BAB 2
TINJAUAN PUSTAKA

2.1. Landasan Teori

Landasan teori merupakan penjelasan berbagai konsep dasar dan teori-teori yang berkaitan dalam pembangunan aplikasi *mobile* android untuk memberi rekomendasi wisata di Kota Bandung. Teori-teori yang terkait serta mendukung penelitian ini akan dibahas pada bab ini.

2.2. Kota Bandung

Secara geografis, kota Bandung terletak di tengah-tengah provinsi jawa barat, serta berada pada ketinggian kurang lebih 768 meter di atas permukaan laut. Kota Bandung memiliki luas wilayah 16.731 hektar, yang secara administrative terbagi atas 30 kecamatan, 151 kelurahan, 1.561 rw, dan 9.691 rt. Jumlah penduduk kota bandung tahun 2012 tercatat 2.655.160 jiwa terdiri dari 1.358.623 laki-laki, dan 1.29.537 perempuan. Dari aspek pemerintahan, kota Bandung dipimpin oleh walikota dan wakil walikota. Serta dibantu sekretaris daerah, yang membawahi 3 asisten sekretaris daerah, 17 kepada dinas, 6 kepala badan, 8 kepala bagian, 1 kepala kantor, 4 perusahaan daerah, 1 inspektorat dan 1 kepala satuan polisi pamong praja [5].

2.3. Wisata

Pengertian wisata menurut Intosh dan Goeldner (1984) mengartikan wisata sebagai ilmu, seni, dan bisnis yang berhubungan dengan menarik, memindahkan, mengakomodasikan secara ramah untuk memenuhi kebutuhan dan keinginan para pengunjung. Wisata merupakan segala sesuatu yang dapat menarik minat wisatawan untuk dating, melihat, dan melakukan aktivitas wisata. Sumber daya terdapat dalam ruang atau disebut *geosfer* [6]. *Geosfer* merupakan objek wisata yang dapat dijabarkan sebagai berikut ;

1. Iklim dan cuaca, yaitu berupa panas, sejuk dan dingin. Masing-masing cuaca tersebut memiliki daya tarik sendiri dan dapat mempengaruhi aktivitas wisata yang dilakukan.
2. Bentuk lahan, seperti perbukitan, daratan, pegunungan, dan gunung api. Bentuk lahan ini berkorelasi dengan cuaca, sehingga menghasilkan berbagai argowisata, wisata gunung berapi, air panas, gua dan sebagainya, termasuk dalam kategori wisata alam.
3. Tata air, yaitu berupa danau, sungai dan laut, termasuk dalam kategori wisata alam.
4. Flora dan fauna menghasilkan objek wisata yang berupa kebun binatang, taman kota, termasuk dalam kategori wisata edukasi.
5. Kehidupan manusia yang akan menghasilkan objek wisata social dan budaya, berupa adat, kesenian, bangunan bersejarah, hasil teknologi modern seperti dunia fantasi, termasuk dalam kategori wisata sejarah dan budaya.

2.4. Aplikasi

Aplikasi adalah penggunaan dalam suatu komputer,instruksi (*instructiom*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga komputer dapat memproses *input* menjadi *output*. Menurut Kamus Kamus Besar Bahasa Indonesia, Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna. Aplikasi merupakan rangkaian kegiatan atau perintah untuk dieksekusi oleh komputer. Program merupakan kumpulan *instruction set* yang akan dijalankan oleh pemroses, yaitu berupa *software*. Bagaimana sebuah sistem komputer berpikir diatur oleh program ini.

Program inilah yang mengendalikan semua aktifitas yang ada pada pemroses. Program berisi konstruksi logika yang dibuat oleh manusia, dan sudah diterjemahkan ke dalam bahasa mesin sesuai dengan format yang ada pada *instructionset*. Program aplikasi merupakan program siap pakai yang dirancang untuk melaksanakan suatu fungsi bagi pengguna atau aplikasi yang lain. Contoh

aplikasi ialah program pemproses kata dan *web browser*. Aplikasi akan menggunakan *system operasi* (OS) komputer dan aplikasi yang lainnya yang mendukung. Istilah ini mulai perlahan masuk ke dalam istilah Teknologi Informasi semenjak tahun 1993, yang biasanya juga disingkat dengan *app*.

Secara historis, aplikasi adalah *software* yang dikembangkan oleh sebuah perusahaan. App adalah sofware yang dibeli perusahaan dari tempat pembuatnya. Industri PC tampaknya menciptakan istilah ini untuk merefleksikan medan pertempuran persaingan yang baru, yang parallel dengan yang terjadi antar sistem operasi yang dimunculkan [7].

2.5. Android

. Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis linux yang mencakup sistem operasi, middleware dan aplikasi. Android menyediakan platform yang terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/smartphone. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Aliance yang merupakan konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. Pada saat perilisan perdana Android, 5 November 2007, Android bersama Open Handset Alliance menyatakan mendukung pengembangan open source pada perangkat mobile .

Di dunia terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Service* (GMS) dan kedua adalah yang benar – benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD). Sekitar September 2007 Google mengenalkan Nexus One, salah satu jenis smartphone yang menggunakan Android sebagai sistem operasinya. Telepon selular ini diproduksi oleh HTC Corporation dan tersedia di pasaran pada 5 Januari 2008. Pada 9 Desember 2008, diumumkan anggota baru yang bergabung dalam progam kerja Android ARM Holdings, Atheros Communication, diproduksi oleh Asustek Computer Inc,

Garmin Ltd, Softbank, Sony Ericson, Toshiba Corp, dan Vodafone Group Plc. Seiring pembentukan Open Handset Alliance, OHA mengumumkan produk perdana mereka Android, perangkat mobile yang merupakan modifikasi kernel Linux 2.6. Sejak Android dirilis telah dilakukan berbagai pembaharuan berupa perbaikan bug dan penambahan fitur baru [8].

2.5.1. Android SDK (*Software Development Kit*)

Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak untuk ponsel yang meliputi sistem operasi, middleware dan aplikasi kunci yang di-*release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Sebagai platform aplikasi-netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan [8].

2.5.2. Arsitektur Android

Secara Garis besar arsitektur android dapat dijalankan dan digambarkan sebagai berikut [8] :

1. *Application* dan *Widgets*

   Application dan *Widgets* adalah layer dimana user berhubungan dengan aplikasi saja, dimana biasanya user men-download aplikasi, melakukan instalasi dan menjalankan aplikasi.

2. *Application Frameworks*

   Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau member kemampuan untuk membangun aplikasi yang inovatif. Pegembang bebas untu mengakses perangkat keras, akses informasi *resource,* menjalankan *service background,* mengatur alarm, dan menambahkan status notifikasi. Komponen-komponen yang termasuk di dalam *applications frameorks* adalah sebagai berikut :

a. *Views*
b. *Content provider*
c. *Resource manager*
d. *Notification manager*
e. *Activity manager*

3. *Libraries*

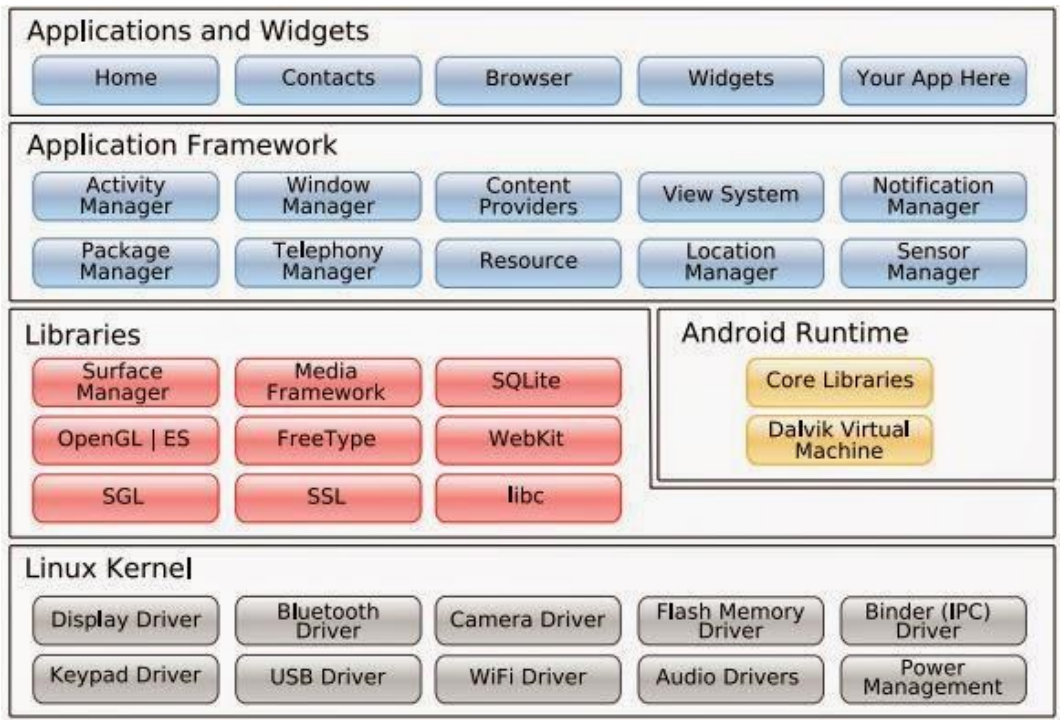
Libraries adalah layer dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses libraries untuk menjalankan aplikasinya.

4. Android *Runtime*

Layer yang membuat aplikasi Android dapat djalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik virtual machine (DVM)* merupakan mesin yang membentuk dasar kerangka aplikasi Android.

5. Linux Kernel

Linux Kernel adalah layer dimana inti dari operating system dari Android itu berada. Berisi *file-file* system yang mengatur sistem *processing*, *memory*, *recource*, *drivers*, dan sistem-sistem operasi android lainnya.



Gambar 2. 1 Arsitektur Android

2.5.3. Fundamental Android

Aplikasi Android ditulis dalam bahasa pemrograman Java. Kode Java dikompilasi bersama dengan data file resource yang dibutuhkan oleh aplikasi, dimana prosesnya di-package oleh tools yang dinamakan "apt tools" ke dalam paket Android sehingga menghasilkan file dengan ekstensi apk. File apk itulah yang disebut dengan apikasi. Ada empat jenis komponen pada aplikasi Android yaitu [8]:

1. *Activities*

   Suatu aktivity akan menyajikan *user interface* (UI) kepada pengguna, sehingga pengguna dapat melakukan interasksi. Satu *activity* biasanya akan dipakai untuk meampilkan aplikasi atau yang bertindak sebagai user interface saat aplikasi diperlihatkan kepada pengguna.

2. *Service*

   Service tidak memiliki *Graphic User interface* (GUI), tetapi *service* berjalan secara background.

3. *Broadcast receiver* berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. *Broadcast receiver* tidak memiliki *user infterface* (UI), tetapi memiliki sebuah *activity* untuk merespon informasi yang diterima.

4. *Content provider*

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

2.5.4. Komponen Android

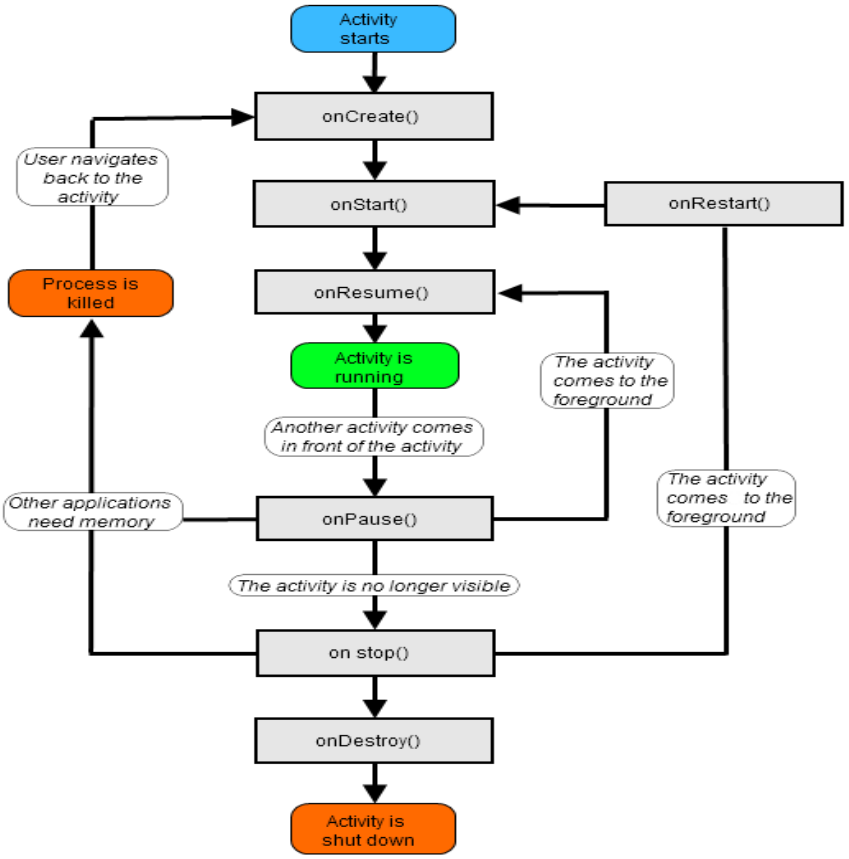
Sebuah aplikasi android akan terdiri atas beberapa komponen penting sebagai berikut [9] :

1. *Activity*

Activity merupakan layar tampilan pada sebuah aplikasi android. Secara sederhananya bahwa *activity* adalah layar yang pengguna dapat melihatnya. Sebuah aplikasi android dapat terdiri atas beberapa *activity* yang dapat dipindah dalam rentang waktu tertentu ketika menjalankan aplikasi. *Activity* ini merupakan

komponen aplikasi yang umum dan sering berhubungan dengan tampilan, dimana user akan berinteraksi dengan aplikasi yang dibuat.

Skema *lifecycle* dari sebuah *activity* adalah sebagai berikut :



Gambar 2. 2 *Lifecycle Activity*

a. Ketika sebuah *activity* dipanggil, maka kode yang ada pada method onCreate() akan dijalankan. Dan method ini juga akan memanggil satu parameter informasi status terakhir yang telah disimpan oleh method onSaveInstanceState().
b. Method onStart() digunakan untuk mengeksekusi *activity* ketika sebuah *user interface* ditampilkan.
c. Method onPause() akan dipanggil ketika ada *activity* lain yang akan mneggantikan *activity* yang ada.

d. Method onStop() digunakan ketika aplikasi sudah tidak dijalankan atau tidak dibutuhkan untuk sementara waktu.
e. Method onRestart() digunakan ketika *activity* di-restart dari posisi semula onStop().
f. Method onDestroy() akan dijalankan setelah *activity* aplikasi di-destroy atau ketika habis memori sehingga aplikasi akan di *terminate.*

2. *Views*

   Views merupakan *widget interface* atau antarmuka pada tampilan aplikasi android, misalnya seperti *button*, *input text*. *Views* akan mempunyai atribut yang dapat diubah (diklik, *double* klik) atau diubah tampilannya (warna, ukuran).

3. *Services*

   Services adalah layanan yang dilakukan dari perintah yang diberikan. Dan dijalankan di belakang layar atau tanpa menggunakan *user interface* pengguna. Misalnya seperti MP3 player, yang menjalankan musik selagi pengguna melakukan berbagai *activity.*

4. *Content Provider*

   Content provider merupakan *interface* terstruktur untuk data.

5. *Intents*

   Intents merupakan pesan yang bersifat asinkon yang akan memerintahkan aplikasi meminta sesuatu dari komponen yang ada pada sistem android. Secara sederhananya, intents merupakan aksi yang akan dilakukan setelah mendapatkan perintah dari *services* atau *activity*.

6. *Broadcast Receiver*

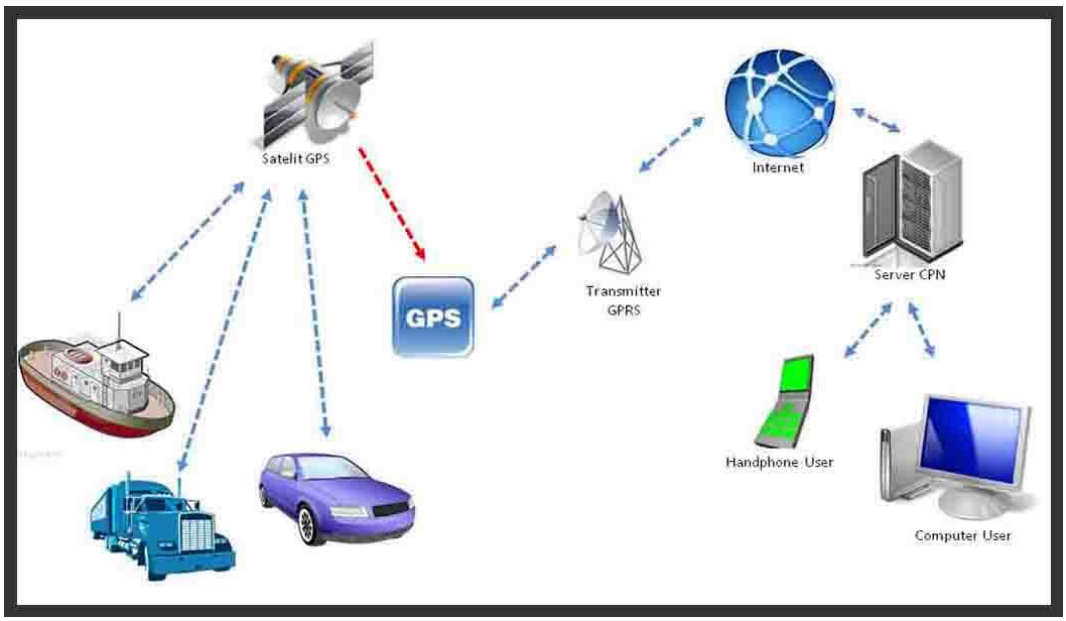
   Broadcast Receiver digunakan untuk menerima pesan dari sistem atau intents.

7. *Widgets*

   Widgets meruapakan komponen interaktif yang ada pada layar utama android. Beberapa ikon *widget* akan berada interaktif pada layar yang memperbolehkan pengguna melakukan aksi tertentu.

2.6. *Global Positioning System*

Global Positioning System (GPS) merupakan sistem navigasi yang menggunakan satelit dalam penggunaannya. GPS terdiri dari tiga bagian yaitu sistem kontrol, satelit dan pengguna. Sistem kontrol adalah bagian yang mengontrol pergerakan satelit – satelit yang ada dan saling berinteraksi satu sama lain kemudian pengguna adalah alat navigasi yang digunakan seperti perangkat mobile yang kini sudah memiliki fitur GPS di dalamnya. GPS biasanya digunakan untuk menunjukan suatu lokasi yang berada di permukaan bumi dengan tingkat akurasi yang cukup baik yaitu kurang dari 10 meter selama tidak ada benda padat yang dapat menghambat sinyal untuk mendapatkan lokasi pengguna [10].



Gambar 2. 3 Cara Kerja GPS

Cara kerja GPS secara logic ada 5 langkah:

1. Memakai perhitungan "*triangulation*" dari satelit.
2. Untuk perhitungan "*triangulation*", GPS mengukur jarak menggunakan travel time sinyal radio.
3. Untuk mengukur travel time, GPS memerlukan akurasi waktu yang tinggi.
4. Untuk perhitungan jarak, kita harus tahu dengan pasti posisi satelit dan ketingian pada orbitnya.

5. Terakhir harus menggoreksi delay sinyal waktu perjalanan di atmosfer sampai diterima reciever.

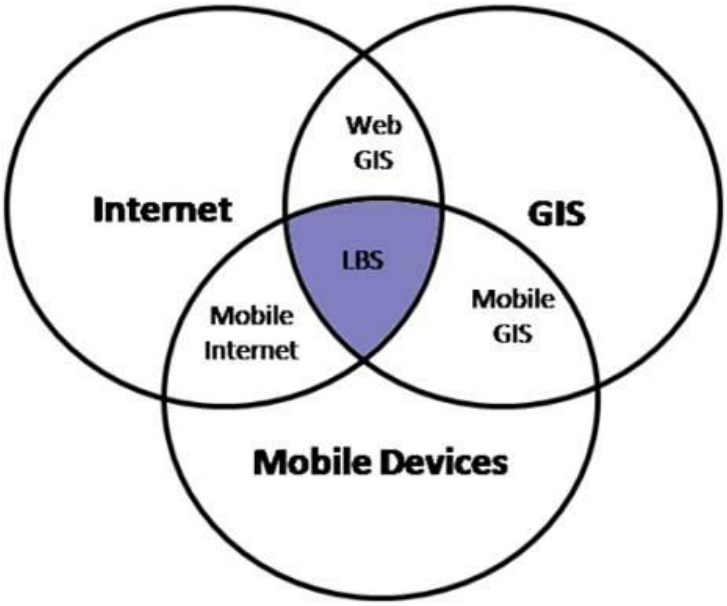
GPS dapat melakukan perhitungan dan menentukan posisi user dan menampilkan dalam *maps*. Jika sudah dapat menyimpan posisi *user* selanjutnya GPS dapat menghitung informasi lain, seperti kecepatan, arah tuju, rute, tujuan perjalanan serta jarak tujuan. Perhitungan waktu yang akurat sangat menentukan informasi lokasi *user*.

2.7. *Location Based Services*

Layanan lokasi didefinisikan sebagai perangkat lokasi/posisi dengan adanya informasi lain sehingga dapat memberikan layanan yang mengintegrasikan nilai tambah pengguna mobile. LBS (*Location Based Services*) atau layanan berbasis lokasi adalah sebuah layanan informasi yang dapat diakses dengan perangkat bergerak (*mobile*) melalui jaringan dan mampu menampilkan posisi secara geografis keberadaan perangkat tersebut. LBS dapat berfungsi sebagai layanan untuk mengidentifikasi lokasi dari seseorang atau objek tertentu.

Hal yang paling penting dari LBS adalah dapat bekerja sesuai yang diinginkan oleh pengembang aplikasi android. Androidpun memungkinkan pengembang untuk menentukan metode pencarian lokasi yang dibutuhkan dan juga dapat mengatur kebutuhan daya, biaya dan akurasi berdasarkan spesifik yang dibutuhkan untuk aplikasi yang dikembangkan.

LBS dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu,*Geographic Information System*, *Internet Service*, dan *Mobile Devices*. Hal ini dapat dilihat pada gambar dibawah ini [11]:



Gambar 2. 4 Sebagai Simpang Tiga

LBS dapat dibagi menjadi dua yaitu [11]:

1. *Pull Services* : layanan diberikan berdasarkan permintaan dari pelanggan akan kebutuhan suatu informasi.
2. *Push Services* : Layanan ini diberikan langsung oleh service provider tanpa menunggu permintaan dari pelanggan, tentu saja informasi yang dberikan tetap berkaitan dengan kebutuhan.

2.7.1. Unsur Utama LBS

LBS atau layanan berbasis lokasi adalah istilah umum yang digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang kita gunakan. Dua unsur utama dari Location Based Services (LBS) adalah [11]:

1. *Location Manager* (API Maps): Menyediakan perangkat bagi sumber atau source untuk LBS.API (*Application Programming Interface*)Maps menyediakan fasilitas untuk menampilkan atau memanipulasi peta. Paket ini berada pada “com.goole.android.maps;”.

2. *Location Providers* (API *Location*): Menyediakan teknologi pencarian lokasi yang digunakan oleh perangkat. PAI Location berhubungan dengan dara GPS (*Global Positioning System*) dan data lokasi real-time. API Location berada pada paket Android yaitu dalam paket "android.location". Lokasi, perpindahan, serta kedekatan dengan lokai tertentu dapat ditentukan melalui *Location Manager*.

2.7.2. Komponen Utama LBS

Terdapat lima komponen pendukung utama dalam teknologi layanan berbasis lokasi, antara lain [11]:

1. Piranti *Mobile*, adalah satu komponen penting dalam LBS. Piranti ini berfungsi sebagai alat bantu bagi pengguna untuk meminta informasi. Hasil dari informasi yang diminta dapat berupa teks, suara, gambar dan lain sebagainya. Piranti *mobile* yang dapat digunakan bisa berupa PDA, *smartphone*, laptop. Selain itu, piranti *mobile* dapat juga berfungsi sebagai alat navigasi di kendaraan seperti halnya navigasi berbasis GPS.
2. Jaringan Komunikasi, komponen ini berfungsi sebagai jalur penghubung yang dapat mengirimkan data-data oleh pengguna dari piranti mobile-nya untuk kemudian ditujukan ke penyedia layanan dan kemudian hasil permintaan tersebut dikirimkan kembali oleh penyedia layanan kepada pengguna.
3. Komponen *Positioning* (penujuk posisi/lokasi), setiap layanan yang di berikan oleh penyedia layanan biasanya akan berdasarkan pada posisi pengguna yang meminta layanan tersebut. Oleh karena itu diperlukan komponen yang berfungsi sebagai pengolah/pemroses yang akan menentukan posisi pengguna layanan saat itu. Posisi pengguna tersebut bisa didapatkan melalui jaringan komunikasi mobile atau juga menggunakan *Global Positioning System* (GPS).
4. Penyedia layanan dan aplikasi, merupakankomponen LBS yang memberikan berbagai macam layanan yang bsia digunakan oleh pengguna. Sebagai contoh ketika pengguna meminta layanan agar bisa tahu posisinya saat itu, maka aplikasi dan penyedia layanan langsung memproses permintaan tersebut, mulai dari menghitung dan menetukan posisi pengguna, menentukan rute jalan,

mencari *Yellow Pages* sesuai dengan permintaan, dan masih banyak lagi yang lainnya.

5. Penyedia data dan konten, penyedia layanan tidak selalu menyimpan seluruh data dan informasi yang diolahnya. Karena bisa jadi berbagai macam data dan informasi yang diolah tersebut berasal dari pengembang atau pihak ketiga yang memang memiliki otoritas untuk menyimpannya.

2.8. *Geofence*

Geofence adalah parameter *virtual* untuk wilayah geografis di dunia nyata. Parameter *virtual* yang ditetapkan ini bisa menjadi statis ataupun dinamis. Salah satu contoh khas dari *geofence* dinamis adalah radius sekitar toko atau titik lokasi. *geofence* statis dapat di atur dari batas yang telah ditetapkan, seperti zona kehadiran di sekolah atau batas lingkungan. Kostum digital *geofence* juga dapat digunakan, jika perangkat yang menyediakan layanan berbasis lokasi (LBS) pengguna masuk atau keluar *geofence*, perangkat akan menerima pesan *notifikasi*.

geofence adalah parameter virtual untuk wilayah geografis di dunia nyata. Parameter virtual yang ditetapkan ini bisa menjadi statis ataupun dinamis. Salah satu contoh dari *geofence* dinamis adalah radius sekitar *took* atau titik lokasi suatu objek tempat. *Geofence* dapat diatur dari batas yang telah ditetapkan, seperti zona kehadiran di sekolah atau batas lingkungan. Kostum digital *geofence* juga dapat digunakan, jika perangkat yang menyediakan layanan berbasis lokasi (LBS) pengguna masuk atau keluar dari *geofence*, perangkat akan menerima pesan *notifikasi*. *Notifikasi* ini berisi informasi tentang lokasi objek tertentu yang sudah ditandai sebelumnya.

geofence menggabungkan kesadaran lokasi pengguna dengan kesadaran kedekatan pengguna pada lokasi *geofence*. Untuk menandai lokasi, di perlukan untuk menentukan lintang dan bujur. Pada pembangunan aplikasi pemandu wisata kota bandung ini teknologi *geofence* dipakai untuk mengetahui tempat-tempat wisata di sekitar area pengguna [12].

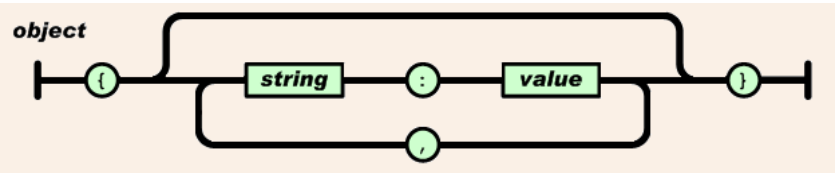
2.9. JSON (*JavaScript Object Notation*)

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. Format ini dibuat berdasarkan bagian dari Bahasa Pemrograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran data. JSON terbuat dari dua struktur [13] :

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau associative array.
2. Daftar nilai terurutkan (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

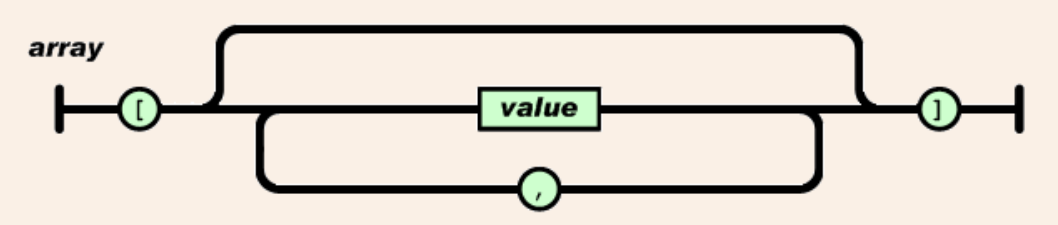
Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemrograman modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemrograman yang juga berdasarkan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

1. Objek adalah sepasang nama / nilai yang tidak terurutkan. Objek dimulai dengan { (kurung kurawal buka) dan diakhiri dengan } (kurung kurawal tutup). Setiap nama diikuti dengan : (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh , (koma). Objek biasanya digunakan untuk menyimpan data tunggal dalam bentuk JSON .



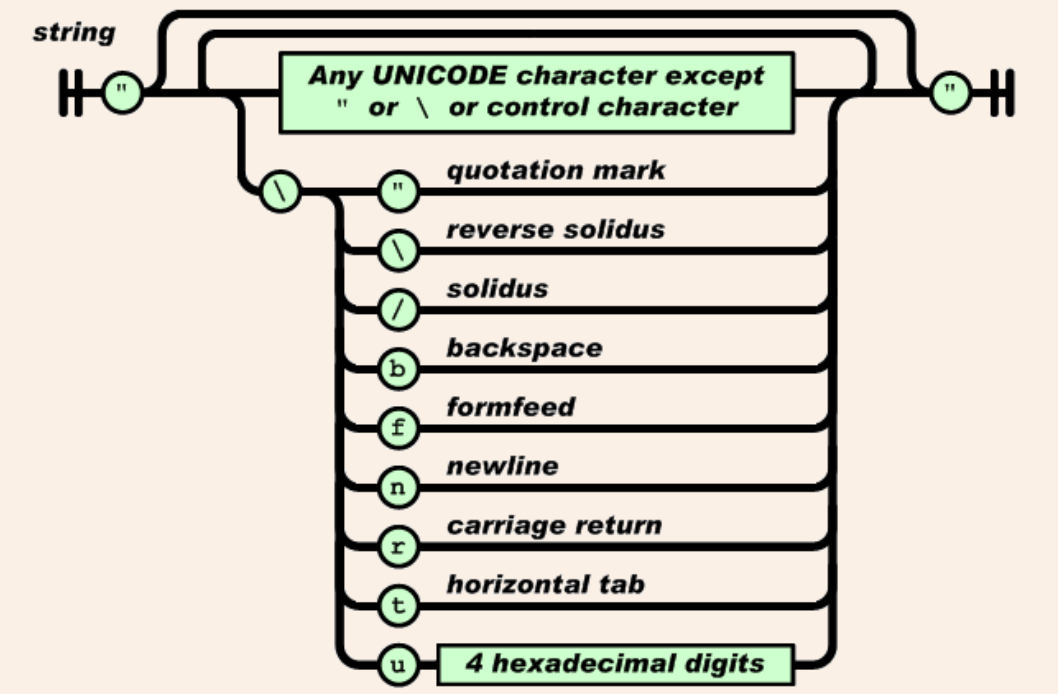
Gambar 2. 5 Objek JSON

2. Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma). Larik dalam JSON dapat digunakan sebagai value dari JSON object hal ini dapat berguna jika JSON menyimpan data bertingkat .



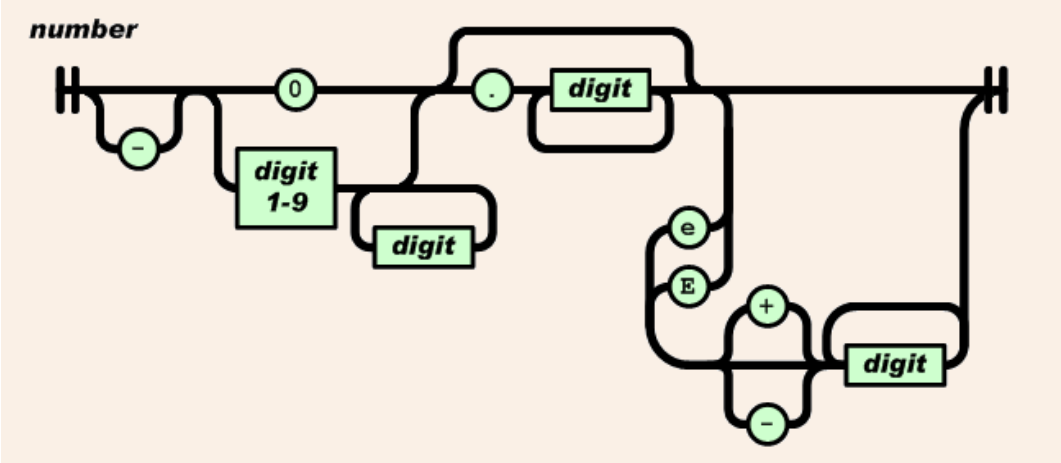
Gambar 2. 6 *Array* JSON

3. String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.



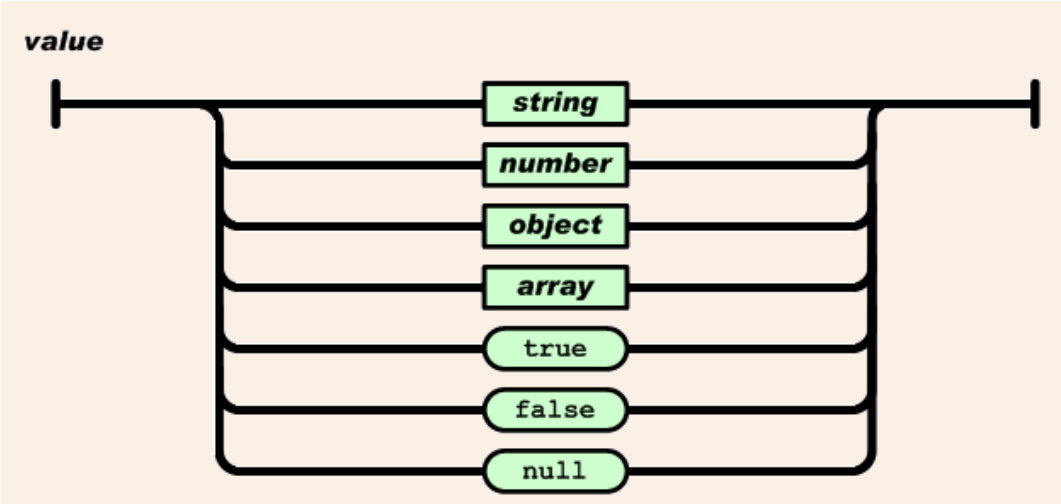
Gambar 2. 7 *String* JSON

4. Angka adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan.



Gambar 2. 8 Angka JSON

5. Nilai (*value*) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur-struktur tersebut dapat disusun bertingkat.



Gambar 2. 9 Nilai JSON

2.10. Google Maps API

Google Maps API adalah sebuah layanan yang memungkinkan Google Maps agar dapat diintegrasikan ke dalam aplikasi yang memang membutuhkan fitur

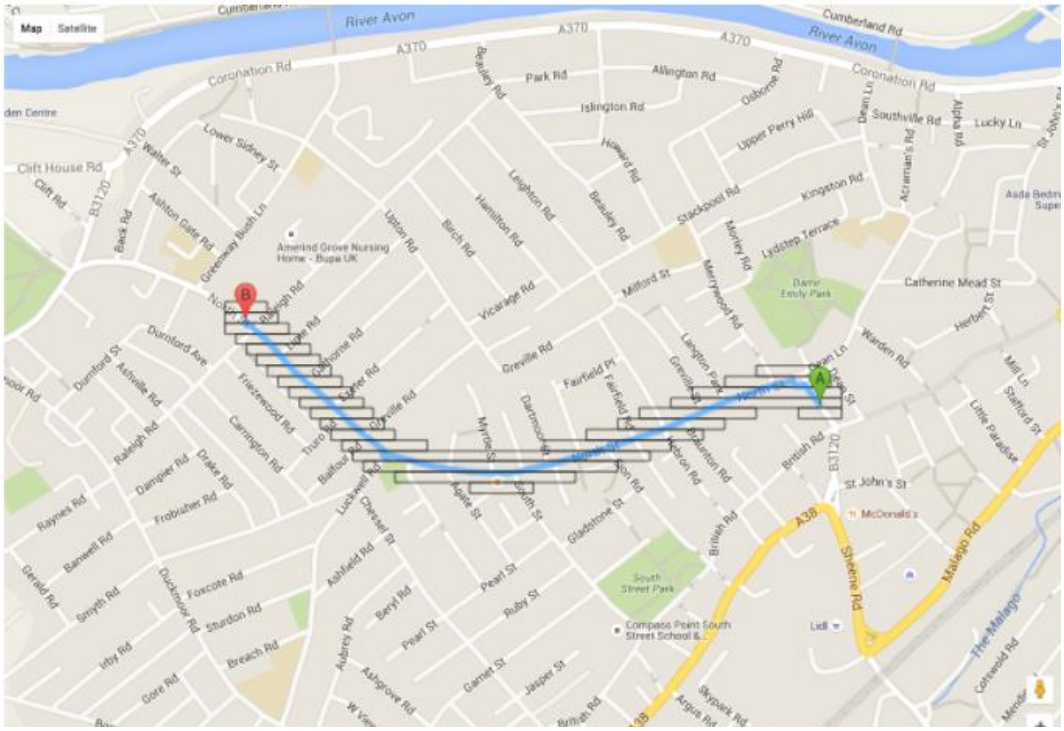
dari Google Maps ini. Google Maps API adalah sebuah *library* Javascript. Dengan menggunakan Google Maps API ini dapat memudahkan pengembang dalam membuat aplikasi yang membutuhkan teknologi peta di dalamnya dan tidak perlu memikirkan cara membuatnya secara manual karena dengan mengunakan Google Maps API ini sudah semua yang dibutuhkan terkait teknologi peta digital sudah cukup lengkap dan dapat dimanfaatkan untuk aplikasi yang membutuhkannya.

Saat ini Google Maps API telah mengalami perkembangan, hingga saat ini Google Maps API telah mencapai versi 3. Namun, Google Maps API versi 3 ini masih hanya bisa digunakan oleh pemrograman berbasis web dengan menggunakan Javascript. Untuk penggunaan Google Maps API pada perangkat mobile seperti misalnya android Google Maps API yang bisa digunakan saat ini telah mencapai versi 2.

Perkembangan pada Google Maps Versi 2 ini adalah dengan memiliki fitur–fitur yang lengkap dari versi sebelumnya salah satunya adalah dapat menampilkan peta dengan sudut yang di inginkan dan berupa 3D view. Selain itu pada Google Maps API terdapat beberapa pilihan model peta yang ditampilkan seperti *Roadmap*, *Satellite*, *Terrain* dan *Hybrid* [14].

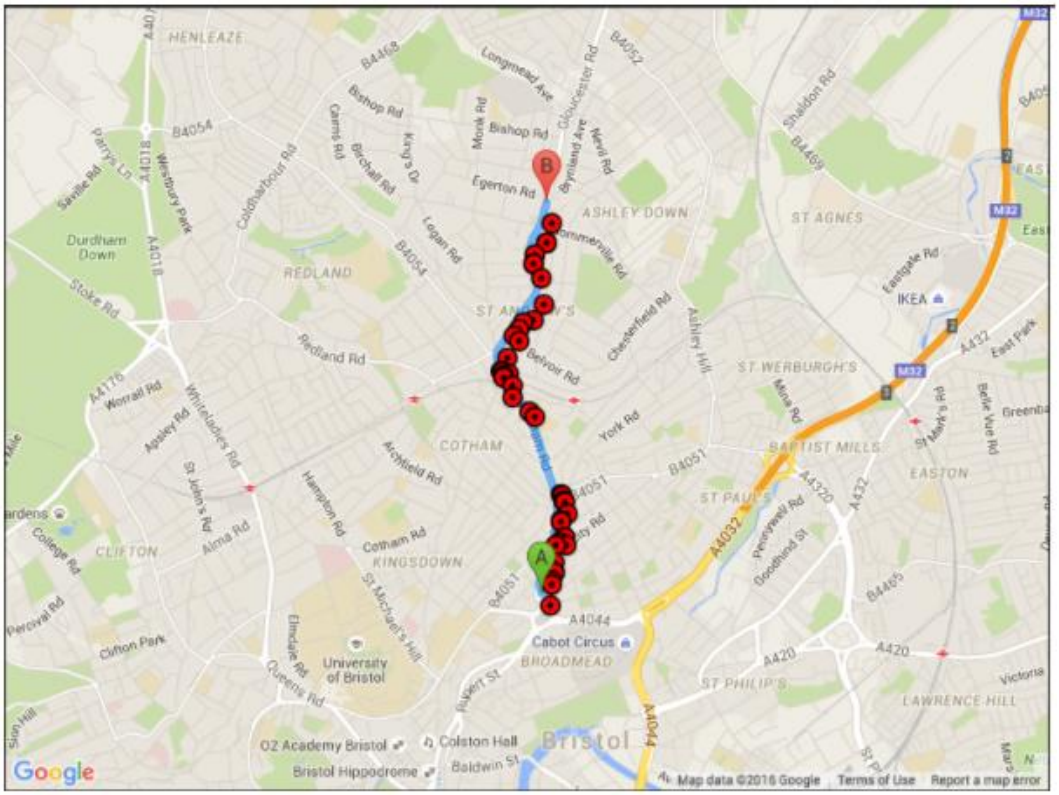
2.11. *Routeboxer*

Pada *library* ini memiliki dua bidang masukkan yang menggunakan API pelengkap otomatis Google Place untuk mendapatkan titik awal dan akhir serta arah antar dua lokasi saat penelusuran. Sederhananya *routeboxer* akan mengarah objek. Kemudian mengembalikan set batas yang mencakup keseluruhan peta [15].



Gambar 2. 10 *Routeboxer*

Semakin banyak kotak yang ada, semakin banyak panggilan API yang akan dibuatnya, namun aplikasi akan mendapatkan hasil yang lebih relevan di sepanjang rute. Hal yang baik tentang *routeboxer* adalah bisa mengurangi kotak pembatas jadi hanya hasil dalam beberapa meter saja yang akan ditampilkan. *routeboxer* juga dapat meningkatkan kotak pembatas sehingga tidak banyak melakukan permintaan, namun trade off adalah karena library berisiko mendapatkan hasil yang akan menyebabkan aplikasi menyimpang dari rute [15] .



Gambar 2. 11 Hasil *Routeboxer*

Tipe-tipe tempat pada google place sebagai berikut [16] :

accounting
airport
amusement_park
aquarium
art_gallery
atm
bakery
bank
bar
beauty_salon
bicycle_store
book_store
bowling_alley
bus_station
general_contractor (tidak digunakan lagi)
grocery_or_supermarket (tidak digunakan lagi)
gym
hair_care
hardware_store
health (tidak digunakan lagi)
hindu_temple
home_goods_store
cafe
campground
car_dealer
car_rental
car_repair
car_wash
casino
cemetery
church
city_hall
clothing_store
convenience_store
courthouse
dentist
department_store
doctor
electrician
electronics_store
embassy
establishment (tidak digunakan lagi)
finance (tidak digunakan lagi)
fire_station
parking
pet_store
pharmacy
physiotherapist
place_of_worship (deprecated)
plumber
police
post_office
real_estate_agency
restaurant
roofing_contractor
rv_park
school
shoe_store
shopping_mall
spa
stadium
storage
store
subway_station
synagogue
taxi_stand
train_station
transit_station

hospital
insurance_agency
jewelry_store
laundry
lawyer
library
liquor_store
local_government_office
locksmith
lodging
florist
food (tidak digunakan lagi)
funeral_home
furniture_store
gas_station
moving_company
museum
night_club
painter
park
travel_agency
university
veterinary_care
zoo
meal_delivery
meal_takeaway
mosque
movie_rental
movie_theater

2.12. PHP (*Personal Hypertext Preprocessor*)

PHP singkatan dari *Personal Hypertext Preprocessor* yaitu bahasa pemrograman *web server-side* yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berada pada *server*. PHP adalah *script* yang digunakan untuk membuat halaman *website* yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru. Semua script PHP dieksekusi pada *server* dimana *script* tersebut dijalankan [17].

PHP pertama kali di kembangkan pada tahun 1995 oleh Rasmus Lerdorf, namun sekarang di ambil oleh oleh *The PHP Group*. Pada awalnya PHP adalah singkatan dari *Personal Home Page*, namun dalam perkembangannya, di ubah menjadi PHP*: Hypertext Preprocessor*, sebuah kepanjangan *rekursif* PHP berjalan pada sisi server sehingga PHP disebut juga sebagai bahasa *ServerSideScripting*. Artinya bahwa dalam setiap menjalankan PHP, wajib adanya web server.PHP ini bersifat *opensource* sehingga dapat dipakai secara cuma-cuma dan mampu lintas *platform*, yaitu dapat berjalan pada sistem operasi *Windows* maupun *Linux*.

PHP juga dibangun sebagai modul pada web apache dan sebagai binary yang dapat berjalan sebagai CGI. Pada pembangunan aplikasi pemandu wisata Kota Bandung khususnya halaman *backend* menggunakan *script* PHP untuk membuat halaman pelaporan.

2.13. *Object Oriented Programming* (OOP)

Booch menyatakan "pemrograman berorientasi objek adalah metode implementasi dimana program diorganisasikan sebagai kumpulan objek yang

bekerja sama. Masing-masing objek merepresentasikan instan dari kelas, dan kelas-keals itu anggota suatu hirarki kelas-kelas yang disatukan lewat keterhubungan pewarisan".

Stroustrup mendeskripsikan pemrograman berorientasi objek sebagai penggabungan abstraksi data dengan pewarisan. Fitur-fitur penting untuk mendukung abstraksi data adalah *constructor* serta *destructor* untuk penciptaan dan pemusnahan objek. Tipe berparameter, mekanisme *exception, object coercions* dan *iterato* [18].

2.14. UML (*Unified Modeling Language*)

UML memiliki 13 jenis diagram resmi yaitu activity diagram , *Class, Comunnication, Component, Composite structure, Deployment, Interaction overview, Object, Package, Sequence, State mechine, Timing,* dan diagram *Use case*. Meskipun jenis jenis diagram ini merupakan cara orang-orang memperlakukan UML. Para perancang UML tidak memandang diagram sebagai bagian yang sentral. Dan hasilnya, jenis-jenis diagram bukanlah hal yang mutlak. secara legal dapat menggunakan elemen-elemen satu jenis diagram untuk diagram yang lain [19].

Pada penelitian ini diagram yang digunakan adalah diagram *Use case, Class, activity,* dan *Sequence.*

1. Diagram *Use Case*

Use case adalah teknik untuk merekam persyaratan fungsional sebuah sistem. *Use case* mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan. Sebuah *use case* adalah serangkaian skenarion yang dikemas menjadi satu oleh tujuan pengguna umum .

Dalam bahasa *use case*, para pengguna disebut sebagai aktor. Aktor merupakan sebuah peran yang dimainkan oleh seorang pengguna dalam kaitannya dengan sistem. aktor tidak harus manusia. Jika sebuah sistem melakukan layanan untuk sebuah sistem komputer lain, sistem lain tersebut merupakan aktor .

Cockburn menjelaskan sebuah skema tingkatan *use case*. Terdapat tingkat *sea level, fish level dan kite level.*

a. *Sea level*

   Inti *use case* berada pada tingkat *sea level. Use case sea level* khususnya mewakili sebuah interaksi diskrit antara aktor utama dan sistem. *use case* semacam ini akan memberikan suatu nilai pada aktor utama dan biasanya memberi waktu beberapa menit sampai setengah jam bagi aktor utama untuk menyelesaikan.

b. *Fish level*

   Use case yang ada pada tinkat *fish level* adalah *use case* yang ada hanya karena mereka dimasukkan oleh *use case sea level.*

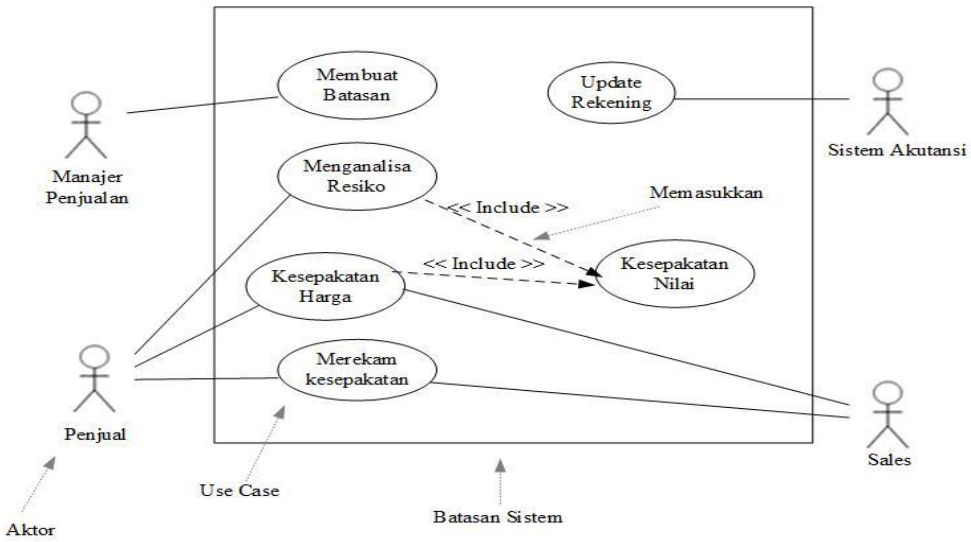
c. *Kite level*

   Use case kite level menampilkan bagaimana *use case sea level* sesuai dengan interaksi bisnis yang lebih luas. *Use case kite level* biasanya merupakan *use case* bisnis.

Use case diagram menampilkan aktor, *use case,* dan hubungan antara mereka:

a. Aktor mana yang menggunakan *use case* mana.
b. *Use case* mana yang memasukkan *use case* lain.

Berikut ini adalah contoh dari diagram *use case* :

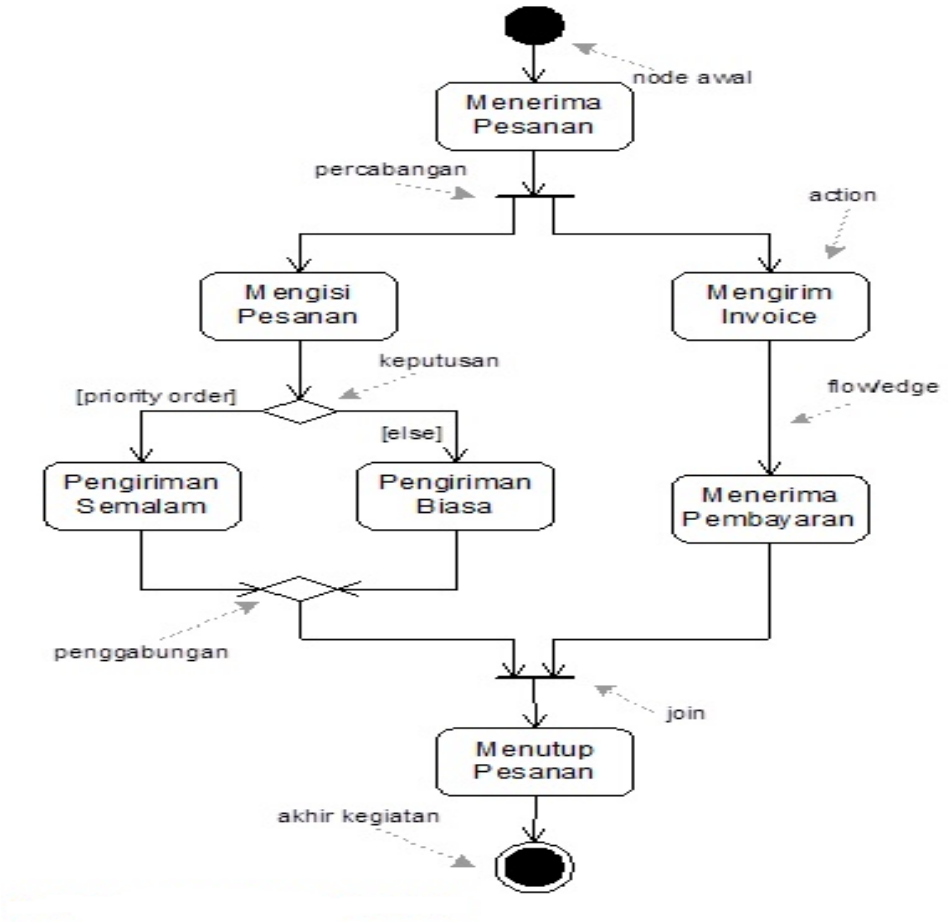


Gambar 2. 12 Contoh *Use Case Diagram*

2. *Activity Diagram*

Activity diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. Dalam beberapa hal, activity diagram memainkan peran mirip sebuah diagram alir, tetapi perbedaan prinsip antara activity diagram dan diagram alir adalah diagram alir mendukung *behavior* pararel.

Activity diagram dapat dipisahkan ke dalam partisi-partisi yang menampilkan *action* mana yang dilakukan oleh sebuah *class* atau organisasi. Model ini biasa disebut dengan *swim lanes.*



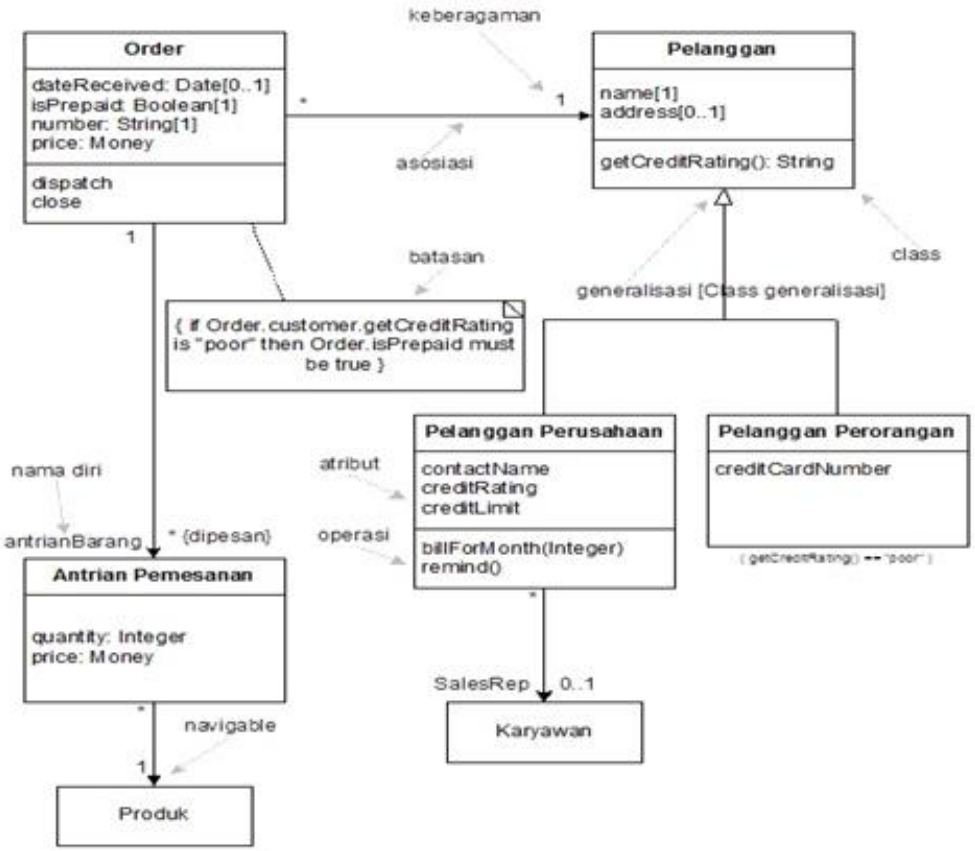
Gambar 2. 13 Contoh *Activity Diagram*

3. *Class Diagram*

Class diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat diantara mereka. Diagram *class* juga

menunjukan properti dan operasi sebuah *class* dan batasan-batasan yang terdapat pada sebuah hubungan-hubungan objek. UML menggunakan istilah fitur sebagai istilah umum yang meliputi properti dan operasi sebuah *class*.

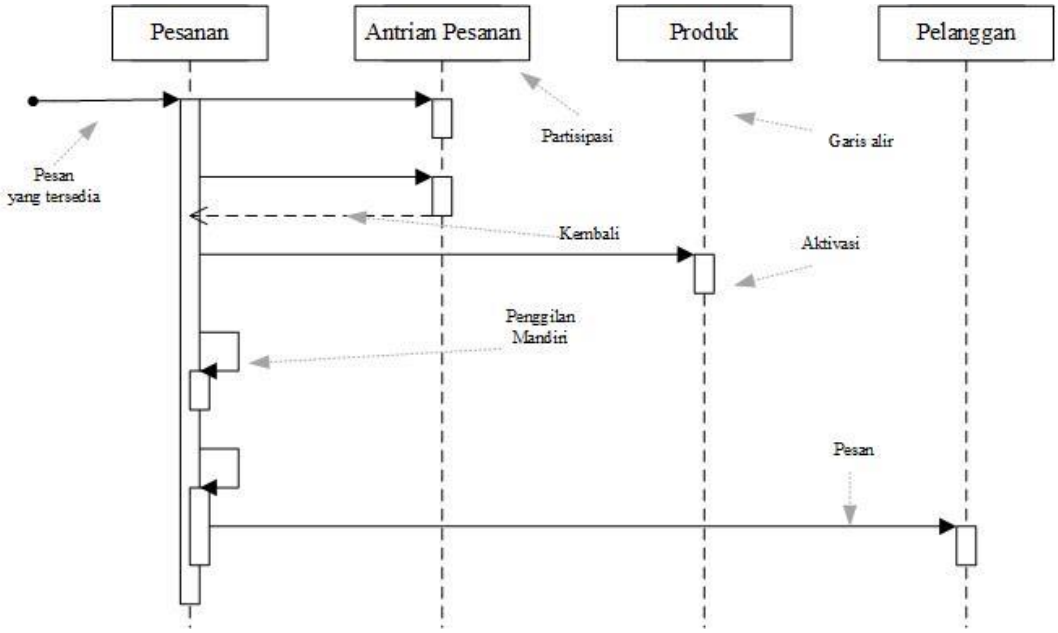
Didalam class dibagi menjadi tiga bagian. Nama *class*, atributnya, dan operasinya.



Gambar 2. 14 Contoh *Class Diagram*

4. *Sequence Diagram*

Sequence diagram secara khusus menjabarkan *behavior* sebuah skenario tunggal. Diagram tersebut menunjukan sejumlah objek contoh dan pesan-pesan yang melewati objek-objekdi dalam *use case*. Sequence diagram menunjukkan interaksi dengan menampilkan setiap partisipan dengan garis alir secara vertikal dan pengurutan pesan dari atas ke bawah.



Gambar 2. 15 Contoh *Sequence diagram*

2.15. *Entity Relationship Diagram* (ERD)

Pemodelan awal basi data yang paling banyak digunakan adalah menggunakan *Entity Relationship Diagram* (ERD). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan basis data tidak perlu menggunakan ERD. ERD biasanya memiliki hubungan *binary* (satu relasi menghubungkan dua buah entitas). Beberapa metode perancangan ERD menoleransi hubungan relasi *ternary* (satu relasi menghubungkan tiga buah relasi) atau N-ary (satu relasi menghubungkan banyak entitas) tapi banyak metode perancangan ERD yang tidak mengizinkan hubungan ternary dan N-ary [20].

2.16. *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) awalnya dikembangkan oleh Chris Gane dan Trish Sarson pada tahun 1979 yang termasuk dalam Structured System Analysis and Design Methodology (SSADM) yang ditulis oleh Chris Gane dan Tris Sarson. Sistem yang dikembangkan ini berbasi pada dekomposisi fungsional dari sebuah sistem. Namun DFD Edward Yourdon dan Tom DeMarco populer digunakan

sebagai model analisis sistem perangkat lunak untuk sistem perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur.

Informasi yang ada di dalam perangkat lunak dimodifikasi dengan beberapa transformasi yang dibutuhkan. DFD sering juga disebut sebagai aliran data atau representasi grafik yang menggambarkan aliran informasi dan transformasi informasi yang diaplikasikan sebagai data yang mengalir dari masukan dan keluaran.

DFD dapat digunakan untuk merepresentasikan sebuah sistem atau perangkat lunak pada beberapa level abstraksi. DFD dapat dibagi menjadi beberapa level yang lebih detail untuk mereprensentasikan aliran informasi atau fungsi yang lebih detail. DFD menyediakan mekanisme untuk pemodelan fungsional atau pemodelan aliran informasi. Oleh karena itu, DFD lebih sesuai digunakan untuk memodelkan fungsi-fungsi perangkat lunak yang akan diimplementasikan menggunakan pemrograman terstruktur karena pemrograman terstruktur membagi-bagi bagiannya dengan fungsi-fungsi dan prosedur-prosedur [20].

2.17. *Travelling Salesman Problem* (TSP)

Travelling Salesman Problem (TSP) adalah masalah untuk mengoptimasi dan menemukan perjalanan yang paling terpendek dan menentukan urutan dari sejumlah lokasi yang harus dilalui satu kali dalam perjalanannya, perjalanan tersebut harus berakhir pada lokasi keberangkatannya dimana user tersebut memulai perjalannya, dengan jarak antara setiap lokasi dengan lokasi lainnya sudah diketahui.

Asumsikan terdapat (*latitude* dan *longitude*) dari 3 tempat wisata tersebut seperti berikut:

Tabel 2. 1 Tabel Lokasi User

Lokasi Awal	Latitude	Longitude
Lokasi Awal (A)	-6.887096	107.61506

Tabel 2. 2 Tabel Lokasi Tempat Wisata

Nama Tempat Wisata	Latitude	Longitude
Tempat wisata satu (B)	-6.890155	107.6069588
Tempat wisata dua (C)	-6.8781668	107.594626
Tempat wisata tiga (D)	-6.9007084	107.6214906

Kemudian tampung semua lokasi-lokasi tersebut pada suatu *array* yang akan dikirim ke google maps api direction untuk mengetahui jarak-jaraknya.

2.18. Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu - *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA [21]. Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas saat membuat aplikasi Android, misalnya:

1. Sistem versi berbasis Gradle yang fleksibel
2. Emulator yang cepat dan kaya fitur
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android
4. Instant Run untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh
6. Alat pengujian dan kerangka kerja yang ekstensif
7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain
8. Dukungan C++ dan NDK
9. Dukungan bawaan untuk Google Cloud Platform, mempermudah pengintegrasian Google Cloud Messaging dan App Engine.

2.19. Pengujian *Black Box*

Pengujian untuk mengetahui apakah semua fungsi perangkat lunak telah berjalan semestinya sesuai dengan kebutuhan fungsional yang telah didefinsikan. Metode *Black Box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program.

Pengujian *Black Box* dapat menemukan kesalahan dalam kategori berikut [22] :

1. Fungsi-fungsi yang tidak benar atau hilang
2. Kesalahan interface
3. Kesalahan dalam strutur data atau akses basisdata eksternal
4. Inisialisasi dan kesalahan terminasi
5. validitas fungsional
6. kesensitifan sistem terhadap nilai input tertentu
7. batasan dari suatu data

2.20. Pengujian Beta

Pengujian beta dianggap sebagai bentuk pengujian penerimaan perangkat lunak oleh pengguna eksternal. Dibagikan kepada pengguna yang berada di luar tim pemrogram sebagai penguji beta. Perangkat lunak dibagikan ke sekelompok orang sehingga pengujian bisa memastikan bahwa produknya memiliki kesalahan atau bug yang kecil [23].

2.21. Skala *Likert*

Skala *Likert* digunakan untuk mengukur sikap, pendapat dan persepsi seseorang atau sekelompok orang tentang penelitian. Dengan skala *likert,* variabel yang akan diukur dijabarkan menjadi indikator variabel. Kemudian indikator tersebut dijadikan sebagai titik tolak untuk menyusun item-item instrumen yang dapa berupa pernyataan atau pertanyaan.

Tabel 2. 3 Panduan Pemberian Skor

Jenis Pertanyaan	Alternatif Jawaban				
	SS	S	RG	TS	STS
Skor	5	4	3	2	1

Kemudian dengan teknik pengumpulan data angket, maka instrumen tersebut misalnya diberikan kepada 100 orang karyawan yang diambil secara random. Dari 100 orang pegawai setelah dilakukan analisis misalnya :

25 orang menjawab SS
40 orang menjawab ST
5 orang menjawab RG
20 orang menjawab TS
10 orang menjawab STS

Jumlah skor ideal 25 orang yang menjawab SS 25 x 5 = 125
Jumlah skor ideal 40 orang yang menjawab ST 40 x 4 = 160
Jumlah skor ideal 5 orang yang menjawab RG 5 x 3 = 15
Jumlah skor ideal 20 orang yang menjawab TS 20 x 2 = 20
Jumlah skor ideal 10 orang yang menjawab STS 10 x 1 = 10
Jumlah total = 350

Jumlah skor ideal (kriterium) untuk seluruh item = 5 x 100 = 500. Jumlah skor yang diperoleh dari penelitian = 350. Jadi berdasarkan data itu maka tingkat persetujuan terhadap metode kerja baru itu = (350/500) x 100% = 70 % dari data yang diharapkan 100% [24].