
UNIVERSITAS KOMPUTER INDONESIA
UNIKOM

BAB 2
TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori merupakan teori-teori pendukung yang berkaitan dengan penelitian yang sedang dilakukan. Teori pendukung dapat mempermudah dalam memahami aspek-aspek yang berkaitan dengan penelitian. Teori-teori pendukung pada penelitian ini adalah sebagai berikut :

2.1.1 Aplikasi

Aplikasi merupakan perangkat lunak yang dirancang untuk melakukan sebuah proses dengan menerima *input* data hingga menghasilkan *output* sesuai dengan tugas atau spesifikasi aplikasi bersangkutan. Para ahli memiliki pendapat yang beragam mengenai pengertian dari aplikasi. Berikut adalah pengertian aplikasi menurut ahli : [5]

a. Menurut Jogiyanto

   Menurut Jogiyanto aplikasi merupakan penggunaan dalam suatu komputer, instruksi atau pernyataan yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi output.

b. Menurut Verman

   Menurut Verman aplikasi adalah seperangkat instruksi khusus dalam komputer agar kita menyelesaikan tugas-tugas tertentu.

c. Menurut Eko. I dan Djokopran

   Menurut Eko. I dan Djokopran aplikasi merupakan proses atau prosedur aliran data dalam infrastruktur teknologi informasi yang dapat dimanfaatkan oleh para pengambil keputusan yang sesuai dengan jenjang dan kebutuhan.

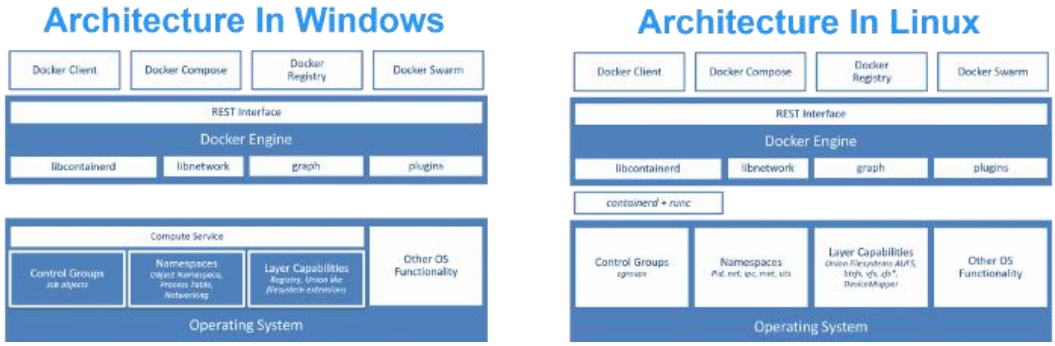
d. Menurut Yuhefizar

   Aplikasi merupakan program yang dikembangkan untuk memenuhi kebutuhan pengguna dalam menjalankan pekerjaan tertentu.

2.1.2 Sistem Operasi

Sistem operasi adalah mekanisme yang disediakan untuk pengguna agar dapat mengakses sumber daya perangkat keras [6]. Jadi sistem operasi akan menjadi penghubung antara perangkat keras dengan pengguna karena pengguna tidak dapat mengakses langsung sumberdaya dari perangkat keras. Ketika pengguna menggunakan perangkat lunak, pengguna tidak dapat langsung menginstallnya kedalam perangkat keras namun harus di install pada sistem operasi. Jadi aplikasi yang digunakan akan diakses melalui sistem operasi yang mengendalikan sumber daya perangkat keras, baik itu *hardisk, RAM, processor, dan VGA.*

Sistem operasi memiliki banyak jenisnya dan memiliki arsitektur tersendiri. Sistem operasi paling umum saat ini adalah windows, MAC OS, Linux, dan android untuk *mobile device*. Contoh arsitektur sistem operasi windows dan linux dapat dilihat pada gambar 2.1



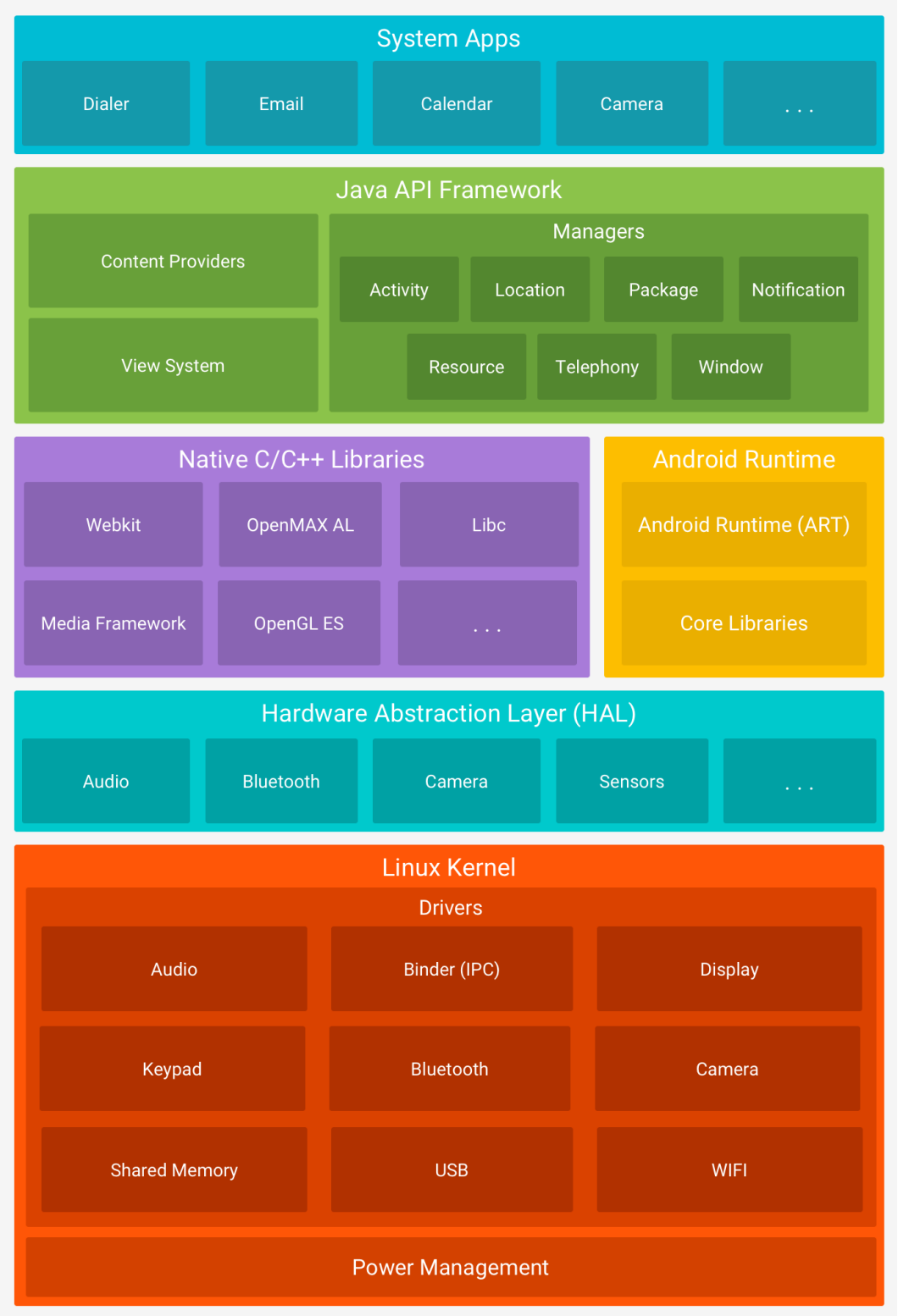
Sumber : https://medium.com/@putrasulung2108/windows-architecture-d2b022f136d3

Gambar 2. 1 Arsitektur Windows dan Linux

2.1.3 Android

Android merupakan sistem operasi yang digunakan pada mobile device (smartphone, smartwatch, tablet, televisi) dan sangat populer karena banyak merk smartphone yang mempercayakan android sebagai sistem operasinya. Sistem operasi android berbasis pada sistem operasi linux dan pertama kali dikembangkan oleh Android, Inc., yang didirikan oleh Andy Rubin, Rich Miner, Nick Sears, dan Chris White kemudian pada tahun 2005 dibeli oleh Google [7]. Hingga saat ini

proses pengembangan android masih dipegang oleh Google. Sistem operasi android bersifat open source atau gratis sehingga semua pihak dapat melakukan pengembangan software berbasis mobile dengan sistem operasi android. Sistem operasi android memiliki arsitekturnya sendiri dan berbeda dengan arsitektur windows. Gambaran arsitektur android dapat dilihat pada gambar 2.2



Sumber : https://developer.android.com/guide/platform/

Gambar 2. 2 Arsitektur android

Arsitektur android memiliki lapisan-lapisan yang didalamnya memiliki komponen masing-masing. Misalnya pada lapisan *system apps* merupakan lapisan terluar yang akan berhubungan dengan pengguna ketika menggunakan device. Kemudian pada lapisan *Java API Framework*, terdapat *activity manager, window manager, content providers,* dan lainnya. Lebih dalam lagi adalah lapisan libraries dan android runtime. *Hardware Abstraction Layer* (HAL) memberikan antarmuka standar yang mengungkap kemampuan perangkat keras ke *Java API Framework.* Kemudian lapisan paling dalam adalah *linux kernel*, yang akan berhubungan dengan perangkat keras.

Perkembangan android sangat pesat ditandai dengan banyaknya rilis dari sistem operasi ini. Tabel 2.1 menunjukkan beberapa versi yang dirilis oleh android [7] [8].

Tabel 2. 1 Versi-versi android

Versi Android	API Level	Tanggal Rilis	Codename
1.0	1	23 September 2008	Base
1.1	2	9 Februari 2009	Base
1.5	3	30 April 2009	Cupcake
1.6	4	15 September 2009	Donut
2.0/2.0.1/2.1	5/6/7	26 Oktober 2009	Eclair
2.2-2.2.3	8	20 Mei 2010	Froyo
2.3-2.3.2/2.3.3-2.3.7	9/10	6 Desember 2010	Gingerbread
3.0/3.1/3.2-3.2.6	11/12/13	22 Februari 2011	Honeycomb
4.0-4.0.2/4.0.3-4.0.4	14/15	19 Oktober 2011	Ice Cream Sandwich
4.1-4.1.2/4.2-4.2.2/4.3-4.3.1	16/17/18	27 Juni 2012	Jelly Bean
4.4-4.4.4/4.4w	19/20	31 Oktober 2013	KitKat
5.0.5.0.2/5.1-5.1.1	21/22	12 November 2014	Lollipop
6.0-6.0.1	23	5 Oktober 2015	Marshmallow
7.0	24	Agustus 2016	Nougat
7.1	25	Oktober 2016	Nougat

8.0	26	Agustus 2017	Oreo
8.1	27	Desember 2017	Oreo
9	28	Agustus 2018	Pie
10	29	Juli 2020	Q
11	30		R

Semakin tinggi versi dari android maka semakin beragam fitur-fitur yang ditawarkan didalamnya. Ketika versi android diperbaharui, tidak hanya penambahan fitur baru tetapi juga penghapusan fitur-fitur yang sudah tidak lagi relevan.

Pada tahun 2008 aplikasi android dapat diunduh melalui android market pada tautan http://android-market.co/ kemudian android market melebur bersama Google Music, Google Ebookstore menjadi Google Play atau lebih dikenal dengan Play Store dan dapat diakses melalui tautan https://play.google.com/store [7]. Developer dapat mempublish aplikasinya ke Play Store dan akan melalui tahapan-tahapan verifikasi sebelum akhirnya dapat dinikmati oleh pengguna. Pengguna juga dapat mengunduh beragam aplikasi di Play Store baik yang berbayar ataupun gratis. Play Store menyediakan beragam jenis aplikasi mulai dari pengolah dokumen, manipulasi gambar, game, hingga sosial media.

Android memiliki komponen-komponen yang membangunnya. Beberapa komponen bergantung pada komponen lainnya. Berikut adalah komponen-komponen pada android [9] :

1. *Activities*

   Activity menjadi representasi tampilan pada antarmuka pengguna. Misalnya aplikasi email memiliki satu *activity* yaitu untuk membuat email, namun pada satu *activty* tersebut dapat memiliki beberapa *activity* yang bekerjasama untuk menyelesaikan tugasnya.

2. *Services*

   Komponen *service* merupakan komponen yang bekerja pada belakang layar untuk menjalankan sebuah proses yang menunjang kebutuhan aplikasi. Aplikasi

atau proses berjalan dibelakang layar berarti aplikasi atau proses tersebut tidak muncul atau tampil pada tampilan antarmuka pengguna.

3. *Content Providers*

   Conten Provider berguna untuk mengatur seperangkat data aplikasi. Komponen ini berguna untuk membaca dan menulis data untuk disimpan pada sistem, SQLite, dan penyimpanan presisten lainnya.

4. *Broadcast Receivers*

   Broadcast Receivers adalah komponen yang merespon pengumuman *broadcast* ke semua sistem. Pada umumnya komponen ini berguna sebagai pintu gerbang untuk komponen lain dan dimaksudkan untuk melakukan jumlah pekerjaan yang sedikit.

2.1.4 Java

Java merupakan bahasa pemrograman yang dikembangkan pertama kali oleh James Gosling ketika masih bergabung di Sun Microsystems dan saat ini sudah diakuisisi oleh Oracle dan menjadi bagian dari Oracle [10]. Java dirilis pada tahun 1995 dengan syntax yang mirip dengan bahasa pemrograman C dan C++.

Aplikasi-aplikasi berbasis java akan dikompilasi kedalam p-code(bytecode) dan dijalankan pada berbagai Java Virtual Machine (JVM). JVM merupakan mesin virtual yang terdapat pada perangkat keras untuk mengeksekusi bytecode java sehingga java dapat berjalan diberbagai platform karena dieksekusi dengan JVM[8]. Program java dieksekusi dengan JRE (Java Runtime Environtment) yang dapat diperoleh dari situs Oracle untuk menjalankan aplikasi java di berbagai sistem operasi pada berbagai versi. Dengan kemampuannya yang dapat dijalankan pada berbagai JVM maka Java merupakan bahasa pemrograman yang diperuntukkan untuk tujuan umum (general purpose) dan dapat berjalan pada berbagai macam platform baik itu desktop atau mobile. Saat ini Java merupakan bahasa pemrograman yang cukup populer bahkan dapat digunakan untuk membangun website dengan bantuan berbagai framework yang tersedia.

Karena java dapat berjalan diberbagai platform menjadikan bahasa pemrograman java cukup populer. Seseorang dapat mempelajari java untuk tujuan

yang sangat banyak dan umum. Mereka mempelajari java dan dapat digunakan untuk mengembangkan mobile application, desktop application, atau web application dan tentunya dengan berbagai penyesuaian dengan masing-masing platform tersebut. Untuk mobile application java digunakan untuk membuat aplikasi pada sistem operasi android dan terbilang cukup tangguh karena bahasa pemrograman java dirilis sejak 1995 dan sudah mengalami pendewasaan diri yang cukup untuk mengatasi masalah-masalah modern yang mungkin dihadapi oleh programmer. Berikut adalah karakteristik dari bahasa pemrograman java [11] :

1. *Simple*

   Java adalah bahasa pemrograman yang sederhana namun memiliki kemampuan yang tinggi. Java menggunakan alokasi memori otomatis dan memiliki *garbage collection.*

2. *Object Oriented*

   Bahasa pemrograman java menggunakan paradigma berorientasi objek. Bahasa pemrograman yang berorientasi objek melihat hal-hal dalam pemrograman sebagai object.

3. *Distributed*

   Distributed Computing adalah metode komputerisasi dengan menggunakan beberapa komputer yang dihubungkan dengan jaringan untuk mengelola tugas-tugas tertentu. Java memiliki kemampuan networking yang baik, yang menjadikan programmer networking dapat mengirim dan menerima data dari sebuah file.

4. *Interpreted*

   Java adalah bahasa pemrograman yang menggunakan interpreter atau penerjemah untuk menjalankan program. Java menerjemahkan kode kedalam bahasa mesin agar dapat dimengerti oleh komputer.

5. *Robust*

   Java dapat diandalkan untuk berbagai keperluan dan dapat mengantisipasi berbagai macam gangguan dan kesalahan-kesalahan yang umum terjadi dalam pemrograman.

6. *Secure*

Java digunakan pada lingkungan *networking* dan terdistribusi. Jika kita mendownload java applet dan menjalankannye, kita tidak perlu khawatir tentang kerusakan yang mungkin akan ditimbulkan karena java tidak menyediakan akses secara bebas ke sistem secara langsung.

7. *Architecture-Neutral*

   Program yang dihasilkan oleh java tidak bergantung pada arsitektur komputer tertentu karena java dijalankan pada lingkungan JVM sehingga dapat berjalan pada arsitektur yang berbeda-beda.

8. *Portable*

   Karena java netral terhadap arsitektur komputer maka java dapat dijalankan dimanapun. Java dapat dijalankan pada mesin lainnya tanpa perlu dikompilasi ulang.

9. *Performance*

   Kinerja java dianggap lambat karena dijalankan melalui JVM namun kekurangan ini dapat teratasi dengan teknologi prosesor yang memiliki kecepatan proses yang tinggi.

10. *Multithreaded*

    Java memiliki kemampuan untuk melakukan proses atau tugas secara bersamaan dalam satu waktu. Kemampuan ini akan sangat berguna untuk pemrograman GUI dan jaringan.

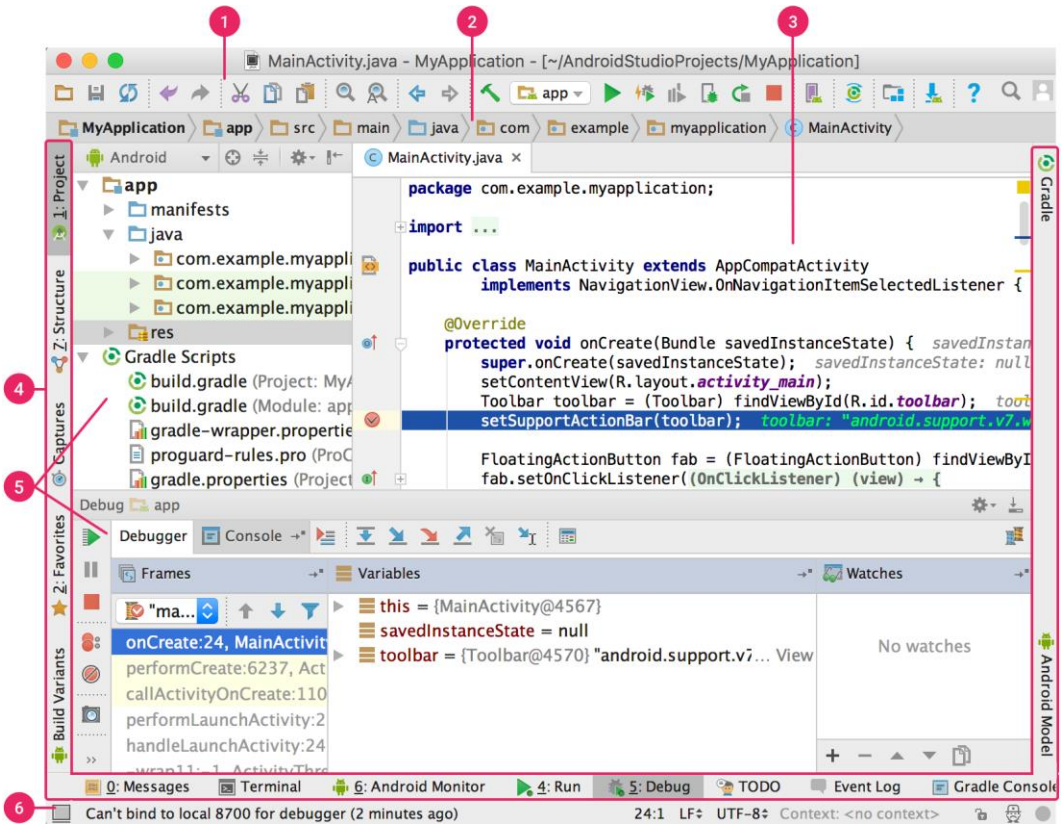
11. *Dynamic*

    Java didesain untuk dapat beradaptasi dengan lingkungan yang selalu berubah-ubah.

2.1.5 Android Studio

Android studio merupakan lingungan pengembangan terintegrasi (*Integrated Development Environtment* / IDE) yang diluncurkan oleh Google pada tanggal 16 Mei 2013 untuk mengembangkan aplikasi android dan dilengkapi dengan *Android System Development Kit* (SDK) yang berisikan *libraries dan build tools* yang dibutuhkan[6]. Android SDK sudah disertakan ketika menginstal android studio dan dapat diunduh secara gratis melalui

http://developer.android.com/sdk/index.html. Mengembangkan aplikasi android menjadi lebih mudah karena *libraries* dan *build tools* yang disediakan sudah mencukupi kebutuhan pembangunan aplikasi android baik itu dari segi pengembangan logika program ataupun pengembangan desain antarmuka. Layout antarmuka pengguna dapat dilihat pada gambar 2.3 [12]

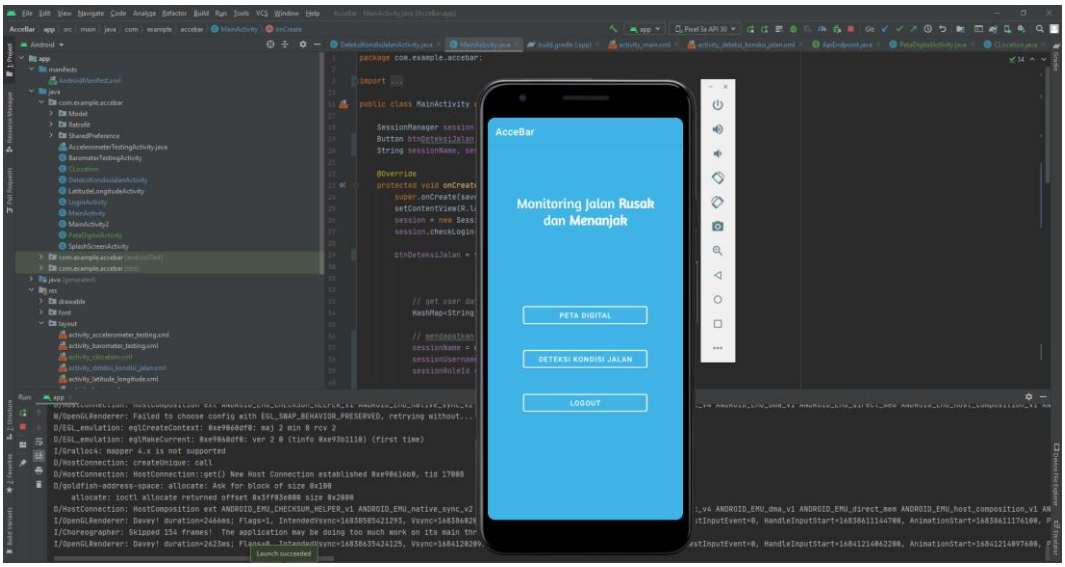


Gambar 2. 3 Layout tampilan antarmuka android studio

1. Toolbar untuk mengakses alat yang dibutuhkan selama proses pengembangan aplikasi termasuk untuk menjalankan aplikasi.
2. Menu Navigasi untuk menjelajahi project yang akan dibuka dan diedit.
3. Jendela Editor untuk memodifikasi kode sesuai dengan kebutuhan pengembangan.
4. Panel Jendela Fitur berada di pinggir tampilan dan digunakan untuk memunculkan fitur-fitur yang dibutuhkan.
5. Jendela Fitur memberikan akses pada fitur-fitur yang tersedia seperti pengelolaan project, penelusuran, kontrol versi, dan lainnya.

6. Status Bar untuk menampilkan status dari project yang sedang dikerjakan baik itu berupa kesalahan atau peringatan.

Ketika aplikas sudah selesai dibangun tentunya diperlukan uji coba aplikasi tersebut pada berbagai perangkat dengan variasi versi ataupun layar. Menggunakan banyak *smartphone* tentu bukan menjadi pilihan yang ekonomis. Android Studio menyediakan *Android Virtual Devices* (AVDs) yang sudah terintegrasi dengan Android Studio IDE yang dapat digunakan oleh pengembang untuk melakukan uji coba terhadap aplikasi yang sedang dibangun. Jenis perangkat keras yang dibuat secara virtual terdapat beberapa kategori diantaranya adalah *phone*, *tablet*, *wear*, dan TV. Selain menggunakan profile yang sudah tersedia juga dapat membuat spesifikasi perangkat keras yang baru dan dapat disesuaikan dengan kebutuhan. Setelah memilih spesifikasi perangkat keras yang akan dibuat virtual selanjutnya adalah memilih *system image* yang merupakan versi sistem operasi android. Android virtual device dapat dilihat pada gambar 2.4

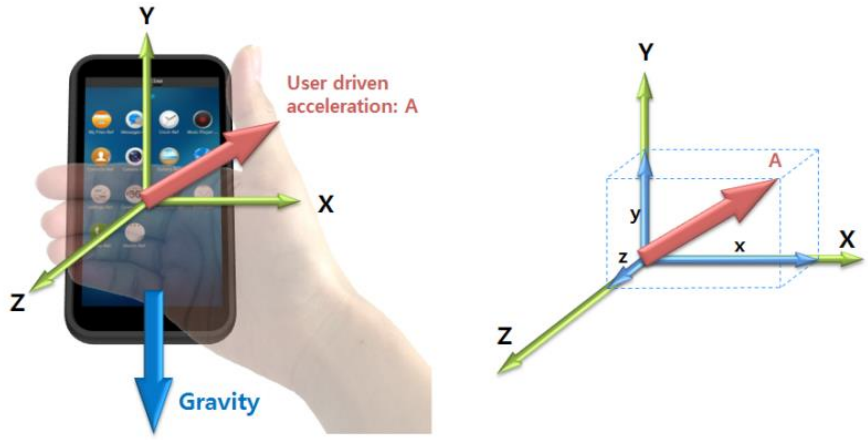


Gambar 2. 4 Android virtual device

2.1.6 Sensor Accelerometer

Accelerometer adalah sensor yang terdapat pada *smartphone* yang digunakan untuk untuk mengukur atau menghitung percepatan pada 3 dimensi [13]. Pada penelitian ini sensor accelerometer digunakan untuk mendeteksi jalan yang

rusak melalui getaran yang dihasilkan ketika kendaraan melalui jalan tersebut. Gambar sumbu koordinat pada accelerometer dapat dilihat pada gambar 2.5



Sumber : http://www.averagearticles.com

Gambar 2. 5 Gambar sumbu accelerometer

Accelerometer menghitung akselerasi tidak dengan menghitung seberapa cepat perubahan yang terjadi terhadap waktu, tapi dengan menghitung gaya. Secara umum hal ini dilakukan dengan merasakan seberapa besar tekanan yang diberikan ketika gaya tersebut terdorong [14]. Percepatan merupakan perubahan kecepatan suatu benda terhadap waktu. Percepatan terbagi menjadi dua yaitu percepatan statis misalnya gaya gravitasi bumi dan percepatan dinamis misalnya ketika terjadi getaran. Pada *accelerometer* percepatan yang terjadi adalah percepatan secara vertikal yaitu terhadap gravitasi bumi dan percepatan horizontal terhadap gerak secara horizontal. *Accelerometer* yang diletakkan di permukaan bumi dapat mendeteksi percepatan 1g (ukuran gravitasi bumi) pada titik vertikalnya, untuk percepatan yang dikarenakan oleh pergerakan horizontal maka *accelerometer* akan mengukur percepatannya secara langsung ketika bergerak secara horizontal.

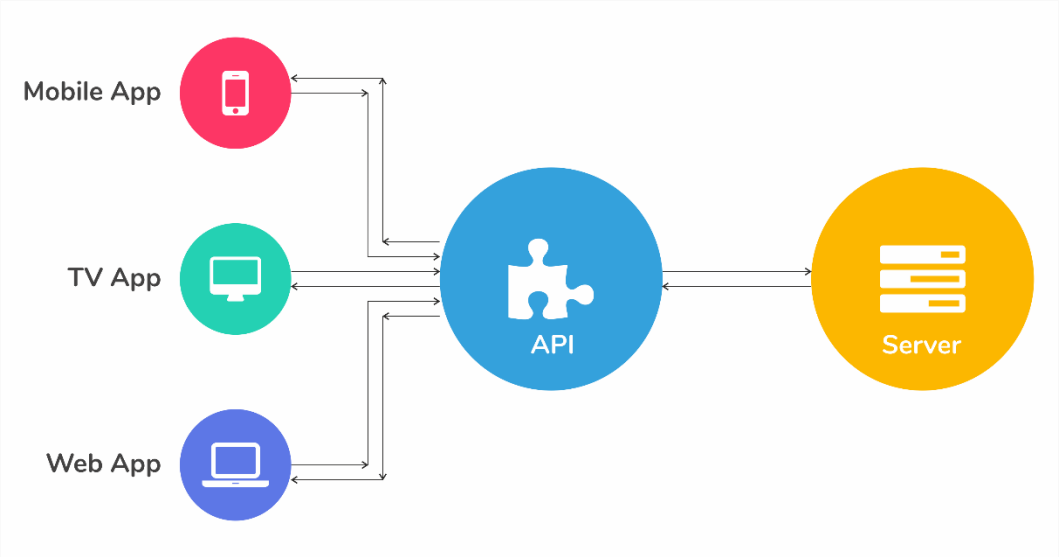
Smartphone saat ini umumnya sudah memiliki *built-in* sensor *accelerometer* untuk memenuhi kebutuhannya. Sensor *accelerometer* pada *smartphone* dapat digunakan untuk mendeteksi orientasi *device* dan memberikan fitur rotasi otomatis pada *smartphone*. Selain itu juga dapat dimanfaatkan untuk memutar musik berikutnya pada *playlist* musik. Pada bidang kesehatan *accelerometer* dapat dimanfaatkan untuk menghitung jumlah langkah dengan menempelkan *device* pada bagian tubuh tertentu, misalnya lengan yang nantinya

dapat dimanfaatkan untuk memprediksi jumlah kalori yang keluar dan rekomendasi asupan kalori yang dibutuhkan.

Sensor *accelerometer* juga ada pada mobil dan digunakan untuk memicu kantong udara ketika mobil mengalami kecelakaan. Sensor *accelerometer* pada mobil juga dapat digunakan untuk mengukur getaran akibat jalan rusak dan memiliki sensitifitas lebih tinggi daripada sensor *accelerometer* pada *smartphone*, namun sensor *accelerometer* pada *smartphone* lebih ekonomis karena merupakan *built-in* sensor.

2.1.7 API (*Application Programming Interfaces*)

Application Programming Interfaces (API) adalah pengembangan dan inovasi perangkat lunak dengan memungkinkan aplikasi bertukar data dan fungsionalitas dengan mudah dan aman [15]. API memungkinkan perusahaan untuk membuka data dan fungsi aplikasi mereka kepada pengembang pihak ketiga eksternal ataupun pihak internal dari departemen yang berebda. Pengembang tidak perlu tahu bagaimana API diimplementasikan, mereka hanya menggunakan antarmuka untuk berkomunikasi dengan produk dan layanan lain. Gambaran hubungan antara server dan aplikasi yang dihubungkan dengan API dapat dilihat pada gambar 2.6



Sumber : https://junaidsshaikh.medium.com/application-programming-interface-api-46762173b32a

Gambar 2. 6 Appliaction programming interface

API berada diantara aplikasi dan server web dan bertindak sebagai perantara yang memproses transfer data antar sistem. Cara kerja API adalah klien meminta request melalui API ke server web melalui *Uniform Resource Identifier* (URI) dan menyertakan kata kerja permintaan, header, dan terkadang isi permintaan. Setelah menerima permintaan yang valid, API melakukan panggilan ke server web. Kemudian server mengirimkan respon kepada API dengan informasi yang diminta. Setelah menerima informasi dari server, API mengirimkan informasi tersebut kepada pihak yang meminta.

API memiliki berbagai jenis dan banyak digunakan. Jenis-jenis API antara lain sebagai berikut :

1. Open API

   Open API adalah API yang dapat diakses melalui HTTP protocol dan bersifat gratis. Open API juga biasa disebut API public karena dapat digunakan secara bebas, terbuka, dan gratis.

2. Partner API

   Partner API adalah API yang dibuat untuk mendapatkan data atau informasi antar rekan bisnis agar dapat saling memenuhi kebutuhan informasi masing-masing.

3. Internal API

   Internal API adalah API yang bersifat tertutup dan tersembunyi hanya untuk internal perusahaan saja. Digunakan untuk berkomunikasi dan pertukaran data antar departemen atau bidang.

4. Composite API

   API Komposit adalah API yang dibuat dari gabungan beberapa sumber untuk kemudian diakses melalui satu pintu *end point*.

Seiring dengan perkembangan penggunaan API, beberapa protokol telah dikembangkan untuk memberikan standarisasi atau aturan untuk menentukan jenis data dan perintah yang diterima. Berikut adalah jenis-jenis protokol yang dapat digunakan [15] :

1. SOAP (*Simple Object Access Protocol*)

   SOAP adalah protokol API yang dibangun dengan XML dan mengirim atau menerima data melalui SMTP dan HTTP.

2. XML-RPC

   XML-RPC adalah protokol API yang mengandalkan format XML tertentu untuk mentransfer data dan lebih tua dari SOAP namun lebih sederhana dan relatif ringan karena menggunakan bandwidth minimum.

3. JSON-RPC

   JSON-RPC sama dengan XML-RPC tetapi menggunakan JSON untuk mentransfer data buka format XML.

4. REST (*Representational State Transfer*)

   REST (Representational State Transfer) adalah sekumpulan prinsip arsitektur API web, yang berarti tidak ada standar resmi (tidak seperti yang memiliki protokol). Untuk menjadi REST API (juga dikenal sebagai RESTful API), antarmuka harus mematuhi batasan arsitektur tertentu. RESTful API dapat dibuat dengan protokol SOAP, tetapi kedua standar tersebut biasanya dipandang sebagai spesifikasi yang bersaing.

2.1.7.1 Global Positioning System (GPS)

Global Positioning System (GPS) adalah suatu sistem navigasi menggunakan lebih dari 24 satelit MEO (*Medium Earth Orbit* atau *Middle Earth Orbit*) yang mengelilingi bumi sehingga penerima-penerima sinyal di permukaan bumi dapat menangkap sinyalnya [16]. GPS dapat menentukan letak, arah, kecepatan, dan waku melalui sinyal-sinyal yang diterima tersebut. Nama formal GPS adalah NAVSTAR GPS, kependekan dari *Navigation Satellite Timing and Ranging Global Positioning System*.

Untuk mengetahui posisi seseorang diperlukan alat bernama *GPS Recheiver* yang berfungsi untuk menerima sinyal yang dikirim oleh *GPS Recheiver* berbentuk modul yang berisi beberapa *integrated circuit* (IC) yang berisi data posisi [16]. Sinyal yang diterima tersebut di analisis dan dapat berubah menjadi titik-titik atau way-point. Way-point tersebut berupa titik-titik koordinat lintang dan bujur dari penerima sinyal tersebut kemudian ditransformasikan menjadi peta digital.

Dengan adanya GPS sangat membantu dalam menyelesaikan berbagai masalah. GPS dapat digunakan pada peta digital untuk menunjukkan posisi dari penerima sinyal. Selain itu GPS juga dapat digunakan untuk menuju suatu tempat yang rutenya tidak diketahui pengguna sehingga dapat membantu pengguna mencapai tujuan tanpa tersesat. Selain itu juga GPS dapat dimanfaatkan untuk melacak kendaraan baik itu mobil ataupun motor untuk mengetahui lokasi kendaraan apabila terjadi pencurian.

2.1.8 Algoritma *Realtime Pathole Detection*

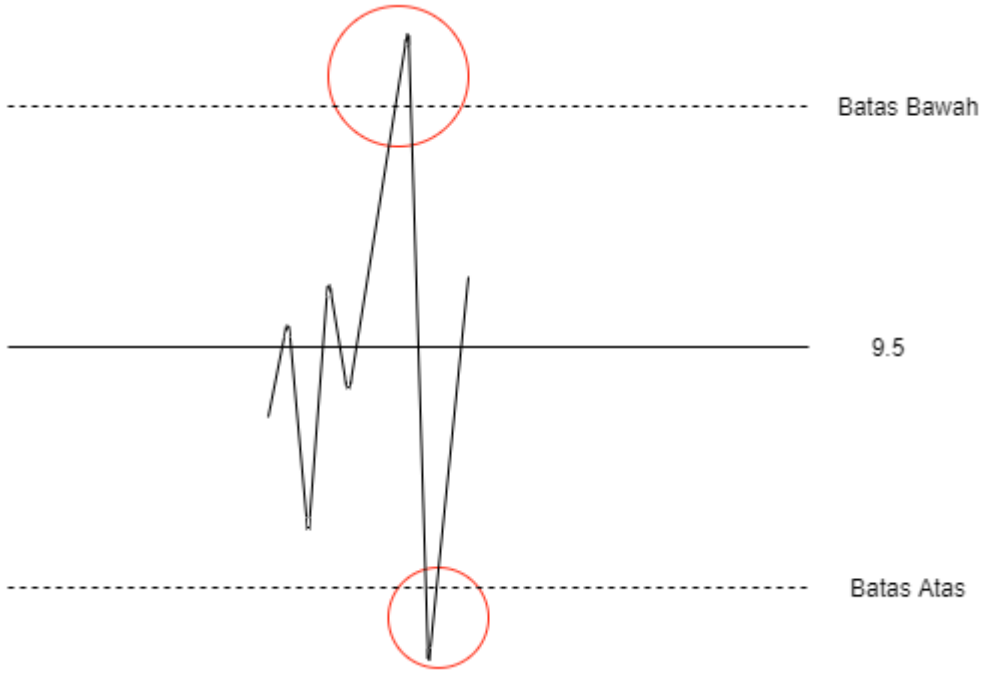
Algoritma yang dapat digunakan untuk mendeteksi jalan rusak diantaranya terdapat 4 macam, yaitu Z-Thresh, Z-Diff, Stdev-z, dan G-Zero. Dari keempat algoritma tersebut, yang paling cocok dan memiliki akurasi terbaik digunakan untuk mendeteksi jalan rusak secara realtime adalah kombinasi antara Algoritma Z-Thresh dan G-Zero [17].

Algoritma Z-Thresh adalah algoritma yang mempertimbangkan nilai minimum dari data sumbu akselerometer sebagai ambang batas untuk mendeteksi lubang [17]. Nilai akselerometer pada sumbu z ketika diletakkan secara horizontal

adalah 9.8 m/s^2. Sumbu data akselerometer akan turun dengan cepat saat kendaraan masuk ke sebuah lubang dan meningkat ketika kendaraan meninggalkan lubang. Algoritma ini mempertimbangkan nilai terendah dari akselerometer saat memasuki lubang. Selanjutnya, nilai maksimum dari nilai minimum akselerometer akan dipilih sebagai ambang batas. Apabila z kurang dari ambang batas maka z dianggap sebagai jalan rusak.

Algoritma G-Zero adalah algoritma yang mempertimbangkan ketiga sumbu *accelerometer* untuk menentukan batas bawah dan batas atas [17]. Ketika kendaraan memasuki jalan berlubang, nilai ketiga sumbu accelerometer mendekati nol. Oleh karena itu algoritma ini menganggap nilai tertinggi dari ketiga sumbu accelerometer sebagai kandidat batas bawah dan nilai terkecil dari ketiga sumbu accelerometer sebagai kandidat batas atas. Penentuan nama batas bawah atau batas atas dilakukan terhadap sumbu z. Jadi disebut batas bawah karena ketika sumbu z sangat tinggi, maka batas ini akan berada dibawah sumbu z tersebut ketika nilai z melewati batas bawah maka disebut batas bawah. Begitu juga sebaliknya disebut batas atas karena ketika sumbu z sangat rendah, maka batas ini akan berada diatas sumbu z tersebut ketika nilai z melewati batas maka disebut batas atas. Nilai minimum dari nilai maksimum ketiga sumbu setiap kali uji coba akan menjadi batas bawah. Nilai maksimum dari nilai minimum ketiga sumbu setiap kali uji coba akan menjadi batas atas. Deteksi akan bernilai 1 atau true apabila nilai z lebih dari batas bawah atau z kurang dari batas atas.

Kombinasi antara Z-Thresh dan G-Zero menjadikan sebuah algoritma *realtime pathole detection* untuk mendeteksi jalan rusak dengan mempertimbangkan sumbu z ke atas dan kebawah. Jadi akan terdapat dua ambang batas, yaitu batas bawah dan batas atas. Apabila sumbu z melewati batas tersebut maka data akan dianggap sebagai jalan berlubang. Dapat ditulis dengan $\theta_1 \leq z \leq \theta_2$ dengan θ_1 sebagai batas bawah dan θ_2 sebagai batas atas.



Gambar 2. 7 Algoritma Realtime Pathole Detection

Disebut batas atas atas karena batas ini berada diatas sumbu z ketika sumbu z melewati ambang batas dan disebut batas bawah karena batas ini berada dibawah sumbu z ketika sumbu z melewati ambang batas karena kedua batas ini dilihat terhadap sumbu z. Untuk mendapatkan nilai batas atas dan batas bawah ini diperlukan uji coba pada jalan berlubang. Percobaan pada jalan berlubang untuk mengetahui nilai ketiga sumbu accelerometer ketika memasuki lubang. Ketiga sumbu tersebut dikumpulkan dan yang menjadi perhatian adalah nilai sumbu ketika kendaraan memasuki lubang. Ketika memasuki lubang, nilai sumbu maksimum dan minimum dikumpulkan. Setelah itu, ambil nilai terendah dari kumpulan nilai maksimum untuk dijadikan batas bawah dan ambil nilai tertinggi dari kumpulan nilai minimum untuk dijadikan batas atas. Setelah itu batas-batas tersebut dapat digunakan untuk memfilter data jalan.

2.1.9 Leaflet

Leaflet adalah pustaka JavaScript *open source* untuk peta interaktif *mobile-friendly* [18]. Leaflet dikembangkan dengan mempertimbangkan kesederhanaan, kegunaan, dan kinerja. Leaflet dapat digunakan di berbagai platform desktop dan

mobile serta memiliki banyak *plugins,* mudah digunakan, terdokumentasi dengan baik dan sederhana. Leaflet awalnya dibuat oleh Vladimir Agafonkin namun saat ini sudah dikembangkan oleh komunitas dan banyak kontributor yang terlibat untuk mengembangkan leaflet agar dapat memberikan manfaat yang lebih banyak kepada developer.

Tools yang disediakan di leaflet sudah sangat *powerfull* untuk digunakan oleh developer dalam mengembangkan aplikasi. Misalnya seperti aplikasi GIS atau sistem informasi geografis yang bertumpu pada peta sebagai fitur utamanya dalam menampilkan informasi. Leaflet dapat memberikan *marker* atau tanda pada titik koordinat tertentu dan marker tersebut juga dapat di custom sesuai dengan keingingan developer. Selain itu leaflet juga bisa memberikan tanda berupa radius lingkaran ataupun *polygon* yang dapat dicustom sesuai dengan kebutuhan. Untuk lingkaran misalnya dapat digunakan untuk memberikan informasi mengenai efek dari letusan gunung berapi. Contoh sederhana penggunaan marker, lingkaran, dan polygon dapat dilihat pada tabel 2.8

Gambar 2. 8 Marker, lingkaran, dan polygon pada leaflet

Leaflet dapat digunakan dengan memanggil beberapa method sebagai berikut :

```
var map = L.map('map').setView([51.505, -0.09], 13);
```

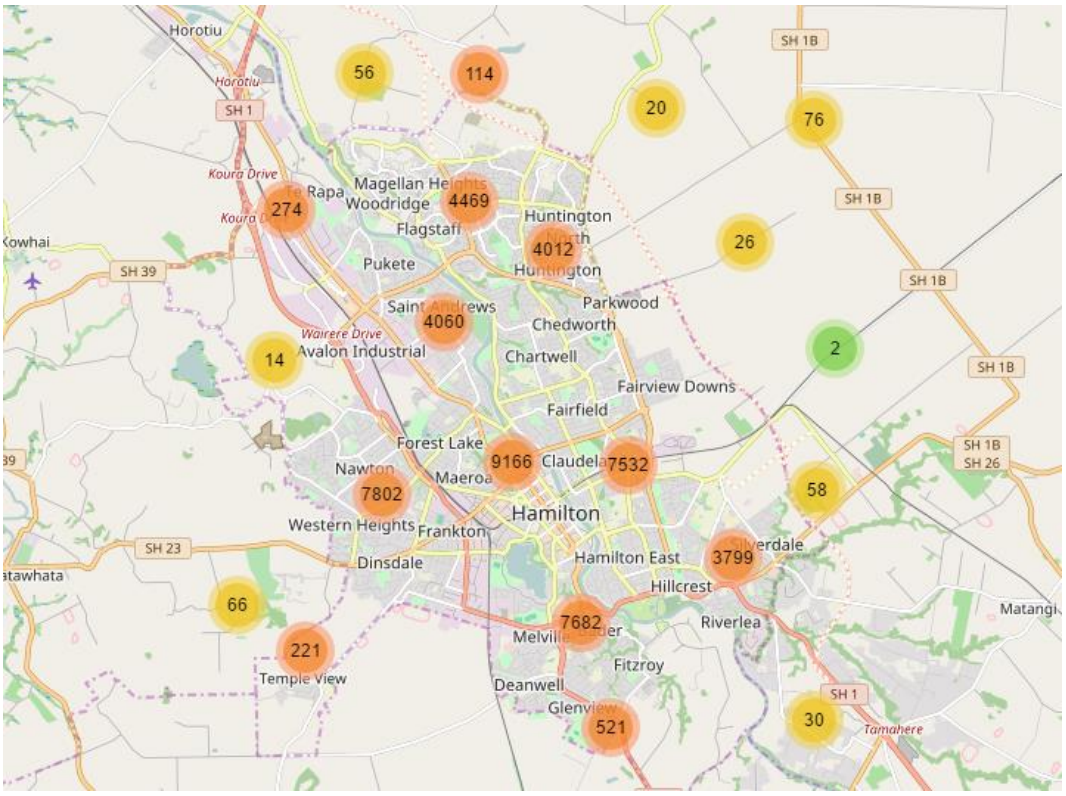
```
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
}).addTo(map);

L.marker([51.5, -0.09]).addTo(map)
    .bindPopup('A pretty CSS3 popup.<br> Easily customizable.')
    .openPopup();
```

Variabel map akan menjadi wadah untuk menambahkan layer dan marker kedalam map. Kita dapat menset koordinat apda saat pertama kali aplikasi dijalankan dengan method setView yang menerima 2 parameter yaitu koordinat yang terdiri dari latitude dan longtiude serta max zoom untuk memberikan batas maksimum zoom peta.

Leaflet dibuat seringan mungkin karena terfokus pada inti dari fungsinya, bahkan file javascriptnya saja hanya 39 KB. Maka leaflet menggunakan *third party plugins* atau pihak ketiga untuk mengembangkan fitur-fitur yang dapat membantu meningkatkan fungsionalitas dari leaflet. Berikut adalah beberapa plugins yang dapat digunakan pada leaflet yang disediakan oleh pihak ketiga :

1. Marker Cluster

   Clustering adalah pengelompokan data berdasarkan kategori terdekat. Leaflet memiliki plugins yang digunakan untuk mengkluster atau mengelompokkan marker yang saling berdekatan sebagai satu marker sehingga marker-marker tersebut tidak saling tumpang tindih dan menjadi sulit untuk dibaca. Contoh plugins clustering adalah MarkerCluster yang dikembangkan oleh Dave Leaver. Penerapan marker cluster dapat dilihat pada gambar 2.9



Gambar 2. 9 Contoh penerapan marker cluster

Contoh penggunaan marker cluster adalah sebagai berikut :

```
var markers = L.markerClusterGroup();
markers.addLayer(L.marker(getRandomLatLng(map)));
... Add more layers ...
map.addLayer(markers);
```

2. Routing Machine

Leaflet Routing Machine merupakan plugin yang dikembangkan oleh Per Liedman untuk mencari rute berdasarkan titik-titik *waypoint*. Untuk mendapatkan sebuah rute minimal dibutuhkan dua titik *waypoints* agar dapat diketahui rute antara kedua titik tersebut. Plugin Leaflet Routing Machine menggunakan dapat menggunakan beberapa *routing enginge* diantaranya adalah OSRM, Mapbox Direction API, GraphHopper, Mapzen Valhalla, TomTom Online Routing API, dan Esri. Secara default plugin ini menggunakan OSRM (*Open Source Routing Machine*). Contoh penggunaan Leaflet Routing Machine adalah sebagai berikut :

```
Var routing = L.Routing.control ({
            Waypoints: [
                    L.latLng(57.74, 11.94),
```

```
                        L.latLng(56.45, 12.82)
                ]
    ).addTo(map);
```

2.1.10 Jalan

Jalan merupakan prasarana transportasi kendaraan darat yang meliputi segala bagian jalan , termasuk bangunan pelengkap dan perlengkapannya yang diperuntukkan bagi lalu lintas yang berada pada permukaan tanah [19]. Kendaraan yang melalui jalan raya banyak jenisnya mulai dari yang ringan hingga kendaraan yang berat. Kendaraan ringan seperti sepeda atau sepeda motor dan kendaraan berat seperti mobil hingga truk bermuatan. Beban yang diberikan oleh kendaraan terhadap jalan raya berbanding terbalik dengan umur jalan raya tersebut. Selain karena beban, faktor lain yang memengaruhi umur dari jalan raya adalah iklim. Pada negara beriklim tropis seperti Indonesia, panas matahari dan curah hujan yang tinggi dapat mengakibatkan kerusakan dini pada jalan raya [2]. Kerusakan perkerasan aspal dapat mengakibatkan jalan bergelombang dan retak-retak.

Terdapat beberapa pengklasifikasian jalan yaitu berdasarkan peruntukannya dan berdasarkan kelasnya. Klasifikasi jalan berdasarkan peruntukannya adalah sebagai berikut [20]:

1. Jalan Umum

   Jalan umum merupakan jalan yang diperuntukkan untuk lalu lintas umum dan dikelola oleh negara. Jalan umum dikelompokkan atas :

   a. Jalan nasional.
   b. Jalan provinsi.
   c. Jalan kabupaten.
   d. Jalan kota.
   e. Jalan desa.

2. Jalan Khusus.

   Jalan khusus merupakan jalan yang dikelola oleh orang atau instansi untuk melayani kepentingan sendiri.

Klasifikasi jalan berdasarkan kelasnya adalah sebagai berikut :

1. Jalan bebas hambatan.

Jalan bebas hambatan adalah jalan yang memiliki minimal 2 jalur untuk masing-masing arah, tidak memiliki persimpangan, dilengkapi dengan pagar dipinggir jalan, dan lebar jalur minimal 3.5 meter.

2. Jalan raya.

   Jalan raya adalah jalan umum untuk lalu lintas secara terus menerus.

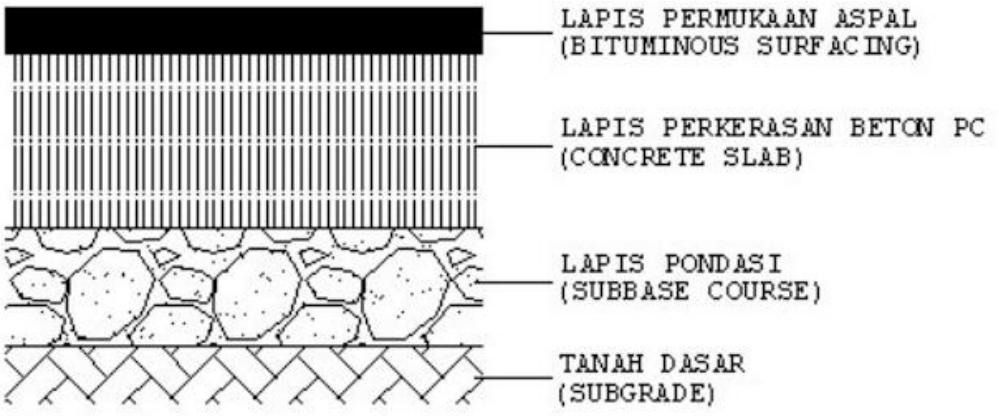
3. Jalan sedang.

   Jalan sedang adalah jalan umum dengan lalu lintas jarak sedang.

4. Jalan kecil.

   Jalan kecil adalah jalan umum untuk melayani lalu lintas setempat.

Klasifikasi jalan berdasarkan perkerasannya dibagi menjadi 3 yaitu konstruksi perkerasan lentur atau *felxible pavement*, konstruksi perkerasan kaku atau *rigid pavement*, dan konstruksi komposit atau *composite pavement*. Untuk perkerasan lentur yang digunakan adalah aspal yang dipanaskan hingga menjadi cair, kemudian dicampurkan dengan bahan lainnya seperti batu atau kerikil untuk menjadi pondasi jalan. Kemudian perkerasan kaku atau *rigid pavement* juga dikenal dengan jalan beton. Pelat beton diletakkan diatas tanah sebagai pondasi jalan dan menahan beban dari kendaraan. Kemudian perkerasan komposit adalah perpaduan antara perkerasan lentur dan perkerasan kaku yang disesuaikan dengan kebutuhan beban yang ditanggung. Gambaran struktur konstruksi komposit dapat dilihat pada gambar 2.9



Gambar 2. 10 Konstruksi perkerasan komposit

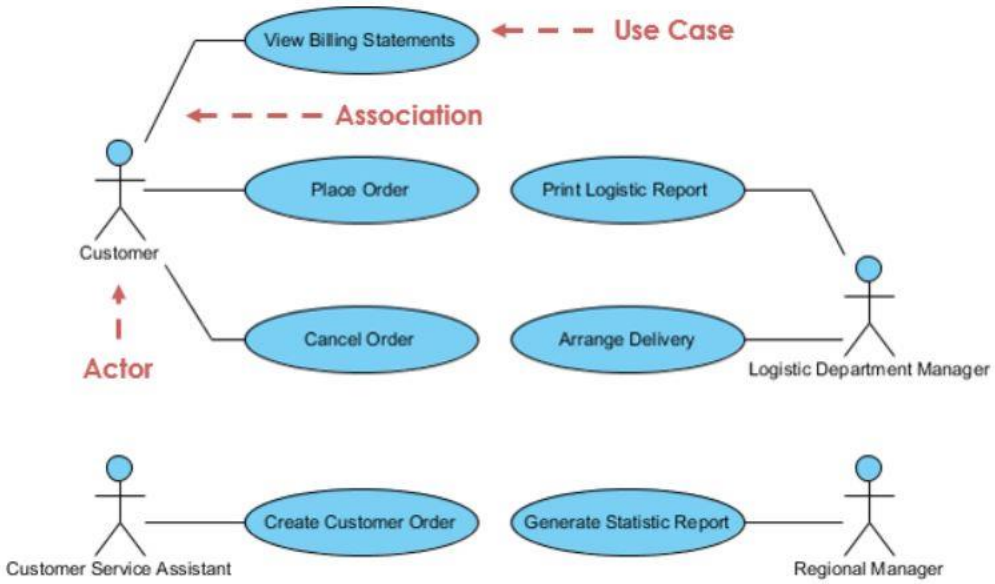
2.1.11 *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah bahasa pemodelan standar yang terdiri dari serangkaian diagram terintegrasi, dikembangkan untuk membantu pengembang sistem dan perangkat lunak untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan artefak sistem perangkat lunak, serta untuk pemodelan bisnis dan sisten non-perangkat lunak lainnya [21]. Beberapa hal mendasar pada model UML adalah sebagai berikut :

1. *Classes*, adalah kelompok dari objek yang mempunyai atribut, operasi, dan hubungan semantik. Kelas dapat digambarkan sebagai sebuah persegi panjang, memiliki nama dan metode pengoperasian.
2. *Interfaces*, adalah sebuah antarmuka yang menghubungkan antarkelas. Umumnya interface digambarkan dengan sebuah lingkaran beserta nama interface.
3. *Collaboration*, adalah interaksi dan sebuah kumpulan dari kelas atau elemen yang bekerja bersamaan.
4. *Use cases*, adalah uraian kelompok yang saling berkaitan dan dilakukan oleh aktor. *Use case* digunakan sebagai bentuk tingkah laku objek dalam sebuah model.
5. *Nodes*, merupakan fisik dari elemen yang ada pada saat dijalankannya sebuah sistem.

2.1.11.1 *Use Case Diagram*

Use case diagram menjelaskan persyaratan fungsional sistem dalam hal kasus penggunaan dan menghubungkan apa yang dibutuhkan dari sistem ke bagaimana sistem memenuhi kebutuhan tersebut [21]. Diagram ini merupakan alat yang sangat kuat dalam pengembangan perangkat lunak karena menyediakan gambaran umum dari fungsionalitas yang harus dipenuhi oleh sistem terhadap lingkungannya. Contoh *use case* diagram dapat dilihat pada gambar 2.10



Sumber : https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/#use-case-diagram

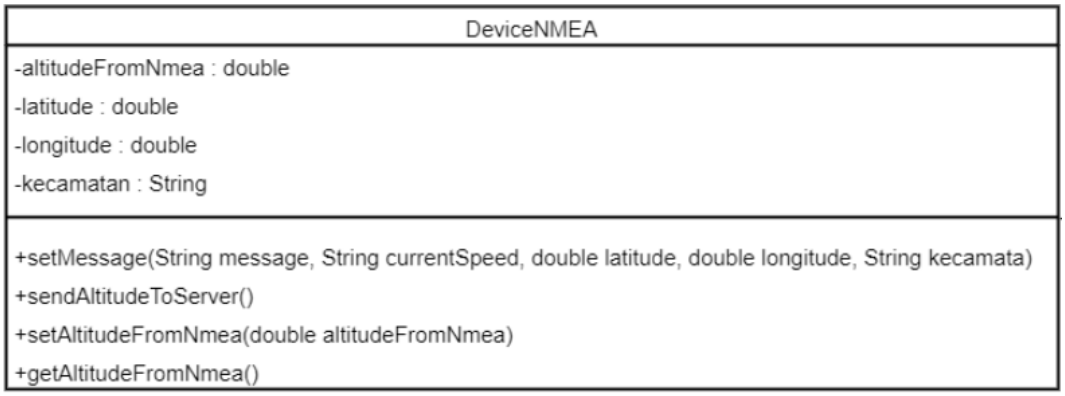
Gambar 2. 11 Use case diagram

Tujuan dari *use case diagram* adalah untuk menentukan konteks dari sistem, menggambarkan kebutuhan sistem, dan memvalidasi arsitektur dari sistem. *Use case diagram* memiliki beberapa notasi untuk menggambarkan diagramnya. Notasi aktor digambarkan seperti *stickman*, sebagai representasi dari seseorang yang berinteraksi dengan sistem. Kemudian *use case* berbentuk oval dengan nama dari *use case* tersebut, untuk merepresentasikan funsgi dari sistem. Antara *use case* dan aktor dihubungkan dengan garis komunikasi. *Boundary of system* atas batas sistem berbetnuk persegi panjang untuk merepresentasikan batas-batas yang menjadi bagian sistem.

Hubungan antar *use case* terdiri dari *extend, include,* dan *generalization* [22]. *Extend* menggambarkan hubungan yang diperpanjang dari satu *use case* ke *use case* lainnya. *Include* menggambarkan sebuah fungsionalitas *use case* membutuhkan *use case* lainnya dalam memenuhi alur proses bisnisnya. *Generalization* menggambarkan hubungan antara *parent-child* pada *use case.*

2.1.11.2 *Class Diagram*

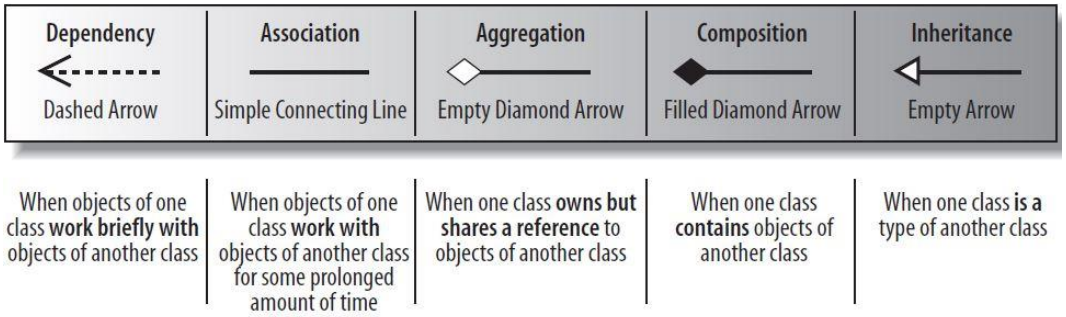
Dalam *software engineering, class diagram* adalah salah satu UML yang memanfaatkan konsep kelas dalam pemodelan digambarkan berbentuk segi empat dan dipisahkan secara vertikal menjadi 3 bagian yaitu nama kelas, atribut atau informasi mengenai kelas, kemudian operasi atau tingkah laku yang dapat dilakukan oleh kelas [23]. Kelas memiliki *visibility* yang merupakan keterangan untuk menunjukkan hak akses kelas lain terhadap atribut atau operasi didalam kelas. Masing-masing *visibility* memiliki notasi masing-masing dan digunakan sesuai dengan kebutuhan. *Public visibility* dinotasikan dengan tanda (+) dan dapat diakses dari kelas manapun. *Protected visibility* dinotasikan dengan tanda (#) dan tidak dapat diaksesk oleh semua kelas. *Protected* dapat diakses oleh kelas yang menjadi bagiannya juga misalnya kelas turunannya. Kemudian *package visibility* dinotasikan dengan (~) dan dapat diakses oleh kelas lain didalam package yang sama. Contoh class dapat dilihat pada gambar 2.11



Gambar 2. 12 Contoh kelas beserta atribut dan method

Kelas tidak bekerja sendiri-sendiri melainkan bekerja bersama-sama dengan kelas lainnya. Masing-masing kelas dihubungkan dengan garis yang menjelaskan jenis hubungan antar kelas tersebut. Hubungan antar kelas terdiri dari *inheritence, association, aggregation, composition,* dan *dependency* [23]. *Inheritence* menunjukkan hubungan pewarisan, antara superclass dan childclass. Jadi kelas yang berada diatas mewarisi properti dan method pada kelas dibawahnya. Ketika objek suatu kelas bekerja sama dengan objek kelas lainnya dalam waktu yang lama maka kelas tersebut dihubungkan dengan *association*. *Aggregation* menunjukkan

hubungan ketika satu kelas memiliki tetapi berbagi referensi ke objek dari kelas lain. *Composition* adalah tipe hubungan antar kelas yang menunjukkan kepemilikan yang lebih kuat. Ketika kelas yang memilikinya dihapus maka kelas yang menjadi bagiannya juga ikut terhapus. *Dependecny* merupakan hubungan saling ketergantungan, ketika satu kelas saling bekerja sama dengan kelas lainnya maka kelas tersebut akan memiliki ketergantungan. Notasi hubungan antar kelas dapat dilihat pada gambar 2.12 [23].



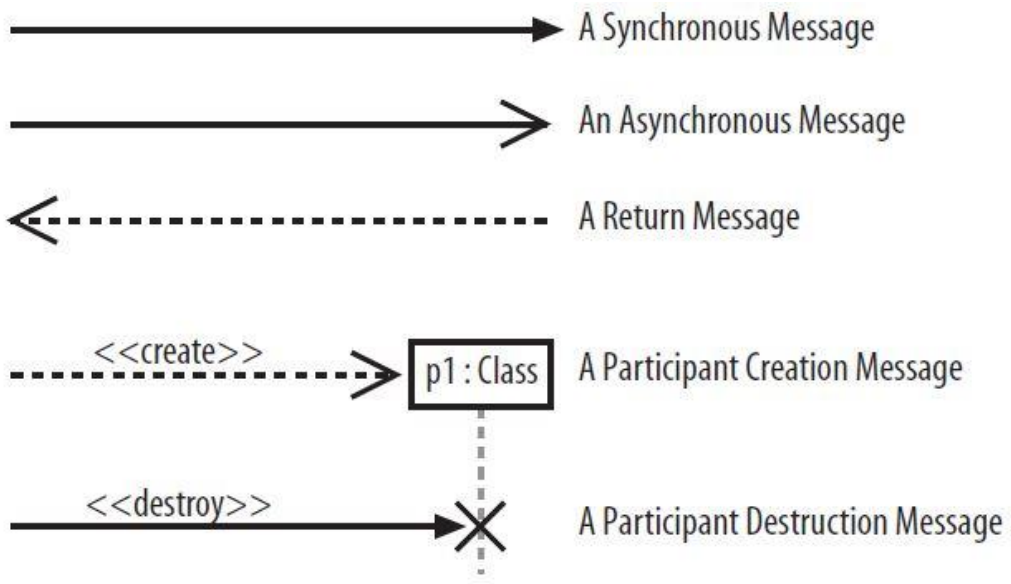
Gambar 2. 13 Hubungan antar kelas

2.1.11.3 *Sequence Diagram*

Sequence diagram adalah diagram yang dibuat dari kumpulan partisipan, bagian dari sistem yang berinteraksi satu sama lain [23]. Partisipan selalu ditempatkan secara horizontal terhadap partisipan lainnya tanpa tumpang tindih pada satu garis vertikal yang sama. Setiap partisipan memiliki garis hidup yang mengarah secara vertikal kebawah. Berfokus pada waktu dan menunjukkan urutan interaksi secara visual dengan menggunakan sumbu vertikal diagram untuk mewakili waktu, pesan apa yang dikirim, dan kapan.

Sequence diagram menunjukkan elemen saat berinteraksi dari waktu ke waktu dan mereka diatur secara horizontal untuk objek dan vertikal untuk waktu. Secara konvensional, objek yang terlibat dalam operasi terdaftar dari kiri ke kanan sesuai dengan saat mereka mengambil bagian dalam urutan pesan. Namun, elemen pada sumbu horizontal dapat muncul dalam urutan apapun. Waktu dalam diagram yang dimaksud adalah urutan bukan durasi. Ruang vertikal dalam diagram tidak relevan untuk durasi interaksi. Tanda panah pesan pada sequence diagram secara

umum dibagi menjadi 2 jenis yaitu *synchronous* dan *asynchronous*. Tanda panah pesan dapat dilihat pada gambar 2.13

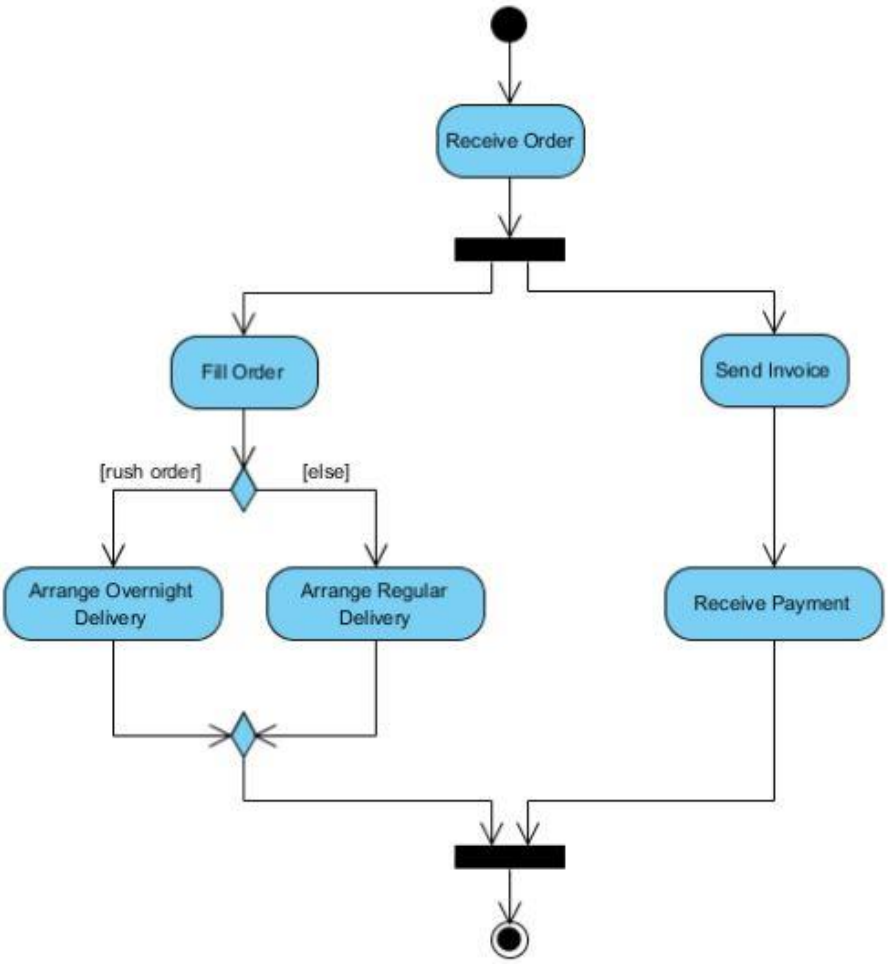


Gambar 2. 14 Tanda panah pesan

Pesan *synchronous* dipanggil ketika pemanggil pesan menunggu penerima pesan kembali dari pemanggilan pesan. Sedangkan pesan *asynchronous* berlaku sebaliknya. Pemanggil pesan dapat melakukan pemanggilan pesan tanpa perlu menunggu penerima pesan. Sebuah partisipan dapat memberikan pesan balik dinotasikan dengan tanda panah kosong dengan garis putus-putus seperti pada gambar 2.13. Tanda silang menandakan bahwa garis hidup tersebut telah dihancurkan.

2.1.11.4 *Activity Diagram*

Activity diagram menjelaskan bagaimana aktifitas dikoordinasikan untuk menyediakan layanan yang dapat berada pada berbagai tingkat abstraksi [24]. *Activity diagram* menggambarkan dengan lebih detail dari fungsionalitas sistem, pada berbagai tingkat abstraksi. Kejadian yang digambarkan termasuk pada kejadian ketika kondisi terpenuhi atau tidak terpenuhi. Contoh *activity diagram* dapat dilihat pada gambar 2.14



Sumber : https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/

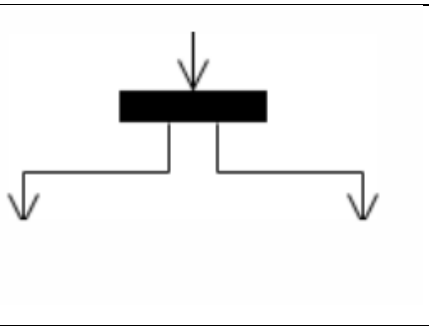
Gambar 2. 15 Activity diagram

Elemen-elemen pada *activity diagram* direpresentasikan pada berbagai notasi. Notasi pada *activity diagram* dapat dilihat pada tabel 2.2

Tabel 2. 2 Notasi activity diagram

No	Deskripsi Notasi	Notasi
1	*Activity* : Merepresentasikan sebuah aktifitas	

2	*Action : sebuah tugas yang sedang dijalankan*	
3	*Control Flow :* menunjukkan urutan dari eksekusi	
4	*Initial node :* menunjukkan awal dari serangkaian aktifitas	
5	*Final node :* menunjukkan akhir dari serangkaian aktifitas	
6	*Decision node :* merepresentasikan kondisi yang diuji coba dan untuk mengendalikan alur aktifitas	
7	*Merge node* : menggabungkan jalur keputusan yang berebda.	

8	*Fork node :* membagi perilaku menjadi sekumpulan arus aktifitas yang paralel atau bersamaan.	
9	*Join node :* menggabungkan sekumpulan arus aktifitas.