BAB II

LANDASAN TEORI

2.1. Tinjauan Jurnal

Pada penyusunan skripsi ini penulis merujuk pada beberapa jurnal tentang *system* aplikasi android *location based system* (LBS).

Menurut Hati (2013:1) tentang aplikasi penanda lokasi peta *digital* berbasis *mobile GIS* pada *smartphone android*, adalah "Berkunjung ke tempat baru dalam kota maupun luar kota merupakan hal yang sangat wajar. Permasalahanya adalah terkadang beberapa orang susah mengingat kembali letak suatu tempat ataupun arah rute menuju tempat tersebut. Dengan bantuan *Global Positioning System (GPS)* yang berfungsi sebagai penunjuk lokasi, *Location Based Service (LBS)* yang menyediakan informasi berdasarkan letak geografis perangkat *mobile*, melalui visualisasi *Google Maps*, maka aplikasi ini akan mudah digunakan. Aplikasi ini dibangun dengan pemograman *java Android* menggunakan *software ADT Bundle*. Hasil akhir dari penelitian ini adalah berupa aplikasi penanda lokasi peta *digital* berbasis *mobile GIS* pada *smartphone android*. Aplikasi ini memiliki beberapa fitur utama seperti *input* data, menampilkan *list* data tersimpan, menampilkan rute pada peta, dan mem*backup* dan meng*import* data".

Sedangkan menurut Rokhman (2013:1) Aplikasi pencarian lokasi fasilitas umum berbasis foursquare APIv2 pada sistem operasi android adalah:

> Sebuah smartphone umumnya dilengkapi dengan *Global Positioning System* (GPS). Pengguna *smartphone* disamping dapat mengetahui lokasi dirinya, umumnya juga ingin mengetahui lokasi sekitarnya. *Foursquare* merupakan salah satu jejaring sosial yang menyediakan layanan berbasis lokasi. *Foursquare* memiliki fitur *check-in*, untuk menandai lokasi pengguna. Dalam penelitian ini akan dikembangkan aplikasi pada perangkat dengan sistem operasi *android* yang dapat mencari lokasi fasilitas umum di sekitar pengguna dengan memanfaatkan teknologi layanan berbasis lokasi. Aplikasi ini memanfaatkan data dari *Foursquare*. Hasil pengujian terhadap aplikasi yang dibangun menunjukan *filter* data dan sistem *auto check-in* berjalan dengan baik sehingga duplikasi data dalam *Foursquare* dapat diminimalkan.

2.2. Konsep Dasar Program

Menurut Rahmat C (2012:2) Program adalah kumpulan instruksi yang disusun menjadi satu kesatuan prosedur yang berurutan langkah yang disusun secara logis dan sistematis untuk

menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemrograman, sehingga dapat dieksekusi oleh suatu program.

Bahasa pemrograman, atau sering diistilahkan juga dengan bahasa komputer, adalah teknik komando/instruksi standar untuk memerintah komputer. Bahasa pemrograman ini merupakan suatu himpunan dari aturan *sintaks* dan semantik yang dipakai untuk mendefinisikan program komputer. Bahasa ini memungkinkan seorang programmer dapat menentukan secara persis data mana yang akan diolah oleh komputer, bagaimana data ini akan disimpan/diteruskan, dan jenis langkah apa secara persis yang akan diambil dalam berbagai situasi. Dalam program ada beberapa pemograman yang digunakan seperti:

2.2.1. *Object Oriented Programming* (OOP)

Menurut Raharjo Budi,dkk (2012:35) Pemograman berorientasi objek atau yang lebih lazim dikenal dengan sebutan asing *Object Oriented Programming* (OOP) adalah inti dari pemrograman java. Semua program java merupakan objek. Maka dari itu, sebelum anda memulai penulisan kode-kode program java pada bab-bab selanjutya, sebaiknya anda mengetahui terlebih dahulu dasar-dasar dari konsep yang terkandung dalam pemrograman berorientasi objek. Beberapa ciri dari pemrograman berorientasi objek adalah abtraksi

1. Objek

Bentuk baik yang nyata atau tidak, seperti manusia, hewan, benda, konsep, aliran, dan lain-lain. Objek merupakan inisiasi (turunan langsung) dari suatu kelas.

2. Kelas

Kumpulan objek yang memiliki kemiripan perilaku (*method*), ciri atau karateristik (*property*). Contoh objek orang dari kelas manusia, potongan sebagai berikut: Manusia orang1=*new* manusia(“Parto”);

3. *Method*

Perilaku dari objek atau kelas tertentu. Merupakann perwujudan aksi atau tindakan dari dunia nyata di dalam pemrograman komputer.

4. Konstruktor

Suatu fungsi yang dideklarasikan atau didefinisikan di dalam kelas, konstruktor dijalankan bersamaan dengan terciptanya kelas tersebut. Dalam suatu kelas biasa terdapat lebih dari satu konstruktor. Konstruktor seperti *method* tetapi tidak mengembalikan nilai dan dapat didefinisikan tanpa parameter atau memakainya.

5. De-konstruktor

Fungsi yang dideklarasikan dalam kelas, nama sama dengan nama fungsinya. Tetapi dijalankan bersamaan dengan dimusnahkannya kelas tersebut.

6. Karateristik / *Properties*

Ciri yang dimiliki oleh suatu objek, karatesistik ini juga sebagai pembeda objek satu dengan objek lainnya dalam kelas yang sama (konsep individu).

7. Variabel

Tempat menampung data sementara, dalam pemrograman objek biasanya disebut data, sedangkan dalam pemrograman prosedural sering disebut dengan variabel.

8. Data

Istilah lain dari variabel dalam OOP. Dalam pemrograman java bisa juga disebut *field*, data *member* atau *instance variable.*

9. Hak Akses (*Access Attribute*)

Hak akses digunakan dapat menentukan data *member* mana yang dapat digunakan oleh kelas lain, dan mana yang tidak dapat digunakan. Hak akses ini sangat penting dalam membuat program program turunan kelas.

a. Public

Data *member* atau *variable* dapat diakses dari kelas mana saja

b. Protected

Dapat mengakses data *member* dari kelas dalam *package* yang sama dan subkelasnya

c. Private

Kelas yang data *member*nya memakai *private* hanya dapat digunakan oleh kelas bersangkutan, tidak dapat digunakan kelas lain

d. Tidak Disebutkan

Data *member* dapat diakses dari kelas dalam *package* yang sama

2.2.2. Android

Android adalah sebuah *system* operasi untuk perangkat mobile berbasis *linux* yang mencakup *system* operasi, *middleware* dan aplikasi android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Safaat H (2012:1)

1. Pada perkembanganya, sistem operasi Android telah mengalami beberapa perubahan dan perbaikan. Dan yang paling menarik adalah versi keluaran Android yang diberi nama seperti nama-nama makanan. Android sendiri merupakan sistem operasi yang berbasis *Linux* yang terlihat dari kernelnya, dan dibuat khusus untuk telepon seluler pada awalnya.

Berikut ini rangkaian versi android dari yang awal sampai yang terbaru sampai saat ini banyak dipakai:

a. Versi 1.5 bernama *Cupcake* yang dirilis pada *April* 30, 2009
b. Versi 1.6 bernama *Donut* yang dirilis pada *September* 15, 2009
c. Versi 2.0-2.1 bernama *Eclair* yang dirilis pada *October* 26, 2009
d. Versi 2.2 bernama *Froyo* yang dirilis pada *May* 20, 2010
e. Versi 2.3-2.3.2 bernama *Gingerbread* yang dirilis pada *December* 6, 2010
f. Versi 2.3.3-2.3.7 bernama *Gingerbread* yang dirilis pada *February* 9, 2011
g. Versi 3.1 bernama *Honeycomb* yang dirilis pada *May* 10, 2011
h. Versi 3.2 bernama *Honeycomb* yang dirilis pada *July* 15, 2011
i. Versi 4.0-4.0.2 bernama *Ice Cream Sandwich* yang dirilis pada *December* 16, 2011
j. Versi 4.0.3-4.0.4 bernama *Ice Cream Sandwich* yang dirilis pada *March* 12, 2012
k. Versi 4.1 bernama *Jelly Bean* yang dirilis pada *July* 9, 2012
l. Versi 4.2 bernama *Jelly Bean* yang dirilis pada *November* 13, 2012
m. Versi 4.3 bernama *Jelly Bean* yang dirilis pada *July* 24, 2013
n. Versi 4.4 bernama *KitKat* yang dirilis pada *October* 31, 2013

2.2.3. Java

Java adalah program yang sudah termasuk dalam *repository* dalam setiap versi dari Ubuntu, tinggal kita hubungkan Ubuntu kita ke *repository* lalu melakukan instalasi java. Safaat H (2012:14). Seperti yang dijelaskan bahwa android adalah aplikasi yang dikembangkan dengan berbasis java, sehingga sebelum kita melakukan coding aplikasi berbasis android, *computer/pc* kita harus sudah terinstal program java.

Sejak dirilis pada tahun 1995, bahasa pemograman Java dengan cepat memperoleh popularitas di kalangan pemograman. Keberhasilan ini disebabkan teknologi baru yang diperkenalkan *Sun Microsystems* yaitu *Java Virtual Machine* (JVM), yang memungkinkan sebuah aplikasi dijalankan di atas *platform* apa saja sepanjang pada mesin tersebut dipasang JVM.

Kekurangan Dan Kelebihan Java

Java merupakan bahasa pemrograman berorientasi objek (OOP)

1. Mudah untuk dikembangkan
2. Sifatnya *multiplatform*
3. Memiliki kemudahan dalam menyusun suatu *script*
4. Apabila programmer berorientasi pada *usability*, maka Java sangat mendukung
5. Bahasa pemograman yang berorientasi terhadap objek
6. Penggunaan *memory* yang cukup tinggi

2.2.4. Android Studio

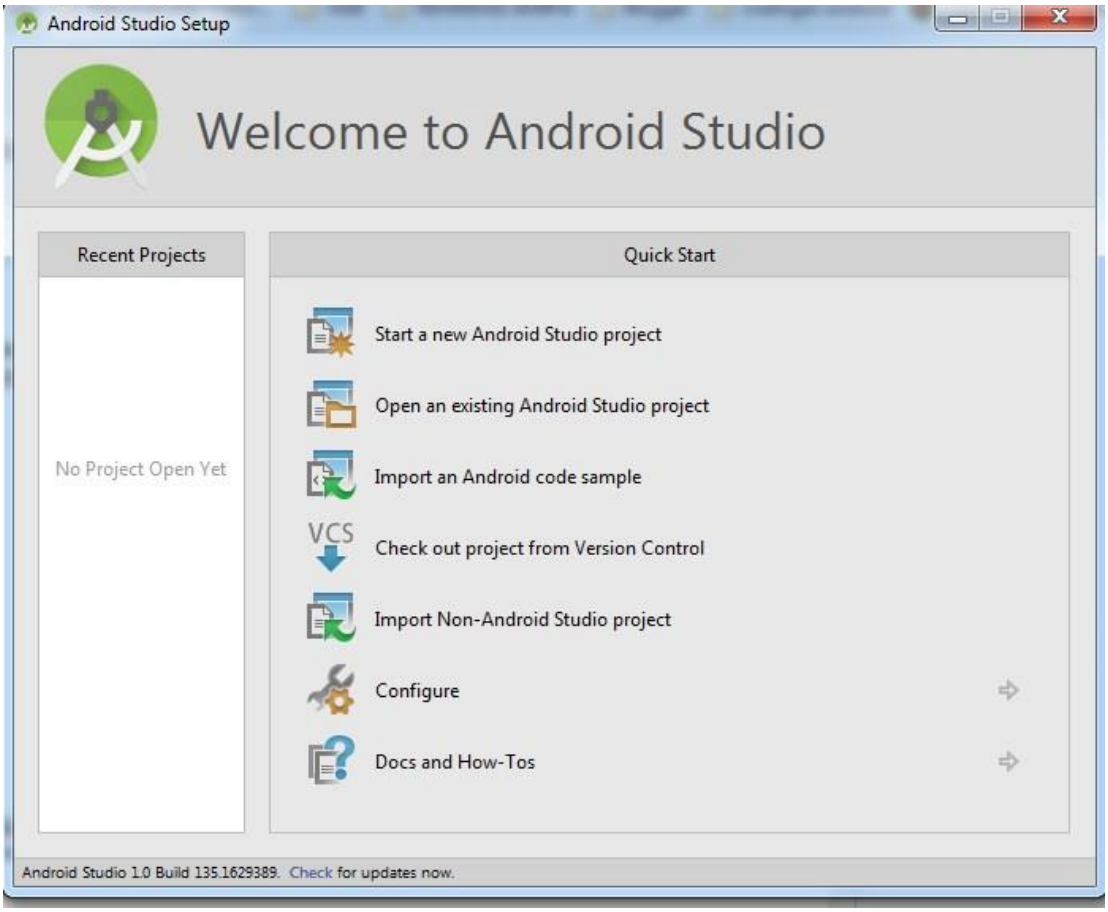
Android *Studio* merupakan sebuah *integrated Development Environment* (IDE) khusus untuk membangun aplikasi yang berjalan pada *platform* android. Android *studio* ini berbasis pada IntelliJ IDEA, sebuah IDE untuk bahasa pemograman java. Bahasa pemograman utama adalah java, sedangkan untuk membuat tampilan atau *layout*, digunakan bahasa XML. Android *Studio* juga terintegrasi dengan Android *Software Development Kit* (SDK) untuk *deploy* ke perangkat android. Jubilee enterprise (2015:10).

Android *Studio* adalah sebuah IDE untuk android *Development* yang diperkenalkan Google I/O 2013. Android *Studio* merupakan pengembangan dari *Eclipe* IDE, dan dibuat

berdasarkan IDE Java terpopuler, yaitu IntelliJ IDEA. Android *studio* merupakan IDE resmi untuk pengembangan aplikasi Android. Sebagai pengembangan dari *Eclipse*, Android *studio* mempunyai banyak fitur-fitur baru dibandingkan dengan *Eclipse* IDE. Berbeda dengan *Eclipse* yang menggunakan *Ant*, Android *studio* menggunakan *Gradle* sebagai *build environment*. Fitur-fitur lainnya adalah sebagai berikut:

1. Menggunakan *Gradle-based build system* yang fleksibel
2. Bisa mem-*build multiple* APK
3. Template *support* untuk *Google Services* dan bebagai macam tipe perangkat
4. *Layout editor* yang lebih bagus
5. *Built-in support* untuk *Google Cloud Platform*, sehingga mudah untuk integrasi dengan *Google Cloud Messaging* dan *App Engine*
6. *Import library* langsung dari *Maven repository*
7. Dan masih banyak lagi lainnya

Tampilan Awal Android Studio:



Gambar 11.1

Tampilan awal android studio

2.2.5. Android SDK

SDK adalah tools untuk para programmer dalam mengembangkan aplikasi yang berbasis pada google android. SDK mempunyai seperangkat tools pengembangan yang yang mudah dipahami dan komprehensif. Android adalah subset software lunak untuk seluler meliputi, Sistem operasi, Middleware, Aplikasi Kunci Release Oleh Google.

Perangkat Lunak Pengembangan Kit (SDK atau bisa disebut *"devkit"*) adalah satu set perangkat lunak pengembangan yang memungkinkan kita menciptakan aplikasi, kerangka kerja perangkat lunak, platform perangkat keras, sistem komputer, konsol video *game*, sistem operasi, atau serupa *platform*. Sesuatu yang sederhana seperti sebuah antar muka pemograman aplikasi (API) dalam bentuk beberapa *file* ke antar muka tertentu, ke bahasa pemograman tertentu atau termasuk *hardware* canggih untuk berkomunikasi dengan sistem *embedded* tertentu. Umumnya IDE sudah termasuk alat bantu *debugging* dan *utilitas* yang sering disajikan dalam lingkungan pengembangan terpadu (IDE). SDK juga sering termasuk kode contoh dan catatan teknis pendukung atau dokumentasi pendukung lainnya untuk membantu membantu memperjelas poin dari bahan refrensi utama.

SDK memungkinkan memiliki terpasang lisensi yang membuat mereka tidak cocok untuk membangun perangkat lunak yang ditujukan untuk dikembangkan di bawah lisensi yang tidak *kompatibel*. Misalnya, SDK *proprietary* mungkin akan tidak sesuai dengan pengembangan perangkat lunak bebas, sementara GPL SDK berlisensi bisa tidak sesuai dengan pengembangan perangka lunak berpemilik. LGPL SDK biasanya aman untuk pengembangan kepemilikan. Seseorang insinyur perangkat lunak biasanya menerima SDK dari pengembang sistem target.

Seringkali SDK dapat di-*download* secara langsung melalui internet. SDK banyak disediakan secara gratis untuk mendorong pengembang untuk menggunakan sistem atau bahasa. Kadang-kadang ini digunakan untuk alat pemasaran. Sebuah SDK untuk sistem operasi *add-on* (misalnya, *quickTime* pada *Mac OS*) mungkin termasuk perangkat lunak itu sendiri akan digunakan untuk tujuan pembangunan, jika tidak tentu untuk *redistribusi*. Sebuah situasi yang menarik muncul di sini antara platform dimana memungkinkan untuk mengembangkan aplikasi yang dapat setidaknya *start up* pada konfigurasi sistem tanpa *add-on* terpasang, dan

menggunakan *Gestalt* gaya *run-time* permintaan lingkungan untuk menentukan apakah *add-on* ini hadir, dan orang-orang dimana aplikasi hanya akan gagal untuk memulai. Dengan kata lain, adalah mungkin untuk membangun sebuah biner tunggal yang akan berjalan pada konfigurasi dengan dan tanpa operasi *add-on* ini, meskipun dengan fungsionalitas yang **l**ebih rendah dalam situasi yang terakhir. Penyedia SDK untuk sistem tertentu atau subsistem kadang-kadang dapat menggantikan istilah yang lebih spesifik, bukan perangkat lunak. Misalnya, baik *Microsoft* dan *Apple* menyediakan kit pengembangan driver (DDK) untuk mengembangkan driver perangkat.

2.3. Metode Algoritma

Menurut Rachmat C (2014:4) Algoritma adalah urutan langkah yang logis untuk menyelesaikan masalah tertentu.

Pada kesempatan ini penulis menggunakan algoritma Pencarian (*searching*), dimana data akan di input oleh penguna di lapangan kemudian data akan di tampilkan di dalam menu-menu di aplikasi.

1. Pengertian Algoritma Pencarian (*searching*)

Pencarian (*searching*) merupakan proses yang *fundamental* dalam pengolahan data. Proses pencarian adalah menemukan nilai (data) tertentu di dalam sekumpulan data yang bertipe sama (baik bertipe dasar maupun bertipe bentukan). Sebuah algoritma pencarian dijelaskan secara luas adalah sebuah algoritma yang menerima masukan berupa sebuah masalah dan menghasilkan sebuah solusi untuk masalah tersebut, yang biasanya didapat dari evaluasi beberapa kemungkinan solusi. Algoritma pencarian (*searching algoritma*) adalah algoritma yang menerima sebuah argumen kunci dan dengan langkah-langkah tertentu akan mencari rekaman dengan kunci tersebut. Setelah proses pencarian dilaksanakan, akan diperoleh salah satu dari dua kemungkinan, yaitu data yang dicari ditemukan (*succesful*) atau tidak ditemukan (*unsuccesful*).

2. Macam-macam Algoritma Pencarian (*Searching*)

a.Pencarian Sekuensial (*Sequential searching*)

Pengertian Pencarian Sekuensial (*Sequential searching*) atau pencarian berurutan sering disebut pencarian linear merupakan metode pencarian yang paling sederhana. Pencarian berurutan adalah proses membandingkan setiap elemen larik satu per satu secara berurutan, mulai dari elemen pertama sampai elemen yang dicari ditemukan atau seluruh elemen sudah diperiksa.

b.Pencarian Beruntun dengan *Sentinel*

Pengertian Pencarian Bruntun dengan *Sentinel* adalah jika pencarian bertujuan untuk menambahkan elemen baru setelah elemen terakhir larik, maka terdapat sebuah varian dari metode pencarian beruntun. Nilai x yang akan dicari sengaja ditambahkan terlebih dahulu. Data yang ditambahkan setelah elemen terakhir larik ini disebut *sentinel*.

3. Algoritma Pencarian String (String Matching)

Algoritma Knuth-Morris-Pratt merupakan salah satu ***algoritma pencarian string*** (*string matcing*) yang dikembangkan secara terpisah oleh Donald E. Knuth pada tahun 1967 dan James H. Morris bersama Vaughan R. Pratt pada tahun 1966, namun keduanya mempublikasikannya secara bersamaan pada tahun 1977. Langkah-langkah yang dilakukan algoritma Knuth-Morris-Pratt pada saat mencocokkan string yaitu (modifikasi):

1. String pattern (kata yang dicari) akan dipecah menjadi array karakter
2. String text (teks, artikel, dsb) akan dipecah menjadi array karakter
3. Menentukan lompatan yang akan dilakukan ketika pencarian (funsi preKMP())

4. Algoritma Knuth-Morris-Pratt mulai mencocokkan pattern pada awal teks. ()
5. Dari kiri ke kanan, algoritma ini akan mencocokkan karakter per karakter pattern dengan karakter di teks yang bersesuaian, sampai salah satu kondisi berikut dipenuhi:
6. Karakter di pattern dan di teks yang dibandingkan tidak cocok (mismatch).
7. Semua karakter di pattern cocok. Kemudian algoritma akan memberitahukan penemuan di posisi ini.
8. Algoritma kemudian menggeser pattern berdasarkan tabel next (lompat), lalu mengulangi langkah 2 sampai pattern berada di ujung teks.

2.4 Pengujian Sistem

Menurut Safaat H(2015:59) Pengujian sistem yang dijelaskan meliputi Perancangan Model dalam UML (*Unifeld Modeling Languange*) yang terdiri dari *Usecase Diagram, Class Diagram, Activity Diagram, dan Sequence Diagram*. Selain itu juga ada pengujian hubungan antar *class* di android dengan modul penghubung dan pengujian sistem yang terdiri dari struktur menu dan perancangan *interface*.

Dalam pengujian sistem ini penulis akan melakukan proses pengujian dengan menggunakan *White Box Testing* yang berfungsi meramalkan cara kerja perangkat lunak secara rinci, oleh karena itu jalur logika perangkat lunak akan ditest dengan menyediakan *test case*. *Black Box Testing* berfungsi untuk menunjukan fungsi perangkat lunak tentang cara beroperasinya, Contoh input dan output data apakah sudah sesuai.

White-Box Testing **(Pengujian kotak putih)** yaitu menguji perangkat lunak dari segi desain dan kotak program apakah mampu menghasilkan fungsi-fungsi, masukan dan keluaran yang sesuai dengan spesifikasi kebutuhan. Pengujian kotak putih dilakukan dengan memeriksa

lojik dari kode program. Pembuatan kasus uji bisa mengikuti standar pengujian standar pemograman yang seharusnya. Contoh dari pengujian kotak putih misalkan menguji alur (dengan menelusuri) pengulangan (*looping*) pada logika pemograman.

Black-Box Testing **(Pengujian kotak hitam)** yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mencoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses login maka kasus uji yang dibuat adalah:

1. Jika *user* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
2. Jika *user* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tapi kata sandi salah, atau sebaliknya, atau keduanya salah.

2.5. Peralatan Pendukung (*Tools System*)

2.5.1. *Unifield Modelling Languange* (UML)

Analisa perancangan pada sistem ini menggunakan UML, yaitu terdiri dari Usecase Diagram, Class Diagram, Activity Diagram, dan Sequence Diagram. Menurut Safaat H (2015:60).

UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak,

dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemograman apapun. UML mendefinisikan beberapa diagram sebagai berikut:

1. *Activity Diagram*

   Activity Diagram merupakan alur kerja pada setiap usecase. *Activity diagram* pada analisa ini mencakup *activity diagram* setiap *usecase.* Menurut Safaat H(2015:62).

   Activity diagram merupakan state diagram kasus, dimana sebagaian besar state adalah *action* dan sebagian besar transisi di- *trigger* oleh selesainya state sebelumnya (*int*ernal *processing*). Menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses, dipakai pada *business modeling* untuk memperlihatkan urutan aktifitas proses bisnis. Struktur diagram ini mirip *flowchart* atau Data *flow* diagram pada perancangan terstruktur. Sangat bermanfaat apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan, dan *activity diagram* dibuat berdasarkan sebuah atau beberapa *use case* diagram.

2. *Usecase Diagram*

   *Us*ecase diagram merupakan suatu aktivitas yang menggambarkan urutan interaksi antara satu atau lebih aktor dan sistem. Menurut Safaat H(2015:60).

3. *Class Diagram*

   Class Diagram menggambarkan struktur dan deskripsi class, package, dan objek yang sering terhubung. *Class diagram* yang dijelaskan pada analisa ini adalah *class diagram* sistem yang terpasang pada perangkat android. Menurut Safaat H(2015:61).

4. *Sequence Diagram*

*S*equence diagram adalah representasi dari interaksi-interaksi ojek yang berjalan pada sistem. Dengan menggunakan *sequence diagram* kita dapat melihat objek-objek bekerja. *Sequence diagram* dapat menampilkan bagaimana sistem merespon setiap kegiatan atu permintaan dari *user*, dapat mempertahankan integritas internal, bagaimana data dipindahkan ke *user interface* dan bagaimana objek-objek diciptakan dan dimanipulasi. Setiap sistem memiliki dua kriteria, yaitu proses sederhana dan kompleks. Dengan demikian tidak seluruh proses pada sistem akan ditampilkan pada *sequence diagram*, melainkan hanya garis besarnya saja. Menurut Safaat H(2015:63).

5. *Deployment Diagram*

   *D*eployement diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, dimana komponen akan terletak (pada mesin, *server* atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal yang bersifat fisikal sebuah mode adalah *server*, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya. Menurut Safaat H(2015:64).

2.5.2 Google Maps Android V2 Api

API adalah kependekan dari *Application programming interface*. Dengan bahasa yang lebih sederhana, API adalah fungsi fungsi pemrograman yang disediakan oleh aplikasi atau layanan agar layananan tersebut bisa di integrasikan dengan aplikasi yang kita buat. *Google maps API* adalah fungsi fungsi pemrograman yang disediakan oleh *Google maps* agar *Google maps* bisa di integrasikan kedalam Web atau aplikasi yang sedang buat.