BAB 2
LANDASAN TEORI

2.1 Aplikasi

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* [18].

Bahasa pemrograman berfungsi untuk menulis program dengan bahasa yang lebih mudah, dan selanjutnya akan diterjemahkan ke dalam bahasa mesin supaya bisa dimengerti oleh komputer. Bila hendak mengembangkan suatu program aplikasi untuk memecahkan permasalahan yang besar dan rumit, maka supaya program aplikasi tersebut dapat berhasil dengan baik, maka dibutuhkan prosedur dan perencanaan yang baik dalam mengembangkannya. Saat ini terdapat program-program aplikasi yang tersedia dalam bentuk paket-paket program. Beberapa perusahaan perangkat lunak telah memproduksi paket-paket perangkat lunak yang mempunyai reputasi internasional. Program-program paket tersebut dapat diandalkan, dapat memenuhi kebutuhan pemakai, dirancang dengan baik, relatif bebas dari kesalahan-kesalahan, *user friendly* (mudah digunakan), mempunyai dokumentasi manual yang memadai, mampu dikembangkan untuk kebutuhan mendatang, dan didukung perkembangannya [18].

Dari pengertian diatas peneliti menyimpulkan bahwa sebuah aplikasi dapat dibuat dan dikembangkan dengan tujuan untuk melakukan tugas yang bersifat umum atau juga dapat dikembangkan untuk melakukan tugas yang bersifat spesifik/khusus. Hal ini didasarkan pada permasalahannya yang bisa saja ditemui bersifat unik sehingga tidak ada paket-paket program yang sesuai untuk digunakan. Oleh karena itu dengan terpaksa diperlukan pengembangan program aplikasi itu sendiri.

2.2 *Smartphone*

Menurut Williams dan Sawyer [19] *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.

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*.

2.3 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* [1]. 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 [20].

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

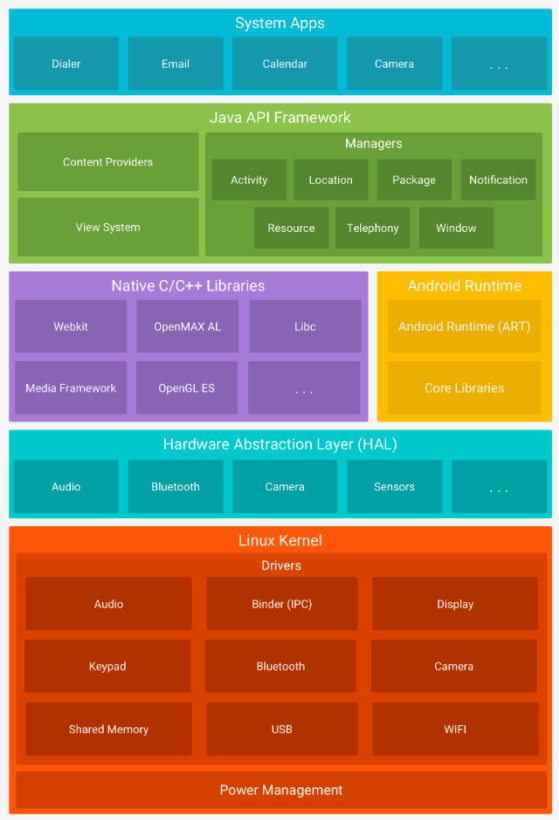
1. Android versi 1.1

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.

2.4 Arsitektur Android

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



Sumber Gambar : https://developer.android.com/guide/platform/index.html

Gambar 2.1 Arsitektur Android

1. *Linux 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 [20].

2. *Hardware Abstraction Layer (HAL)*

Hardware Abstraction Layer menyediakan *interface* standard 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 [21].

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 [21].

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 [21].

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 [21]. 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 sumberdaya *non-code* 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 fihak 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 [21].

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.

2.5 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* [20].

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 *activity* 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 antarmuka bagi pengguna aplikasi, *service* digunakan untuk proses yang bergerak tanpa antarmuka, *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.

2.6 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 [20].

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, RubyMotion, 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.

2.7 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:

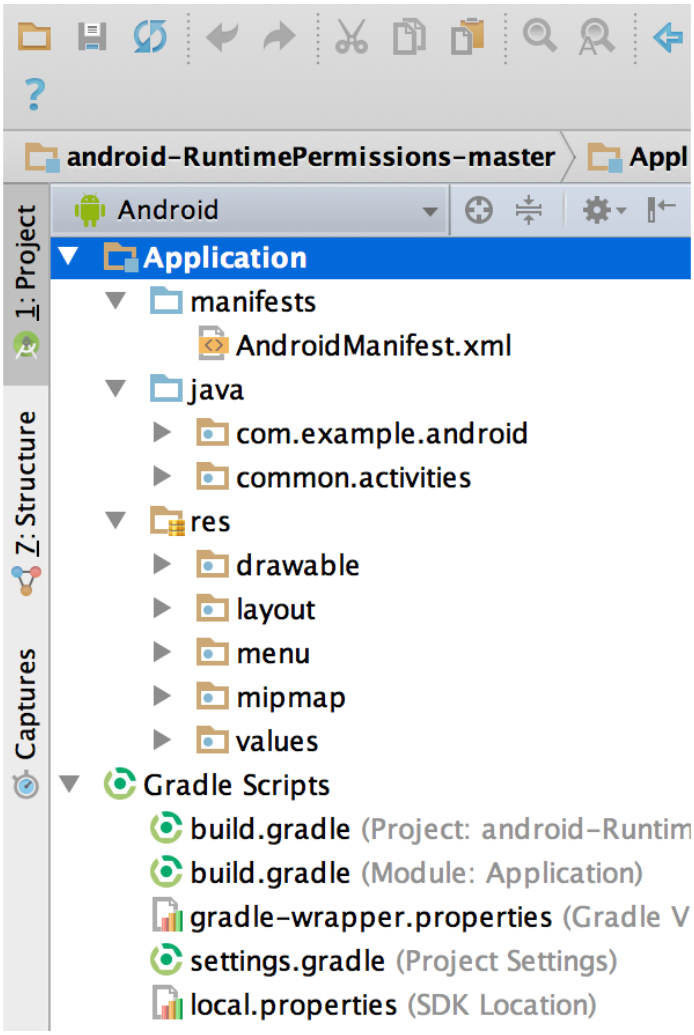
1. Sistem pembuatan berbasis *Gradle* yang fleksibel
2. *Emulator* yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru

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) [22].

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.

2.8 Bahasa Pemrograman Java

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 [23].

Menurut Budi Raharjo, Imam Heryanto, Arif haryono [24] 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. 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 platformsistem 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 secara

luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis web.

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.

2.9 Web Service

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*. [25]

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.

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 [17]. Pada pebangunan 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.

2.10 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 [26].

JSON terbuat dari dua struktur yaitu:

1. 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*.
2. 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.

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.

2.11 MySQL

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 adalah

sebuah perusahaan konsultan *database* dan pengembang *software* (masih menggunakan nama perusahaan TcX DataKonsult AB) [27].

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.

2.12 Bahasa SQL (*Structured 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 [28]. 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*)

2.13 *Short Message Service* (SMS)

Short Message Service (SMS) merupakan sebuah layanan yang banyak diaplikasikan pada sistem komunikasi tanpa kabel, memungkinkan dilakukannya pengiriman pesan dalam bentuk *alphanumeric* antar terminal pelanggan atau antara terminal pelanggan dengan sistem eksternal seperti *email, paging, voice mail,* dan lain-lain. Isu SMS pertama kali muncul di belahan Eropa pada sekitar tahun 1991

bersama sebuah teknologi komunikasi *wireless* yang saat ini cukup banyak penggunanya, yaitu *Global System for Mobile Communication*. Dipercaya bahwa *message* pertama yang dikirimkan menggunakan SMS dilakukan pada bulan Desember 1992, dikirimkan dari sebuah *Personal Computer*(PC) ke telepon *mobile* (bergerak) dalam jaringan GSM milik Vodafone Inggris. Perkembangannya kemudian merambah ke benua Amerika, dipelopori oleh beberapa operator komunikasi bergerak berbasis digital seperti BellSouth Mobility, PrimeCo, Nextel, dan beberapa operator lain. Teknologi digital yang digunakan bervariasi dari yang berbasis GSM, Time Division Multiple Access (TDMA), hingga *Code Division Multiple Access* (CDMA).

Dalam sistem SMS, mekanisme utama yang dilakukan dalam sistem adalah melakukan pengiriman *short message* dari satu terminal pelanggan ke terminal lain. Hal ini dapat dilakukan berkat adanya sebuah entitas dalam sistem SMS yang bernama *Short Message Service Center* (SMSC). SMSC merupakan sebuah perangkat yang melakukan tugas *store* and *forward* trafik *short message*. Didalamnya termasuk penentuan atau pencarian rute tujuan akhir dari *short message*. Sebuah SMS biasanya didesain untuk dapat menangani *short message* dari berbagai sumber seperti *Voice Mail System* (VMS). *Web-based messaging, Email Integration, External Short Messaging Entities* (ESME), dan lain-lain. Dalam interkoneksi dengan entitas dalam jaringan komunikasi wireless seperti *Home Location Register* (HLR) dan *Mobile Switching Center* (MSC), SMSC biasanya selalu menggunakan *Signal Transfer Point* (STP). [29]

Didalam penelitian ini media SMS digunakan peneliti untuk melengkapi kelemahan dari penggunaan internet. Dengan menyampaikan pesan komunikasi menggunakan SMS reliabilitas dapat ditingkatkan apabila dibandingkan dengan internet yang dapat saja terganggu karena sinyal lemah atau pun paket data tidak tersedia.

2.14 Layanan *Short Message Service* (SMS)

Layanan SMS merupakan sebuah layanan yang bersifat non *real-time* dimana sebuah *short message* dapat di-*submit* ke suatu tujuan, tidak peduli apakah

tujuan tersebut aktif atau tidak. Bila dideteksi bahwa tujuan tidak aktif, maka sistem akan menunda pengiriman ke tujuan hingga tujuan aktif kembali. Pada dasarnya sistem SMS akan menjamin *delivery* dari suatu *short message* hingga sampai ke tujuan. Kegagalan pengiriman yang bersifat sementara seperti tujuan tidak aktif akan selalu teridentifikasi sehingga pengiriman ulang *short message* akan selalu dilakukan kecuali diberlakukan aturan bahwa *short message* yang telah melampaui batas waktu tertentu harus dihapus dan dinyatakan gagal terkirim [29].

Karakteristik utama SMS adalah SMS merupakan sebuah sistem pengiriman data dalam paket yang bersifat *out-of-band* dengan *bandwidth* kecil. Dengan karakteristik ini, pengiriman suatu *burst* data yang pendek dapat dilakukan dengan efisiensi yang sangat tinggi. Keuntungan pelanggan dengan adanya SMS adalah pada masalah kenyamanan, fleksibilitas, dan kemudahan integrasi antara *messaging service* dengan data *access*. Dari perspektif ini, manfaat utama yang didapat pelanggan adalah perangkat *handset* yang dimiliki oleh pelanggan seolah-olah merupakan ekstensi dari perangkat komputer. SMS juga mengeliminasi kebutuhan akan perangkat tambahan untuk *messaging service* karena layanan ini dapat dinikmati dengan menggunakan perangkat yang sama, yaitu perangkat komunikasi *wireless*. Akan tetapi, keuntungan bagi pelanggan masih tergantung pada jenis layanan yang diberikan oleh operator, namun paling tidak mereka mendapatkan keuntungan berikut ini:

1. Pengiriman notifikasi dan alert.
2. Jaminan pengiriman *short message*.
3. Pengiriman informasi pendek yang reliable(andal) dan murah.
4. Kemampuan untuk mem-filter pesan dan membalas dengan cara yang selektif.
5. Meningkatkan produktivitas.

Fungsi-fungsi dalam SMS yang lebih kompleks akan memberi keuntungan bagi pelanggan, seperti:

1. Pengiriman short message pada lebih dari satu tujuan dalam satu waktu
2. Kemampuan untuk menerima berbagai informasi yang berbeda.
3. Meng-*generate* email

4. Integrasi dengan aplikasi lain yang berbasis internet dan data.

Berdasarkan penjelasan diatas layanan SMS akan dapat mendukung penelitian ini karena layanan tersebut bersifat non *real-time* dimana pesan yang dikirim ke suatu tujuan tidak memperdulikan apakah tujuan tersebut aktif atau tidak. Selain itu sistem SMS akan menjamin *delivery* dari suatu *short message* hingga sampai ke tujuan. Kegagalan pengiriman yang bersifat sementara seperti tujuan tidak aktif akan selalu teridentifikasi sehingga pengiriman ulang akan selalu dilakukan kecuali diberlakukan aturan bahwa short message yang telah melampaui batas waktu tertentu harus dihapus dan dinyatakan gagal terkirim.

2.15 *Remote Monitoring*

Remote dalam bahasa Indonesia memiliki arti jarak jauh sedangkan *monitoring* didefinisikan sebagai siklus kegiatan yang mencakup pengumpulan, peninjauan ulang, pelaporan, dan tindakan atas informasi suatu proses yang sedang diimplementasikan [30]. *Monitoring* ditinjau dari hubungan terhadap manajemen kinerja adalah proses terintegrasi untuk memastikan bahwa proses berjalan sesuai rencana (*on the track*). *Monitoring* dapat memberikan informasi keberlangsungan proses untuk menetapkan langkah menuju ke arah perbaikan yang berkesinambungan. Umumnya, pelaku *monitoring* merupakan pihak-pihak yang berkepentingan dalam proses, baik pelaku proses (*self monitoring*) maupun atasan/supervisor pekerja [31].

Didalam penelitian ini terdapat kegiatan pelaporan mengenai informasi *smartphone* yang dilakukan secara jarak jauh menggunakan media SMS. Oleh karena itu *remote monitoring* dalam penelitian ini memiliki arti aktivitas pada *smartphone* yang diinginkan oleh pengguna aplikasi yang dapat dilakukan dari jarak jauh.

2.16 QRCode

Quick Response Code sering di sebut QR Code atau Kode QR adalah semacam simbol dua dimensi yang dikembangkan oleh Denso Wave yang merupakan anak perusahaan dari Toyota sebuah perusahaan Jepang pada tahun

1994. Tujuan dari QR Code ini adalah untuk menyampaikan informasi secara cepat dan juga mendapat tanggapan secara cepat. Pada awalnya QR Code digunakan untuk pelacakan bagian kendaraan untuk manufacturing. Namun sekarang, telah digunakan untuk komersil yang ditujukan pada pengguna telepon seluler. QR Code adalah perkembangan dari barcode atau kode batang yang hanya mampu menyimpan informasi secara horizontal sedangkan QR Code mampu menyimpan informasi lebih banyak, baik secara horizontal maupun vertikal.

QR Code biasanya berbentuk persegi putih kecil dengan bentuk geometris hitam, meskipun sekarang banyak yang telah berwarna dan digunakan sebagai brand produk. Informasi yang dikodekan dalam QR Code dapat berupa URL, nomor telepon, pesan SMS, V-Card, atau teks apapun [32]. QR Code telah mendapatkan standarisasi internasional ISO/IEC18004 dan Jepang JIS-X-0510. Beberapa manfaat yang terdapat pada QR Code antara lain [33]:

1. Kapasitas tinggi dalam menyimpan data. Sebuah QR Code tunggal dapat menyimpan sampai 7.089 angka.
2. Ukuran yang kecil. Sebuah QR Code dapat menyimpan jumlah data yang sama dengan barcode 1D dan tidak memerlukan ruang besar.
3. Dapat mengoreksi kesalahan. Tergantung pada tingkat koreksi kesalahan yang dipilih, data pada QR code yang kotor atau rusak sampai 30% dapat diterjemahkan dengan baik.
4. Banyak jenis data. QR Code dapat menangani angka, abjad, simbol, karakter bahasa Jepang, Cina atau Korea dan data biner.
5. Kompensasi distorsi. QR Code tetap dapat dibaca pada permukaan melengkung atau terdistorsi.
6. Kemampuan menghubungkan. Sebuah QR Code dapat dibagi hingga 16 simbol yang lebih kecil agar sesuai dengan ruang. Simbol-simbol kecil yang dibaca sebagai kode tunggal apabila di scan menurut urutan.

Fitur pindai gambar QRCode akan digunakan pada penelitian ini untuk membaca instruksi *monitoring* yang tersemat dalam bentuk gambar QRCode untuk kemudian dikirimkan kepada *smartphone* target. Fitur ini dapat digunakan apabila

pengguna aplikasi tidak akan langsung mengirimkan SMS instruksi *monitoring* setelah meng-*generate* instruksi tersebut melalui antarmuka aplikasi.

2.17 Kriptografi

Kriptografi adalah suatu ilmu atau seni mengamankan pesan yang dilakukan oleh kriptografer. Pesan yang tidak disandikan disebut sebagai *plaintext.* Proses yang dilakukan untuk mengubah *plaintext* kedalam *ciphertext* disebut enkripsi, sedangkan proses untuk mengubah *ciphertext* kembali ke *plaintext* disebut dekripsi. Didalam proses enkripsi maupun dekripsi diatur oleh satu atau lebih kunci kriptografi. Kunci-kunci yang digunakan tidak perlu identik, tetapi dapat disesuaikan dengan sistem yang digunakan.

Secara umum proses enkripsi dan dekripsi dapat diterangkan secara matematis sebagai berikut [34]:

EK(M)=C (Proses Enkripsi)

DK(C)=M (Proses Dekripsi)

Pada saat proses enkripsi pesan M disandikan dengan suatu kunci K lalu menghasilkan pesan C, sedangkan pada proses dekripsi pesan C diuraikan dengan menggunakan kunci K sehingga dihasilkan pesan M yang sama seperti pesan sebelumnya.

Keamanan suatu pesan tergantung pada kunci atau kunci-kunci yang digunakan dan tidak bergantung pada algoritma yang digunakan. Sehingga algoritma-algoritma yang digunakan tersebut dapat dipublikasikan dan dianalisis, serta produk-produk yang menggunakan algoritma tersebut dapat diproduksi massal. Tidak menjadi masalah apabila seseorang mengetahui algoritma yang digunakan. Selama ia tidak mengetahui kunci yang dipakai, ia tetap tidak dapat membaca pesan tersebut.

Didalam penelitian ini akan menerapkan kriptografi kunci asimetris untuk mengamankan SMS instruksi *monitoring* yang akan dikirimkan ke target *smartphone.* Kriptografi ini akan memiliki satu pasang kunci dimana kunci publik digunakan untuk melakukan enkripsi dan kunci privat akan digunakan untuk melakukan dekripsi pesan. Kunci publik akan disimpan di server sedangkan kunci

privat akan disimpan pada *smartphone* sehingga tidak ada informasi keluar yang memberi petunjuk pengguna untuk melakukan dekripsi pesan. Hal ini dilakukan untuk menghindari penyalahgunaan kode instruksi yang dapat menganggu keamanan *smartphone* pengguna.

2.18 Metode Kriptografi RSA (Rivest, Shamir, Adleman)

Di tahun 1978, Rivest, Shamir, dan Adleman membuat sebuah algoritma untuk teori penomoran pada sebuah *public key cryptosystem*. Algoritma ini dikenal dengan RSA *cryptosystem*. RSA adalah salah satu dari *public key cryptosystem* yang sangat sering dipakai. Enkripsi dan dekripsi data model ini terletak pada kesulitan untuk memfaktorkan modulus *n* yang sangat besar. Besarnya bilangan yang digunakan mengakibatkan lambatnya digunakan untuk memberikan privasi terhadap keaslian suatu data digital. Keamanan operasi yang melibatkan algoritma RSA ini lebih lambat saat implementasi terhadap *hardware* jika dibandingkan dengan algorimta *private key data encryption standard* (DES).

Secara umum algoritma metode RSA ini memiliki tiga prosedur yaitu penciptaan kunci, enkripsi, dan dekripsi. Adapun penjelasan dari masing-masing prosedur yaitu sebagai berikut :

1. *Key Generation*

Pada tahapan ini merupakan pembangkitan kunci publik dan kunci privat. Kunci publik akan digunakan untuk proses enkripsi, sedangkan kunci privat akan digunakan untuk proses dekripsi. Adapun tahapan dari *key generation* adalah sebagai berikut :

1. Dapatkan dua bilangan prima besar, *p* dan *q*.
2. Dapatkan nilai *n*, dimana *n=pq*
3. Dapatkan nilai *m*, dimana *m=(p-1)(q-1)*
4. Pilih sebuah angka kecil *e*, coprime untuk *m*
5. Cari nilai *d*, dimana *d.e % m=1*

Setelah selesai maka *publish e* dan *n* sebagai kunci publik. Lalu ambil *d* dan *n* sebagai kunci privat.

2. *Enryption*

Kunci publik akan menentukan bagaimana transformasi teks didalam proses enkripsi berlangsung. Proses enkripsi pada metode RSA dapat dinotasikan sebagai berikut :

$C=P^e \% n$

3. *Decryption*

Kunci privat akan menentukan bagaimana transformasi teks didalam proses dekripsi berlangsung. Proses dekripsi pada metode RSA dapat dinotasikan sebagai berikut :

$P = C^d \% n$

Kriptografi menggunakan metode RSA 512bit digunakan peneliti sebagai bagian dari keamanan sistem karena dapat melakukan enkripsi SMS dengan hasil keluaran *ciphertext* kurang dari 160 karakter. Dengan hal tersebut membuat pengguna cukup mengeluarkan biaya sebesar satu buah SMS untuk mengirimkan satu instruksi *monitoring*.

2.19 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 [35]. 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 menggambarkan

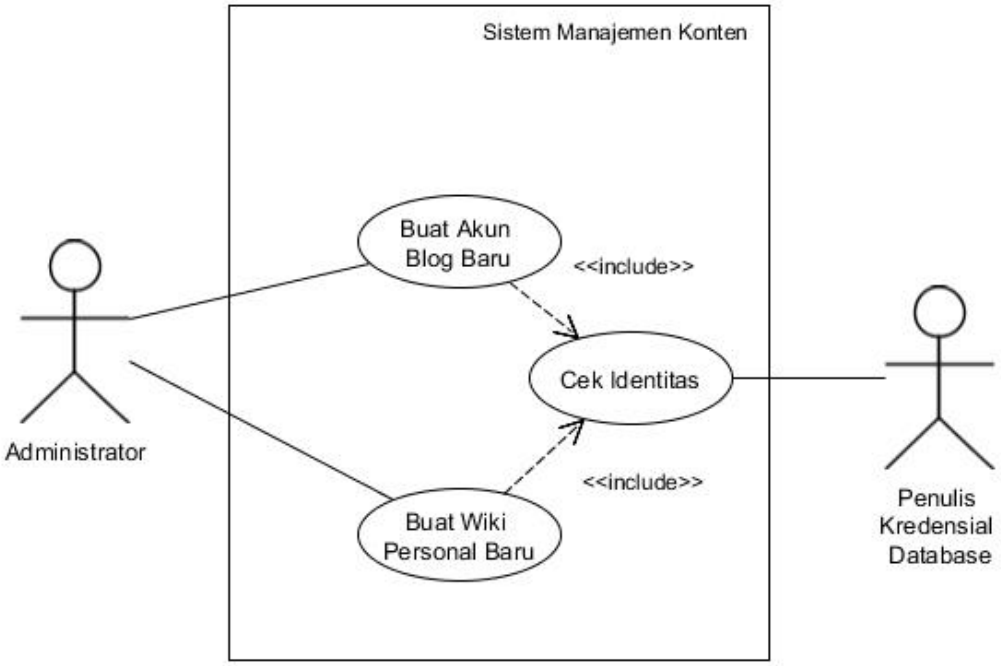
implementasi perangkat lunak yang telah selesai dibangun. Standar yang saat ini digunakan adalah UML 2.0. Didalam UML 2.0 terdapat tiga belas diagram berbeda yang dapat digunaan memodelkan perangkat lunak, diantaranya yaitu *use case diagram, activity diagram, sequence diagram, class diagram, component 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.

2.19.1 *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 [35].

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(2006) [35]

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 [35]:

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*.

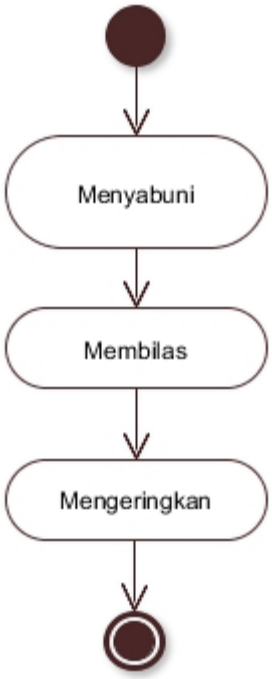
Sumber Tabel: Learning UML 2.0 (2006) [35]

2.19.2 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 [35].

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.



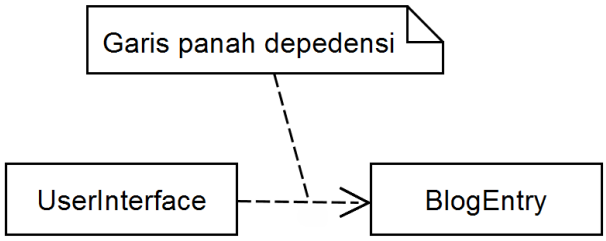
Sumber Gambar : Learning UML 2.0(2006) [35]

Gambar 2.4 Contoh *Activity Diagram* Menyuci Mobil

2.19.3 *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 [35]. Sebagai contoh, *class diagram* dapat dilihat pada Gambar 2.5 berikut.



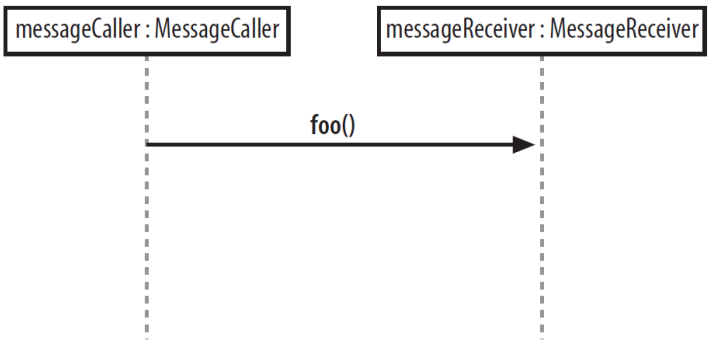
Sumber Gambar : Learning UML 2.0(2006) [35]

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* [35].

2.19.4 *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 [35]. Sebagai contoh, *sequence diagram* dapat dilihat pada Gambar 2.6 berikut.



Sumber Gambar : Learning UML 2.0(2006) [35]

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 [35].