BAB II
TINJAUAN PUSTAKA

2.1 Landasan Teori

Landasan teori menjelaskan beberapa teori yang berkaitan dengan permasalahan yang akan dibahas sebagai pemahaman sistem yang akan dibangun.

2.2 Sepatu

Sepatu adalah jenis alas kaki (*footwear*) yang biasanya terdiri dari bagian-bagian kap (tutup), tali, sol, hak dan lidah. Normalnya, ukuran sepatu mengikuti beberapa standar-standar sepatu berbeda diseluruh dunia. Penentuan ukuran sepatu pada umumnya menggunakan *Brannock* supaya pas dan hasil sepatu tidak terlalu kecil atau terlalu besar.Setiap jenis sepatu memiliki bagian-bagian sepatu yang kadang berbeda dari sepatu jenis lainnya. Untuk material atau bahan utama sepatu, terdapat beberapa bahan diantaranya kulit, kulit sintetis, kulit lak, *suede*, kulit buk, *canvas*, *nylon*, bludru dan denim.

Fungsi dan kegunaan sepatu adalah untuk melindungi kaki dari benda tajam dan batu krikil dijalan. Sepatu Juga biasa digunakan untuk keperluan Fashion, Sekolah, Kerja, Olahraga, Pesta dan sebagainya.

2.3 *Laundry*

Laundry merupakan suatu tempat yang menyediakan jasa untuk pencucian pakaian seperti baju , kaos , kemeja, celana. Tidak hanya itu saja, sekarang sudah ada jasa yang menawarkan pembersihan untuk jenis lain yang dapat dicuci. Seperti sepatu, ada cukup bayak tempat *laundry* sepatu yang menyediakan pembersihan dan perawatan untuk sepatu.

2.4 Definisi Warna

Berikut ini adalah definisi warna menurut para ahli :

a) Teori Sir Isacc Newton

Warna adalah spectrum tertentu yang terdapat dalam suatu cahaya sempurna berwarna putih (Nugroho, 2007).

b) Albert H. Munsell

Warna merupakan elemen penting dalam semua lingkup disiplin seni rupa, bahkan secara umum warna merupakan bagian penting dari berbagai aspek kehidupan manusia (Prayoga, 2015).

c) Albert Stuart

Dalam seni rupa, warna bisa berarti pantulan tertentu dari cahaya yang dipengaruhi oleh pigmen yang terdapat dipermukaan benda (Aziz, 2012).

d) J.Linschoten dan Drs. Mansyur

Warna menurut psikologis, warna-warna itu bukanlah suatu gejala yang hanya dapat diamati saja, warna itu juga mempengaruhi kelakuan, memegang peran penting dalam penilaian estetis dan turut menentukan suka tidaknya kita akan bermacam-macam benda. Dari pemahaman diatas dapat dijelaskan bahwa warana selain hanya dapat dilihat dengan mata ternyata mampu mempengaruhi prilaku seseorang, mempengaruhi penilaian estetis dan turut menentukan suka tidaknya manusia terhadap suatu benda (Anugrah, 2012).

2.5 Model Warna *RGB*

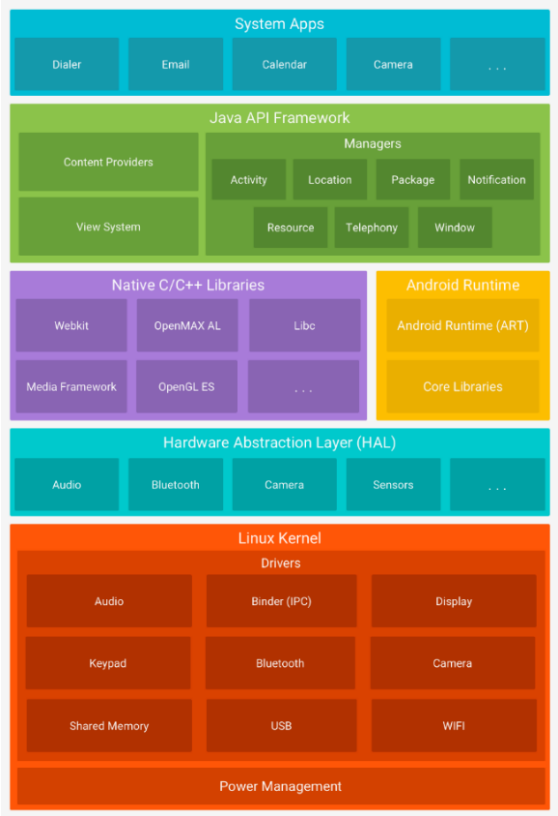
Model warna RGB adalah model warna berdasarkan konsep penambahan kuat cahaya primer yaitu Red, Green dan Blue. Dalam suatu ruang yang sama sekali tidak ada cahaya, maka ruangan tersebut adalah gelap total. Tidak ada signal gelombang cahaya yang diserap oleh mata kita atau RGB (0,0,0). Apabila kita menambahkan cahaya merah pada ruangan tersebut, maka ruangan akan berubah warna menjadi merah misalnya RGB (255,0,0), semua benda dalam ruangan tersebut hanya dapat terlihat berwarna merah. Demikian apabila cahaya kita ganti dengan hijau atau biru. [1]

2.6 *Android*

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis *linux* yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka.

Awalnya, *Google Inc*. membeli *Android Inc*. yang merupakan pendatang baru yang membuat piranti lunak untuk ponsel pintar atau biasa disebut *Smartphone*. Kemudian untuk mengembangkan *Android*, dibentuklah *Open Handset Alliance*, *konsorsium* dari 34 perusahaan piranti keras, piranti lunak, dan telekomunikasi, termasuk *Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia* [2].

Sistem Operasi *Android* memiliki beberapa komponen utama yang disebut dengan Arsitektur *Platform* Android. Berikut adalah diagram komponen-komponen utama dari *platform Android* :



Gambar 2. 1 Arsitektur Android

Berikut adalah penjelasan dari setiap komponen utama arsitektur pada *platform android* pada Gambar 2.1 :

1. Linux Kernel

Fondasi *platform Android* adalah kernel Linux. Sebagai contoh, *Android Runtime (ART)* bergantung pada *kernel Linux* untuk fungsionalitas dasar seperti *threading* dan manajemen memori tingkat rendah. Menggunakan *kernel Linux* memungkinkan Android untuk memanfaatkan fitur keamanan inti dan memungkinkan produsen perangkat untuk mengembangkan *driver* perangkat keras untuk *kernel* yang cukup dikenal.

2. Hardware Abstraction Layer

Hardware Abstraction Layer (HAL) menyediakan antarmuka standar yang mengekspos kemampuan perangkat keras di perangkat ke kerangka kerja *Java API* yang lebih tinggi. *HAL* terdiri atas beberapa modul pustaka, masing-masing mengimplementasikan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth. Bila *API* kerangka kerja melakukan panggilan untuk mengakses perangkat keras, sistem Android memuat modul pustaka untuk komponen perangkat keras tersebut.

3. Android Runtime

Perangkat Android yang menjalankan *Android* versi 5.0 (*API* level 21) atau yang lebih tinggi, setiap aplikasi menjalankan proses masing-masing dengan tahap *Android Runtime (ART)*. *ART* ditulis guna menjalankan beberapa mesin *virtual* pada perangkat bermemori rendah dengan mengeksekusi file *DEX*, format *bytecode* yang didesain khusus untuk Android yang dioptimalkan untuk *footprint* memori minimal. Beberapa *fitur* utama *ART* mencakup:

a) Kompilasi mendahului waktu (*AOT*) dan tepat waktu (*JIT*)
b) Pengumpulan sampah (*GC*) yang dioptimalkan
c) Dukungan *debug* yang lebih baik, mencakup *profiler sampling* terpisah, pengecualian diagnostik mendetail dan laporan kerusakan dan kemampuan untuk mengatur titik pantau guna memantau bidang tertentu.

Sebelum ke *Android* versi 5.0 (*API* level 21), *Dalvik* adalah waktu proses *Android*. Jika aplikasi berjalan baik pada *ART*, semestinya berfungsi baik juga pada *Dalvik*, tetapi mungkin tidak sebaliknya.

Android juga menyertakan serangkaian pustaka waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman Java, termasuk beberapa fitur bahasa *Java 8*, yang digunakan kerangka kerja *Java API*.

4. Native C/C++ Libraries

Banyak komponen dan layanan sistem Android inti seperti *ART* dan *HAL* dibuat dari kode asli yang memerlukan pustaka asli yang tertulis dalam C dan C++. *Platform Android* memungkinkan kerangka kerja *Java API* mengekspos fungsionalitas beberapa pustaka asli pada aplikasi. Misalnya, Anda bisa mengakses *OpenGL ES* melalui kerangka kerja *Java OpenGL API* Android guna menambahkan dukungan untuk menggambar dan memanipulasi grafik 2D dan 3D pada aplikasi Anda.

5. Native C/C++ Libraries

Banyak komponen dan layanan sistem Android inti seperti *ART* dan *HAL* dibuat dari kode asli yang memerlukan pustaka asli yang tertulis dalam C dan C++. *Platform Android* memungkinkan kerangka kerja *Java API* mengekspos fungsionalitas beberapa pustaka asli pada aplikasi. Misalnya, Anda bisa mengakses *OpenGL ES* melalui kerangka kerja *Java OpenGL API* Android guna menambahkan dukungan untuk menggambar dan memanipulasi grafik 2D dan 3D pada aplikasi Anda.

6. Java API Framework

Keseluruhan rangkaian fitur pada Android OS tersedia untuk Anda melalui *API* yang ditulis dalam bahasa *Java*. *API* ini membentuk elemen dasar yang Anda perlukan untuk membuat aplikasi Android dengan menyederhanakan penggunaan kembali inti, komponen dan layanan sistem modular, yang menyertakan berikut ini:

a) Tampilan Sistem yang kaya dan luas bisa Anda gunakan untuk membuat UI aplikasi, termasuk daftar, kisi, kotak teks, tombol, dan bahkan browser web yang dapat disematkan
b) Pengelola Sumber Daya, memberikan akses ke sumber daya bukan kode seperti string yang dilokalkan, grafik, dan file layout

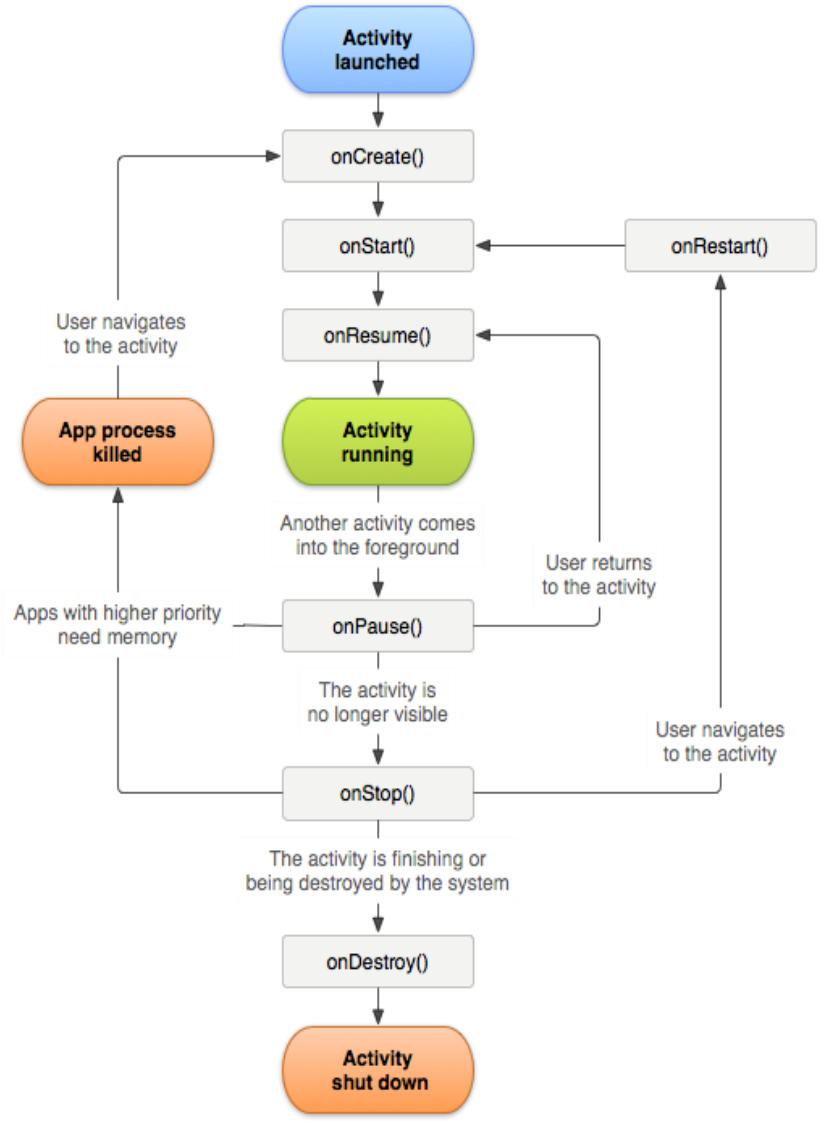
c) Pengelola Notifikasi yang mengaktifkan semua aplikasi guna menampilkan lansiran khusus pada bilah status
d) Pengelola Aktivitas yang mengelola daur hidup aplikasi dan memberikan back-stack navigasi yang umum
e) Penyedia Materi yang memungkinkan aplikasi mengakses data dari aplikasi lainnya, seperti aplikasi Kontak, atau untuk berbagi data milik sendiri.

7. System Apps

Android dilengkapi dengan serangkaian aplikasi inti untuk *email*, perpesanan, kalender, menjelajahi *internet*, kontak, dll. Aplikasi yang disertakan bersama *platform* tidak memiliki status khusus pada aplikasi yang ingin dipasang pengguna. Jadi, aplikasi pihak ketiga dapat menjadi *browser web* utama, pengolah pesan *SMS* atau bahkan *keyboard* utama (beberapa pengecualian berlaku, seperti aplikasi *Setting sistem*).

Aplikasi sistem berfungsi sebagai aplikasi untuk pengguna dan memberikan kemampuan kunci yang dapat diakses oleh *developer* dari aplikasi mereka sendiri. Misalnya, jika aplikasi Anda ingin mengirimkan pesan *SMS*, Anda tidak perlu membangun fungsionalitas tersebut sendiri-sebagai gantinya Anda bisa menjalankan aplikasi *SMS* mana saja yang telah dipasang guna mengirimkan pesan kepada penerima yang Anda tetapkan.

Di dalam Sistem operasi *Android* terdapat Siklus hidup dimana pada siklus ini untuk menavigasi antara tahap *Activity Life-Cycle, Android* itu sendiri dengan menyediakan 6 *method* inti *callback* yaitu *onCreate(), onStart(), onResume(), onPause(), onStop(), and onDestroy()*. Dalam siklus hidup *method callback*, kita dapat mendeklarasikan cara perilaku *activity* saat pengguna meninggalkan dan memasuki kembali ke sebuah *activity* itu. Disitulah peranan *method callback* dalam sebuah aplikasi pada platform Android . Berikut adalah Activity Life-Cycle pada platform Android:



Gambar 2. 2 Sistem Apps

Keenam *method callback* tersebut memiliki perananya masing-masing dan dijelaskan secara rinci sebagai berikut :

a) *OnCreate*()

Method ini adalah *method* utama dari setiap *Activity. Method* ini akan dipanggil pertama kali ketika menjalankan sebuah sistem. Pengembang harus mengimplementasikan metode *onCreate()* untuk menjalankan logika memulai aplikasi dasar yang hanya boleh terjadi satu kali selama hidup aktivitas. Misalnya, implementasi *onCreate()* Anda

harus mendefinisikan antarmuka pengguna dan mungkin membuat instance beberapa variabel dalam cakupan-kelas.

b) *OnStart*()

Method ini dipanggil ketika method *onCreate()* telah dipanggil. *onStart()* dipanggil ketika terlihat oleh *user*. *Method* ini selesai dengan cepat dan dilanjutkan dengan method setelahnya yaitu *onResume().*

c) *OnResume*()

Method ini dipanggil ketika method *onStart()* selesai dipanggil. Method ini adalah keadaan dimana pengguna berinteraksi dengan aplikasi. Aplikasi akan tetap dalam keadaan ini sampai terjadi suatu statement dari aplikasi semisal menerima panggilan telepon atau mematikan layar *smartphone.*

d) *OnPause*()

Method ini dipanggil ketika pengguna meninggalkan *activity (*meskipun tidak selalu berarti activity dihancurkan). *Method* ini berguna untuk menghentikan sementara operasi yang sedang berjalan semisal menjeda pemutaran musik dan lain-lain.

e) *OnStop*()

Method ini dipanggil ketika *activity* tidak terlihat lagi oleh pengguna, dengan kata lain *activity* berhenti dijalankan. Hal ini dapat terjadi semisal ada aktivitas baru dijalankan meliputi seluruh layar. Sistem juga dapat menghubungi *method* ini ketika *activity* selesai berjalan, dan akan segera dihentikan.

f) *OnDestroy*()

Method ini adalah *method callback* ketika *activity* telah selesai dijalankan dan kemudian memanggil *method finish()* atau karena sistem untuk sementara menghancurkan proses yang berisi *activity* tersebut untuk menghemat ruang memori.

2.7 Android Studio

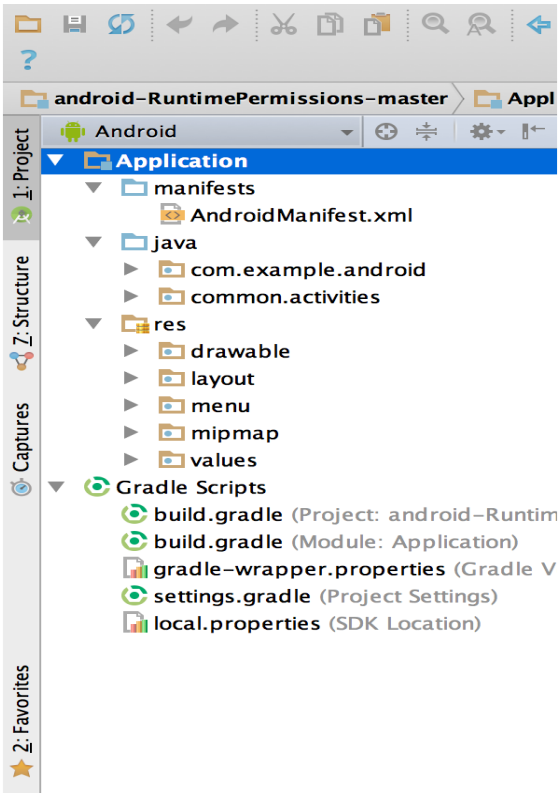
Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment (IDE)* untuk pengembangan aplikasi *Android* berdasarkan *IntelliJ IDEA* . Selain merupakan *editor* kode *IntelliJ* dan alat pengembang yang berdaya guna, *Android Studio* menawarkan *fitur* lebih banyak untuk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

a) Sistem pembuatan berbasis *Gradle* yang fleksibel.
b) Emulator yang cepat dan kaya *fitur.*
c) Lingkungan yang menyatu untuk pengembangan bagi semua perangkat *Android.*
d) *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat *APK* baru.
e) *Template* kode dan integrasi *GitHub* untuk membuat *fitur* aplikasi yang sama dan mengimpor kode contoh.
f) Alat penguji dan kerangka kerja yang *ekstensif.*
g) Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain.
h) Dukungan *C++* dan *NDK*
i) Dukungan bawaan untuk *Google Cloud Platform*, mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*

Android Studio mempunyai struktur *project* yang berisi satu atau beberapa modul dengan file kode sumber dan *file* sumber daya. Jenis-jenis modul mencakup:

a) Modul aplikasi *Android*.
b) Modul perpustakaan.
c) Modul *Google App Engine.*

Pada tampilan IDE Android studio, struktur project dapat terlihat pada bagian kiri IDE seperti Gambar berikut :



Gambar 2. 3 *Struktur Project*

Secara *default*, *Android Studio* menampilkan *file* proyek Anda dalam tampilan proyek *Android* seperti yang ditunjukkan dalam Gambar 2.4.

```
<?php
 $keyword =isset($_GET['keyword']) ? urlencode($_GET['keyword']) : '';
 $lat = isset($_GET['lat']) ? $_GET['lat'] : '';
 $lng = isset($_GET['lng']) ? $_GET['lng'] : '';
$pagetoken = isset($_GET['pagetoken']) ? $_GET['pagetoken'] : '';
 if (($lat == "" || $lng == "") && $pagetoken == "")
 { die("Tentukan dulu kooordinat lokasi di <a href='places.php'>link berikut ini</a>");
 }
?>
<html>
<head>
<title>Pencarian detail lokasi</title>
</head>
<body>
<h2>Pencarian detail lokasi</h2>
<form name='Tanya' method='GET'>
Apa yang anda cari? (restaurant, bank, sekolah, dsb):
<input id="keyword" name='keyword' type='text' value='<?php echo $keyword ?>'>
<input type="hidden" name="lat" value="<?php echo $lat ?>">
<input type="hidden" name="lng" value="<?php echo $lng ?>">
<input type="submit" name="cari" value="Search">
</form>
<?php
  $url = "";
  $api_key="=== MASUKKAN API KEY GOOGLE DI SINI ===";
  if ($pagetoken != "")  // jika bukan halaman pertama
  {
$url="https://maps.googleapis.com/maps/api/place/textsearch/json?pagetoken=$pagetoken&sensor=
false&key=$api_key&";
  }
  else if ($pagetoken == "" && $keyword != "")  // halaman pertama
  {
$url="https://maps.googleapis.com/maps/api/place/textsearch/json?query=$keyword&sensor=false&
key=$api_key&location=$lat,$lng&radius=100";
  }
  if ($url != "")   // jika ada url yg akan dicari
  {  $result_json = json_decode(file_get_contents($url));
    echo '<div id="HasilPencarian">';
if ($result_json->status == 'OK')
{ for ($i=0; $i<sizeof($result_json->results); $i++)
                                    { $name = $result_json->results[$i]->name;
$address = $result_json->results[$i]->formatted_address;
    echo "$name, $address<br/>";
  }
  echo "<p><a href='javascript:back()'>Previous page</a>";
  if (isset($result_json->next_page_token))
  { $next_page = $result_json->next_page_token;
    echo "<p><a href='?pagetoken=$next_page'>Next page</a>";
  }
```

Gambar 2. 4 *File Project*

Tampilan ini diatur menurut modul untuk memberi akses cepat ke file sumber kunci project. Semua file versi terlihat di bagian atas di bawah *Gradle Scripts* dan masing-masing modul aplikasi berisi folder berikut:

a) Manifests : Berisi file *AndroidManifest.xml*.

b) Java : Berisi file kode sumber *Java*, termasuk kode pengujian *JUnit*.

c) Res : Berisi semua sumber daya bukan kode, seperti tata letak *XML*, string UI, dan gambar *bitmap*.

2.8 API Clarifai

Pada aplikasi ini mengunakan *API Clarifai* dimana pengertian *API Clarifai* adalah sebagai berikut :

2.8.1 *API (Application Programming Interface)*

API adalah sekumpulan perintah, fungsi dan *protocol* yang dapat digunakan saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan *programmer* untuk menggunakan fungsi *standar* untuk berinter aksi dengan sistem operasi. *API* atau *Application Programming Interface* juga merupakan satu dokumentasi yang terdiri dari antar muka.kelas struktur untuk membangun sebuah perangkat lunak.

Dengan adaya API, maka memudahkan seorang *programmer* untuk membongkar suatu *software* untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. *API* dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainya. Suatu rutin *standar* yang memungkinkan *developer* menggunakan *system function*. proses ini dikelola melalui *operating system*. keunggulan lain dari *API* ialah memungkinkan suatu aplikasi dengan aplikasi lain dapat saling berinteraksi.

2.8.2 *Clarifai*

Clarifai adalah alat pengenalan gambar dan video yang secara otomatis memberikan tag kepada objek dan kategori yang hanya mengambil *pixel* sebagai input. Sistem ini didasarkan kepada jaringan saraf. Teknik pembelajaran mesin *scalable* yang dpat menagani skala besar konten visual yang mengalir melalui *API*. *Clarifai* juga menggunakan kesamaan semantik dan *visual* untuk membandingkan gambar yang diunggah dengan gambar lainnya di *library* mereka untuk menampilkan kesamaan. *API* digunakan diaplikasi ini untuk menentukan tingkat

kekotoran yang terdapat pada sepatu, sehingga dapat diketahui seberapa tinggi tingkat kekotorannya [3].

2.9 JAVA

Java adalah salah satu bahasa pemrograman yang paling populer. *Java* dapat digunakan untuk berbagai hal, termasuk pengembangan perangkat lunak, aplikasi *mobile*, dan pengembangan sistem yang besar. Inilah yang membuat bahasa pemrograman *Java* sangat terkenal di lingkungan pengembang perangkat lunak [6]. Seperti bahasa pemrograman lain, bahasa *Java* memiliki struktur sendiri, aturan sintaks, dan paradigma pemrograman. paradigma pemrograman bahasa Java didasarkan pada konsep *OOP*. Bahasa *Java* merupakan turunan bahasa C , sehingga aturan *sintaks* yang terlihat akan seperti bahasa C. Misalnya, blok kode yang modular dalam metode dan dibatasi oleh karakter '({' dan '})', dan variabel dideklarasikan sebelum digunakan. Secara struktural, bahasa *Java* diatur dengan *package*. Di Dalam package ada *class*, dan dalam *class* ada *method*, variabel, konstanta, dan banyak lagi.

```
Publicclassstring{public static void main(String[]args)
{
Scanner MyScanner=new Scanner(System.in);
String[]buah=new String[100];
String temp;
int banyak;
System.out.print("masukkan berapa banyak buah yang mau dimasukkan =");
banyak=MyScanner.nextInt();
for(int i=0;i<banyak;i++)
{
System.out.print("Buah ke "+(i+1)+" =");
```

Gambar 2. 5 *Create Database*

2.10 PHP

PHP adalah singkatan dari PHP: *Hypertext PreProcessors*, *PHP* saat ini adalah bahasa pemrograman interpreter yang paling banyak digunakan saat ini dikarenakan bersifat *open source* dan juga paling banyak didukung oleh banyak *web server*. PHP (Hypertext Preprocessor), merupakan bahasa pemrograman pada sisi server yang memperbolehkan *programmer* menyisipkan perintah – perintah perangkat lunak *web server* (*Apache, IIS,* atau apapun) akan dieksekusi sebelum perintah itu dikirim oleh halaman ke *browser* yang me-request-nya.

Sebagai bahasa pemrograman untuk tujuan umum, kode *PHP* diproses oleh aplikasi penerjemah dalam modus baris-baris perintah modus dan melakukan operasi yang diinginkan sesuai sistem operasi untuk menghasilkan keluaran program dichannel output standar. Hal ini juga dapat berfungsi sebagai aplikasi grafis. *PHP* tersedia sebagai prosesor untuk *server web* yang paling modern dan sebagai penerjemah mandiri pada sebagian besar sistem operasi dan komputer *platform*.

2.11 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data *SQL* (*Database Management System*) atau *DBMS* yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. *MySQL AB* (perusahaan asal Finlandia) membuat *MySQL* sebagai perangkat lunak gratis dibawah lisensi *GNU General Public License (GPL)*, tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan *GPL*[8].

Relational Database Management System (RDBMS). *MySQL* merupakan *Relational Database Management System (RDBMS)* yang didistribusikan secara gratis dibawah lisensi *GPL (General Public License)* dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu *SQL (Structured Query Language)*.

Dalam penggunaan database MySQL, setiap perintah yang diketikkan disebut ***query***. Perintah MySQL dapat dikategorikan menjadi 3 sub perintah, yaitu *DDL*

(Data Definition Language), *DML (Data Manipulation Language)*, dan *DCL (Data Control Language)*.

1. *DDL* (*Data Definition Language*)

Perintah dalam *SQL* yang pertama adalah perintah *DDL*. *DDL* sendiri mrupakan kependekan dari apa yang dikenal dengan nama Data Definition Language. Apabila diartikan secara harfiah, Data Definition Language berarti merupakan sebuah bahasa *SQL* yang digunakan untuk mendefinisikan suatu data. Secara teoritis, *DDL* dapat berarti sebuah perintah yang berhubungan dengan pendefinisian dari suatu struktur database. Terdapat beberapa perintah *DDL* pada MySQL sebagai berikut :

a) *CREATE* berfungsi untuk membuat database baru, tabel baru, view baru dan kolom.
b) *ALTER* berfungsi untuk mengubah struktur tabel. Seperti mengganti nama tabel, menambah kolom, mengubah kolom, menghapus kolom maupun memberikan atribut pada kolom.
c) *DROP* berfungsi untuk menghapus database dan tabel.
d) *TRUNCATE* berfungsi untuk Menghapus semua catatan dari tabel.
e) *COMMENT* berfungsi untuk Menambahkan komentar pada data.
f) *RENAME* berfungsi untuk mengubah nama obyek.

Adapun contoh sintaks dari perintah DDL pada MySQL sebagai berikut :

```
CREATE DATABASE NILAI;
ALTER TABLE Mahasiswa ADD (NoTelp char(8));
DROP INDEX nama_index    ;
TRUNCATE TABLE table_barang;
RENAME table_barang to table_barang_gudang;
```

Gambar 2. 6 *Create database*

2. *DML* (*Data Manipulation Language*)

Data Manipulation Language (DML) ialah perintah yang digunakan untuk mengelola/memanipulasi data dalam database. Terdapat beberapa perintah *DML* pada MySQL sebagai berikut :

a) *SELECT* berfungsi untuk mengambil/menampilkan data dari database.
b) *INSERT* berfungsi untuk memasukkan data ke dalam tabel.
c) *UPDATE* berfungsi untuk memperbarui data dalam tabel.
d) *DELETE* berfungsi untuk menghapus data dari tabel.
e) *CALL* berfungsi untuk memanggil subprogram PL / SQL atau Java
f) *EXPLAIN PLAN* berfungsi untuk menjelaskan jalur akses ke data
g) *LOCK TABLE* berfungsi untuk mengunci tabel.

Adapun contoh sintaks dari perintah DML pada MySQL sebagai berikut :

```
INSERT INTO mahasiswa VALUES("08052926",
"Frenky","70");
SELECT nama_mahasiswa FROM mahasiswa;
DELETE FROM mahasiswa;
UPDATE mahasiswa SET nama mahasiswa='ujang'
```

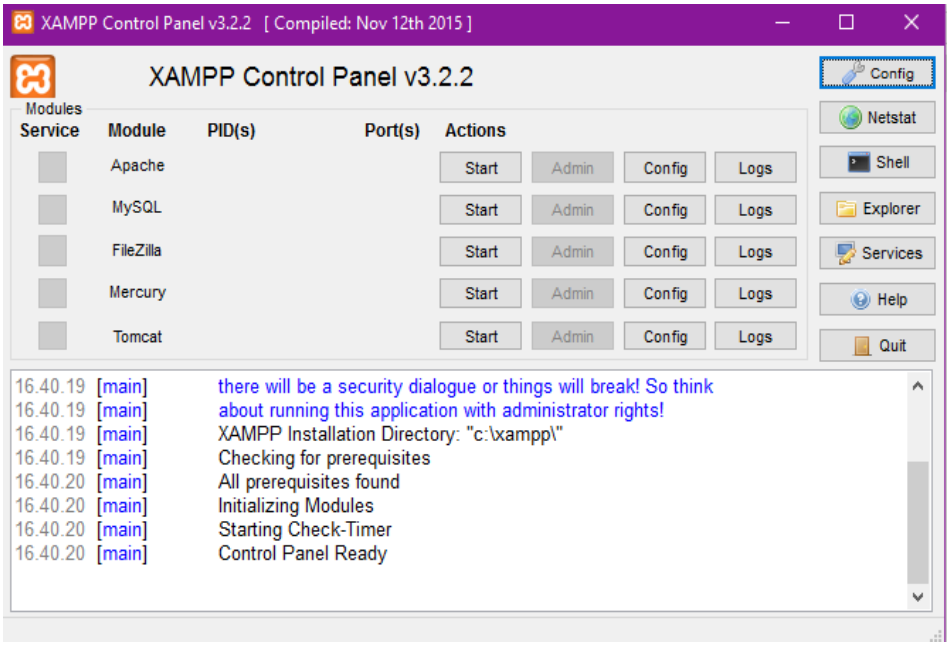
Gambar 2. 7 *Insert Database*

2.12 XAMPP

XAMPP adalah sebuah *software* web server apache yang didalamnya sudah tersedia database server *MySQL* dan dapat mendukung pemrograman *PHP*. *XAMPP* merupakan software yang mudah digunakan, gratis dan mendukung instalasi di *Linux* dan *Windows*. Keuntungan lainnya adalah cuma menginstal satu kali sudah tersedia Apache *Web Server*, *MySQL Database Server*, *PHP Support* (*PHP* 4 dan *PHP* 5) dan beberapa *module* lainnya.

Fungsi *XAMPP* sendiri adalah sebagai server yang berdiri sendiri (*localhost*), yang terdiri beberapa program antara lain : *Apache HTTP Server*, *MySQL database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman *PHP* dan Perl. Nama *XAMPP* sendiri merupakan singkatan dari X (empat sistem operasi apapun), *Apache, MySQL, PHP* dan *Perl*. Program ini

tersedia dalam *GNU (General Public License)* dan bebas, merupakan web server yang mudah untuk digunakan yang dapat menampilkan halaman web yang dinamis.



Gambar 2. 8 *Xampp*

2.13 Web Server

Web Server adalah sebuah perangkat lunak yang bertugas menerima permintaan client melalui port *HTTP* maupun *HTTPS* dan merubahnya ke dalam format *HTML*. Terdapat beberapa format selain *HTML* yaitu *PHP* atau *ASP*, tetapi format–format tersebut hanyalah berfungsi untuk menghubungkan *HTML* dengan database[10].

Web server yang umum digunakan adalah *Apache.* Tugas utama dari *Web server* adalah menerjemahkan permintaan ke dalam respon yang cocok untuk keadaan pada saat itu, ketika klien membuka komunikasi dengan *Apache, Apache* mengirimkan permintaan untuk sumber daya. *Apache* menyediakan sumber daya yang baik atau memberikan respon alternatif untuk menjelaskan mengapa permintaan tidak dapat terpenuhi. Dalam banyak kasus, sumber daya adalah *Hypertext Markup Language (HTML)* halaman *web* yang berada pada *disk* lokal, tetapi ini hanya pilihan sederhana. Hal ini dapat berupa *file* gambar, hasil dari sebuah *script* yang menghasilkan *output HTML*, applet *Java* yang diunduh dan

dijalankan oleh klien, dan seterusnya, *Apache* menggunakan *HTTP* untuk berbicara dengan klien. Permintaan / tanggapan protokol ini, yang berarti bahwa ia mendefinisikan bagaimana membuat permintaan klien dan bagaimana *server* menanggapi mereka. Setiap komunikasi *HTTP* dimulai dengan permintaan dan berakhir dengan jawaban. *Executable Apache* mengambil nama dari protokol, dan pada sistem *Unix* umumnya disebut *httpd*, kependekan *daemon HTTP*.

2.14 *OOP* (*Object Oriented Programming*)

OOP adalah suatu cara untuk mengorganisasi perangkat lunak sebagai kumpulan dari objek tertentu yang memiliki struktur data dan perilakunya. Setiap objek mempunyai sifat yang melekat pada identitasnya. Pendekatan berorientasi objek merupakan paradigma baru dalam rekayasa perangkat lunak yang memandang sistem sebagai kumpulan objek-objek diskrit yang saling berinteraksi. Yang dimaksud beriorientasi objek adalah bahwa mengorganisasikan perangkat lunak sebagai kumpulan objek-objek diksrit yang berkerja sama antara informasi atau struktur data dan perilaku yang mengaturnya.

2.14.1 Objek (*Object*)

Objek adalah abstraksi dan sesuatu yang mewakili dunia nyata seperti benda manusia, satuan organisasi, tempat, kejadian, struktur, status, atau hal-hal lain yang bersifat abstrak. Objek juga bisa merupakan suatu entitas yang mampu menyimpan informasi (status) dan mempunyai operasi (kelakuan) yang dapat diterapkan atau dapat berpengaruh pada status objeknya. Objek mempunyai siklus hidup yaitu diciptakan, dimanipulasi, dan dihancurkan.

Semua objek mempunyai identitas yang berbeda dengan lainnya. Misalnya saja dua apel mempunyai warna, permukaan dan tekstur yang sama tapi tetap saja dipandang sebagai individual apel.

2.14.2 Kelas(*Class*)

Kelas adalah kumpulan objek-objek dengan karakteristik yang sama. Kelas merupakan definisi statik dan himpunan objek yang sama yang mungkin lahir atau

diciptakan. Sebuah kelas akan mempunyai sifat (atribut), kelakuan (operasi/metode), hubungan (relationship) dan arti.

2.14.3 Metode (*Method*)

Metode (*method*) disebut juga service atau operator adalah prosedur atau fungsi seperti yang terdapat dalam bahasa pascal, tetapi cara kerjanya agak berlainan. Metode adalah subprogram yang tergabung dalam objek bersama-sama dengan atribut.

Sebuah kelas boleh memiliki lebih dari satu metode atau operasi yang berfungsi untuk memanipulasi objek itu sendiri. Metode dapat berasal dari event, aktivitas atau aksi keadaan, fungsi, atau kelakuan dunia nyata. Contoh metode atau operasi misalnya Read, Write, Move, Copy, dan sebagainya.

2.14.4 Atribut

Atribut adalah variable global yang dimiliki sebuah kelas. Atribut dapat berupa nilai atau elemen-elemen data yang dimiliki oleh objek dalam kelas objek. Atribut berarti properti, kualitas dan karakteristik yang dapat ditentukan pada orang atau barang dan data dimana objek atau kelas memiliki sebuah nilai.

2.14.5 Abstraksi

Abstraksi bisa juga prinsip untuk merepresentasikan dunia nyata yang kompleks menjadi satu bentuk model yang sederhana dengan mengabaikan aspek-aspek lain yang tidak sesuai dengan permasalahan. Tujuan abstraksi adalah untuk melakukan isolasi aspek-aspek penting demi suatu maksud dan menindak aspek-aspek yang tidak penting. Abstraksi harus selalu mempunyai maksud, karena maksud yang menentukan apa yang penting dan apa yang tidak penting. Model yang bagus adalah model yang dapat menagkap aspek-aspek paling penting dari masalah dan menghilangkan aspek-aspek lain yang tidak penting.

2.14.6 Enkapsulasi (*Encapsulation*)

Pembungkusan atribut data dan layanan (operasi-operasi) yang dipunyai objek untuk menyembunyikan implementasi dan objek sehingga objek lain tidak mengetahui cara kerjanya. Enkapsulasi merupakan daar untuk pembatasan ruang lingkup program terhadap data yang diproses. Data dan prosedur atau fungsi dikemas dalam bersama-sama dalam suatu objek, sehingga prosedur atau fungsi lain dari luar tidak dapat mengaksesnya.

2.14.7 Polimorfisme (*Polymorphism*)

Polimorfisme yaitu konsep yang menyatakan bahwa sesuatu yang sama dapat mempunyi bentuk dan perilaku berbeda. Polimorfisme mempunyai arti bahwa operasi yang sama mungkin mempunyai perbedaan dalam kelas yang berbeda. Polimorfisme berarti suatu objek untuk digunakan di banyak tujuan yang berbeda dengan nama yang sama sehingga menghemat baris program.

2.14.8 Pewarisan (*Inheritance*)

Pewarisan adalah teknik yang menyatakan bahwa anak dari objek akan mewarisi data/atribut dan metode dari induknya langsung. Atribut dan metode dari objek induk diturunkan kepada anak objek, demikian seterusnya. Pewarisan merupakan mekanisme yang memungkinkan satu objek mewarisi sebagaian atau seluruh definisi dan objek lain sebagai bagian dan dirinya.

2.15 *UML* (*Unified Modelling Language*)

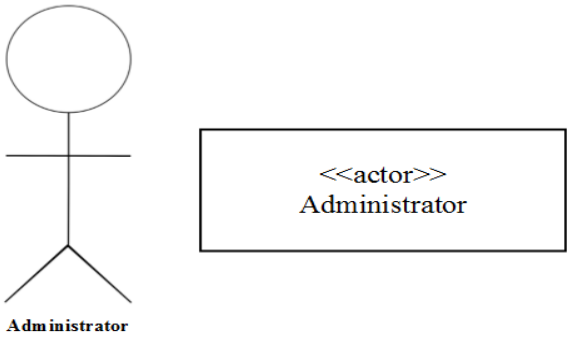
UML adalah bahasa grafis untuk mendokumentasi, menspesifikan, dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak level abstaksi, tidak bergantung proses pengenbangan, tidak bergantung bahasa teknologi. Pemaduan beberapa di beragam metodologi, usaha bersama dari banyak pihak, didukung oleh kakas-kakas yang diintegrasikan lewat XML (XMI). Standar UML dikelola oleh OMG (*Object Management Group*).

UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek UML menyediakan berbagai diagram diantaranya sebagai berikut:

2.15.1 Diagram *Use Case*

Diagram use case merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. Use case mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, use case digunakan untuk mengetahui fungsi apa saja yang berhak menggunakan fungsi-fungsi itu. Ada dua hal yang utama pada use case yaitu pendefinisian apa yang disebut aktor dan use case.

a) Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambaran orang, tapi aktor belum tentu merupakan orang.

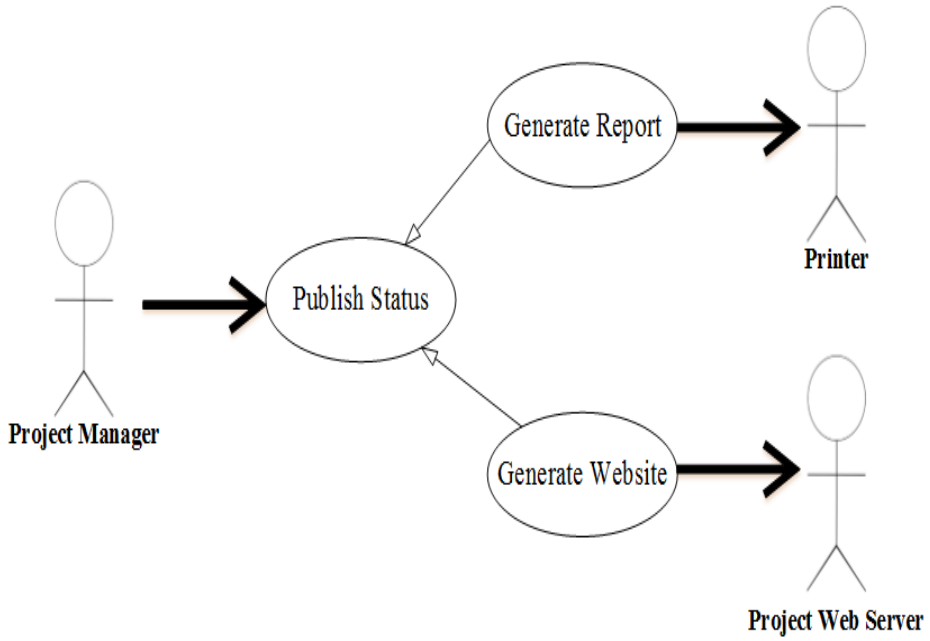


Gambar 2. 9 *Actor*

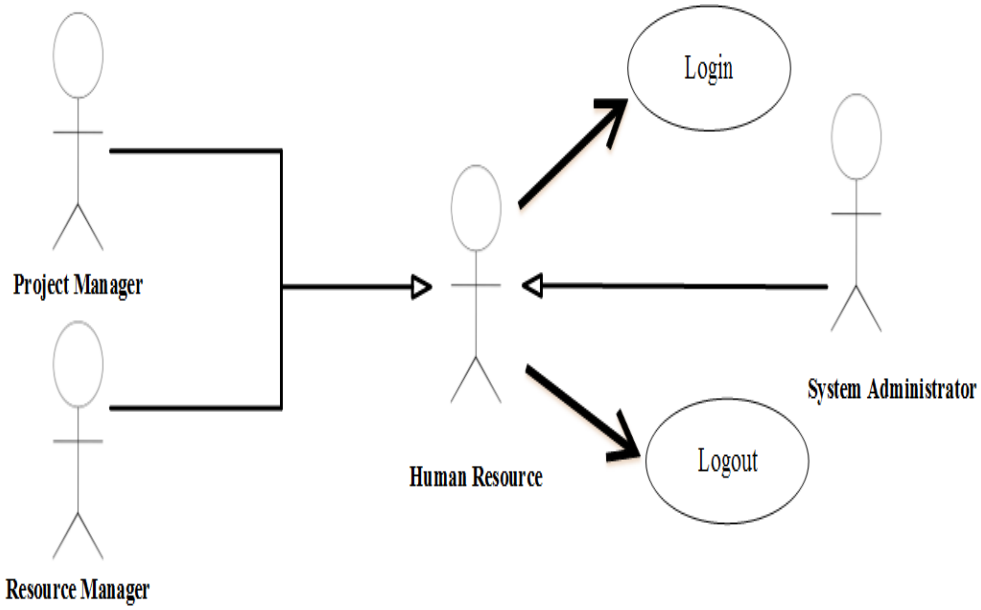
b) Use case merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Gambar 2. 10 *User*

Adapun relasi yang terdapat pada *use diagram* yaitu :



Gambar 2. 11 Merupakan spesialisasi dari elemen lainnya



Gambar 2. 12 *Generalisasi Use Case*

a) *Association*, menghubungkan link antar element
b) *Depedency*, sebuah element bergantung dalam beberapa cara ke element lainnya.

c) *Aggregation*, bentuk *association* dimana sebuah elemen berisi elemen lainnya

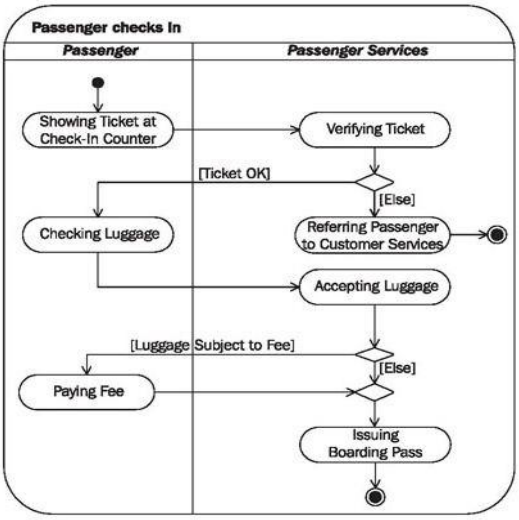
Tipe relasi yang mungkin terjadi pada *use case* diagram :

a) <<include>>, yaitu relasi *use case* terhadap *use case* lainnya, dimana *use case* yang di tambahkan memerlukan *use case* ini untuk menjalankannya.
b) <<extends>>, yaitu relasi *use case* tambahan ke sebuah *use case*, dimana *use case* yang di tambahkan dapat berdiri sendiri.

2.15.2 Diagram Aktivitas

Diagram aktivitas adalah suatu aktivitas dari sebuah sistem atu proses bisnis atau menu yang ada pada perangkat lunak. Diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, tapi aktivitas yang dapat dilakukan oleh sistem. Diagram aktivitas banyak digunakan untuk mendefinisikan hal-hal berikut :

a) Rancangan menu yang ditampilkan pada perangkat lunak
b) Setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

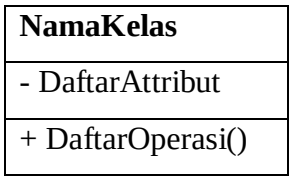


Gambar 2. 13 Activity Diagram

2.15.3 Diagram Kelas

Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Dalam UML, *class* digambarkan dengan segi empat yang dibagi beberapa bagian seperti pada Gambar 2.9



Gambar 2. 14 *Class Diagram*

Atribut dan operasi dapat memiliki salah satu sifat yaitu :

a) *Private*, sifat ini tidak dapat dipanggil dari luar *class* kecuali dengan *class* yang bersangkutan.
b) *Protected*, sifat ini hanya bisa dipanggil oleh *class* yang bersangkutan dan class yang mewarisinya.
c) *Public*, sifat ini dapat dipanggil oleh semua *class*.

2.15.4 Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek.

Diagram sekuen menunjukkan objek sebagai garis vertikal dan tiap kejadian sebagai panah horisontal dari objek pengirim ke objek penerima. Waktu berlalu dari atas ke bawah dengan lama waktu tidak relevan. Diagram ini hanya menunjukkan barisan kejadian, bukan pewaktuan nyata. Kecuali untuk sistem waktu nyata yang mengharuskan konstrain barisan kejadian.