk memodelkan sistem yang menggunakan pendekatan berorientasi objek. Bahasa UML memberikan banyak fitur opsional dalam mempresentasikan semua aspek penting dalam sebuah sistem seperti menggambarkan domain masalah, desain perangkat lunak yang diinginkan, atau bahkan menggambarkanimplementasi perangkat lunak yang telah selesai dibangun. Standar yang saat ini digunakan adalah UML 2.0. Didalam UML 2.0 terdapat tiga belas diagram berbeda yang dapat digunaan memodelkan perangkat lunak, diantaranya yaitu *use case diagram, activity diagram, sequence diagram, class diagram, component diagram, dan deployment diagram.*

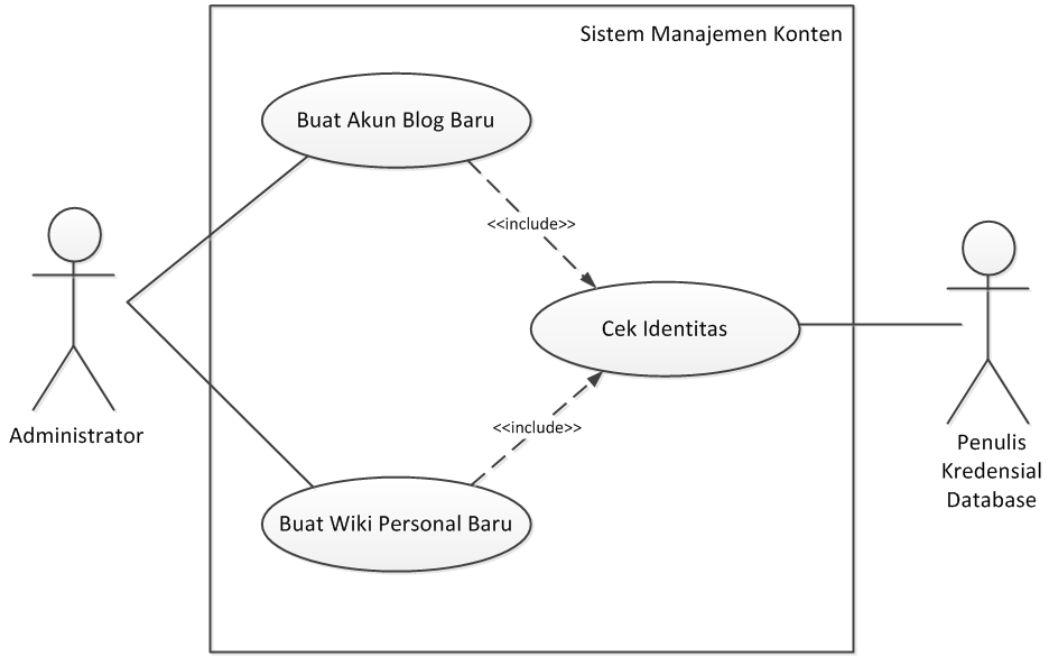
Pemodelan menggunakan UML digunakan peneliti karena lebih baik dalam menggambarkan sistem dari aplikasi remote monitoring smartphone *Android* menggunakan layanan *Short Message Service* (SMS) dimana setiap stimulus akan dapat menggambarkan sistem yang akan dibangun.

2.15.1 *Use Case Diagram*

Use case adalah situasi/*case* dimana sebuah sistem digunakan untuk memenuhi satu atau lebih kebutuhan pengguna. *Use case* dapat digunakan untuk menggambarkan fungsionalitas pada bagian-bagian yang disediakan oleh sistem. *Use case* menspesifikasikan apa yang harus dilakukan oleh sistem tetapi tidak menspesifikasikan apa yang seharusnya sistem tidak lakukan. [23]

Pada Gambar 2.3 adalah contoh diagram *use case*. Kotak besar adalah *system-boundary*. Segala hal yang berada didalam kotak tersebut adalah bagian dari sebuah pengembangan sistem. Dan yang berada diluar kotak dapat dilihat aktor yang bertindak terhadap sistem. Aktor adalah entitas diluar sistem yang memberikan stimulus terhadap sistem. Aktor umunya adalah pengguna. Kemudian didalam system-boundary dapat dilihat terdapat beberapa *use case*. *Use case* berbentuk oval dengan nama kata kerja didalamnya. Para aktor dihubungkan oleh communication line kepada *use case* yang berhubungan. Penggunaan garis

panah perlu dihindari pada *communication line* karena tidak seorang pun mengetahui apa maksud dari kepala panah tersebut.



Sumber Gambar : Learning UML 2.0 [23]

Gambar 2.3 Contoh Diagram *Use case*

Namun, dalam *use case* diagram tersebut belum menjelaskan mendetail mengenai *use case* yang tertera. Untuk menjelaskan *use case* tersebut memerlukan skenario *use case*. Skenario *use case* mendeskripsikan secara detail tentang informasi dari suatu *use case* yang terdapat pada use case diagram. Berikut adalah tipe–tipe informasi yang digunakan dalam *use case scenario* dijelaskan pada Tabel 2.2 [23]:

Tabel 2.1 Tipe Informasi pada Use Case Scenario

Use case name	Memberikan penjelasan singkat tentang nama dari use case.
Related Requirements	Daftar use case yang berhubungan dengan use case tersebut.
Goal In Context	Menjelaskan apa yang aktor coba untuk dapatkan dari use case.
Preconditions	Kondisi sistem sebelum use case dijalankan.
Successful End Condition	Kondisi sistem jika use case berhasil dijalankan.
Failed End Condition	Kondisi sistem jika use case gagal dijalankan.
Primary Actors	Aktor utama yang berpartisipasi pada use case.
Secondary Actors	Aktor yang berpartisipasi pada use case tetapi tidak menjadi yang utama.
Trigger	Event yang dipicu oleh aktor yang menyebabkan use case dijalankan.
Main Flow	Mendeskripsikan langkah yang penting pada eksekusi normal

	sebuah use case.
Extensions	Mendeskripsikan langkah alternatif suatu langkah di Main Flow.

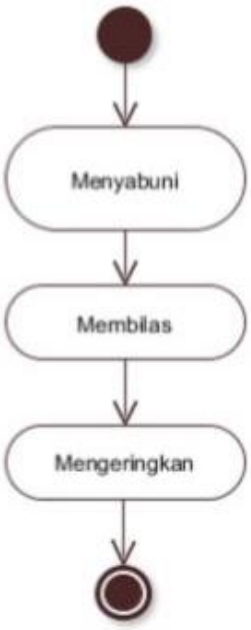
Sumber Tabel: Learning UML 2.0 [23]

2.15.2 *Activity Diagram*

Apabila *use case* menunjukkan apa yang sistem harus lakukan, berbeda halnya dengan *activity diagram* yang lebih membahas tentang bagaimana sistem akan mencapai tujuannya. *Activity diagram* menjelaskan aksi rantai pada tingkat yang lebih tinggi untuk mewakili suatu proses yang terjadi didalam sebuah sistem. [23]

Komponen utama dari suatu *activity diagram* adalah *action node*, diwakili oleh persegi panjang bulat, yang sesuai dengan tugas yang dilakukan oleh sistem perangkat lunak. Panah dari satu *action node* ke *action node* yang lain menunjukkan aliran kontrol. Panah antara dua action node berarti bahwa setelah aksi pertama selesai, aksi kedua dimulai. Sebuah titik hitam yang solid membentuk simpul awal yang menunjukkan titik awal kegiatan. Sebuah titik hitam yang dikelilingi oleh lingkaran hitam merupakan simpul akhir menunjukkan akhir kegiatan.

Fork merupakan pemisah kegiatan menjadi dua atau lebih aktivitas bersamaan. Hal ini digambarkan sebagai bar hitam horisontal dengan satu panah menunjuk ke sana dan dua atau lebih anak panah keluar. Setiap panah keluar mewakili aliran kontrol yang dapat dieksekusi bersamaan dengan arus yang sesuai dengan panah keluar lainnya. Kegiatan bersamaan dapat dilakukan pada komputer dengan menggunakan thread yang berbeda atau bahkan menggunakan komputer yang berbeda. Sebagai contoh, *activity diagram* dapat dilihat pada Gambar 2.4 berikut.



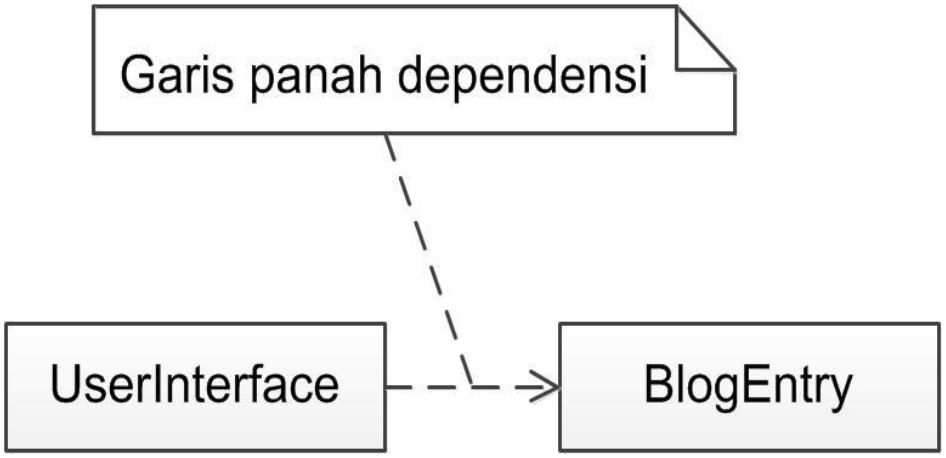
Sumber Gambar : Learning UML 2.0 [23]

Gambar 2.4 Contoh *Activity Diagram* Menyuci Mobil

2.15.3 *Class Diagram*

Untuk memodelkan kelas, termasuk atribut mereka, operasi, dan hubungan dan asosiasi mereka dengan kelas-kelas lain, UML menyediakan *class diagram*. Sebuah *class diagram* memberikan pandangan statis atau struktural dari sebuah sistem. *Class diagram* tidak menunjukkan sifat dinamis dari komunikasi antara objek dari kelas dalam diagram.

Sebuah kelas digambarkan dengan sebuah kotak yang dibagi menjadi tiga bagian. Pada bagian teratas berisi nama kelas, pada bagian tengah berisi atribut, dan pada bagian paling bawah berisi operasi-operasi yang mewakili karakteristik sebuah kelas. Sebuah atribut mengacu pada informasi yang mewakili status objek. Atribut biasanya diimplementasikan sebagai field dari sebuah kelas. Sedangkan operasi mengacu pada objek apa yang kelas dapat lakukan, biasanya diimplementasikan sebagai method dari sebuah kelas. [23] Sebagai contoh, *class diagram* dapat dilihat pada Gambar 2.5 berikut.



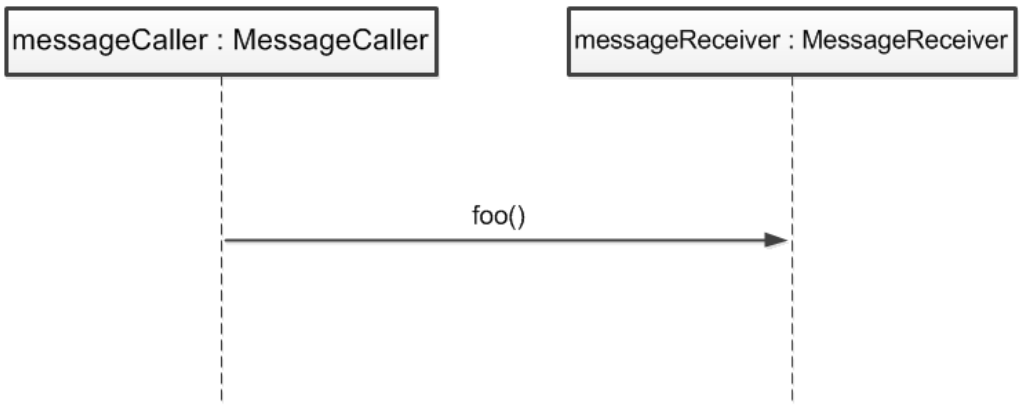
Sumber Gambar : Learning UML 2.0 [23]

Gambar 2.5 Contoh *Class Diagram*

Setiap atribut dapat memiliki nama, jenis, dan tingkat visibilitas. Jenis dan visibilitas adalah opsional. Jenis ini mengikuti nama dan dipisahkan dari nama dengan titik dua. Visibilitas ditunjukkan dengan sebelumnya -, #, ~, atau +, menunjukkan *private*, *protected*, *package*, atau *public*. [23]

2.15.4 *Sequence Diagram*

Sequence diagram berisi tentang urutan interaksi antara bagian-bagian didalam sebuah sistem. Menggunakan *sequence diagram* dapat digambarkan interaksi mana yang akan diberi *trigger* ketika sebagian *use case* dieksekusi dan dalam urutan seperti apa interaksi akan terjadi. *Sequence diagram* menunjukan banyak informasi lain tentang sebuah interaksi, tetapi intinya adalah menunjukan langkah sederhana dan efektif bagaimana mereka mengomunikasikan urutan-urutan kejadian (*events*) didalam sebuah interaksi. [23] Sebagai contoh, *sequence diagram* dapat dilihat pada Gambar 2.6 berikut.



Sumber Gambar : Learning UML 2.0 [23]

Gambar 2.6 Contoh *Sequence Diagram*

Setiap kotak pada bagian atas diagram dulunya merupakan sebuah objek. Setiap objek adalah instansi dari sebuah kelas, dan untuk menandai objek biasanya diberi garis bawah. Didalam UML 2.0, nama kotak ini disebut partisipan untuk mendeskripsikan bagian-bagian yang terlibat dialam interaksi pada sebuah sequence diagram. Di bawah setiap kotak ada garis putus-putus yang disebut *lifeline* objek. Sumbu vertikal dalam sequence diagram sesuai dengan waktu dimana waktu meningkat saat bergerak ke bawah.

Sequence diagram menunjukkan pemanggilan metode menggunakan panah horisontal, diberi label dengan nama metode dan secara opsional termasuk parameter, jenis, dan jenis kembali. Ketika sebuah objek mengeksekusi metode maka ditampilkan bar putih, disebut bar aktivasi, di bawah *lifeline* objek. [23]