BAB 2
TINJAUAN PUSTAKA

2.1 Landasan teori

2.1.1 *Emergency services*

Layanan darurat atau *emergency services* merupakan organisasi yang menjamin keselamatan publik dan kesehatan dengan mengatasi berbagai keadaan darurat, beberapa lembaga ini ada untuk mengatasi beberapa jenis keadaan darurat, seperti polisi, rumah sakit, puskesmas dan pemadam kebakaran, Keadaan darurat merupakan keadaan yang lain dari keadaan normal yang mempunyai kecenderungan atau potensi tingkat yang membahayakan baik bagi keselamatan manusia, harta benda maupun lingkungan jenis-jenis keadaan darurat seperti kebakaran, ledakan, kecelakaan, perampokan dan kerusakan mesin

2.1.2 *Global Positioning System* (GPS)

GPS (Global Positioning System) adalah sistem satelit navigasi dan penentuan posisi yang dimiliki dan dikelola oleh Amerika Serikat. Sistem ini didesain untuk memberikan posisi dan kecepatan tiga-dimensi serta informasi mengenai waktu, secara kontinyu di seluruh dunia tanpa bergantung waktu dan cuaca, bagi banyak orang secara simultan. [5]

Saat ini GPS sudah banyak digunakan orang di seluruh dunia dalam berbagai bidang aplikasi yang menuntut informasi tentang posisi, kecepatan, percepatan ataupun waktu yang teliti. GPS dapat memberikan informasi posisi dengan ketelitian bervariasi dari beberapa millimeter (orde nol) sampai dengan puluhan meter.

Beberapa kemampuan GPS antara lain dapat memberikan informasi tentang posisi, kecepatan, dan waktu secara cepat, akurat, murah, dimana saja di bumi ini tanpa tergantung cuaca. Hal yang perlu dicatat bahwa GPS adalah satu-satunya sistem navigasi ataupun sistem penentuan posisi dalam beberapa abad ini yang memiliki kemampuan handal seperti itu.

Ketelitian dari GPS dapat mencapai beberapa mm untuk ketelitian posisinya, beberapa cm/s untuk ketelitian kecepatannya dan beberapa nanodetik untuk ketelitian waktunya. Ketelitian posisi yang diperoleh akan tergantung pada beberapa faktor yaitu metode penentuan posisi, geometri satelit, tingkat ketelitian data, dan metode pengolahan datanya.

2.1.2.1 Segmen Penyusun Sistem GPS

Secara umum ada tiga segmen dalam sistem GPS yaitu segmen sistem kontrol, segmen satelit, dan segmen pengguna. Satelit GPS dapat dianalogikan sebagai stasiun radio angkasa, yang diperlengkapi dengan antena-antena untuk mengirim dan menerima sinyal –sinyal gelombang. Sinyal-sinyal ini selanjutnya diterima oleh receiver GPS di/dekat permukaan bumi, dan digunakan untuk menentukan informasi posisi, kecepatan, maupun waktu. Selain itu satelit GPS juga dilengkapi dengan peralatan untuk mengontrol attitude satelit. Satelit-satelit GPS dapat dibagi atas beberapa generasi yaitu ; blok I, blok II, blok IIA, blok IIR dan blok IIF. Hingga april 1999 ada 8 satelit blok II, 18 satelit blok II A dan 1 satelit blok II R yang operasional. Secara umum segmen sistem kontrol berfungsi mengontrol dan memantau operasional satelit dan memastikan bahwa satelit berfungsi sebagaimana mestinya Segmen pengguna terdiri dari para pengguna satelit GPS di manapun berada. Dalam hal ini alat penerima sinyal GPS (GPS receiver) diperlukan untuk menerima dan memproses sinyal -sinyal dari satelit GPS untuk digunakan dalam penentuan posisi, kecepatan dan waktu.

Komponen utama dari suatu receiver GPS secara umum adalah antena dengan pre-amplifier, bagian RF dengan pengidentifikasi sinyal dan pemroses sinyal, pemroses mikro untuk pengontrolan receiver, data sampling dan pemroses data (solusi navigasi), osilator presisi , catu daya, unit perintah dan tampilan, dan memori serta perekam data.

2.1.2.2 Prinsip penentuan posisi dengan GPS

Prinsip penentuan posisi dengan GPS yaitu menggunakan metode reseksi jarak, dimana pengukuran jarak dilakukan secara simultan ke beberapa satelit yang telah diketahui koordinatnya. Pada pengukuran GPS, setiap epoknya memiliki empat parameter yang harus ditentukan : yaitu 3 parameter koordinat X,Y,Z atau L,B,h dan satu parameter kesalahan waktu akibat ketidaksinkronan jam osilator di satelit dengan jam di receiver GPS. Oleh karena diperlukan minimal pengukuran jarak ke empat satelit.

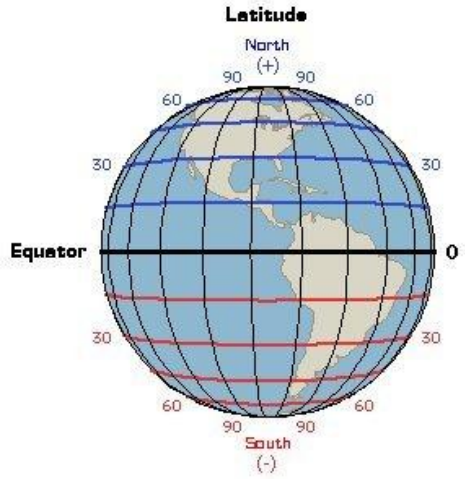
2.1.2.3 Sinyal dan Bias pada GPS

GPS memancarkan dua sinyal yaitu frekuensi L1 (1575.42 MHz) dan L2 (1227.60 MHz). Sinyal L1 dimodulasikan dengan dua sinyal pseudo-random yaitu kode P (Protected) dan kode C/A (coarse/aquisition). Sinyal L2 hanya membawa kode P. Setiap satelit mentransmisikan kode yang unik sehingga penerima (receiver GPS) dapat mengidentifikasi sinyal dari setiap satelit. Pada saat fitur ”*Anti-Spoofing*” diaktifkan, maka kode P akan dienkripsi dan selanjutnya dikenal sebagai kode P(Y) atau kode Y.

Ketika sinyal melalui lapisan atmosfer, maka sinyal tersebut akan terganggu oleh konten dari atmosfer tersebut. Besarnya gangguan di sebut bias. Bias sinyal yang ada utamanya terdiri dari 2 macam yaitu bias ionosfer dan bias troposfer. Bias ini harus diperhitungkan (dimodelkan atau diestimasi atau melakukan teknik differencing untuk metode diferensial dengan jarak baseline yang tidak terlalu panjang) untuk mendapatkan solusi akhir koordinat dengan ketelitian yang baik. Apabila bias diabaikan maka dapat memberikan kesalahan posisi sampai dengan orde meter.

2.1.3 *Latitude* dan *Longitude*

Latitude adalah garis yang melintang di antara kutub utara dan kutub selatan, yang menghubungkan antara sisi timur dan barat bagian bumi. Garis ini memiliki posisi membentangi bumi, sama halnya seperti garis equator (khatulistiwa), tetapi dengan kondisi nilai tertentu[6]. Garis lintang inilah yang dijadikan ukuran dalam mengukur sisi utara-selatan koordinat suatu titik di belahan bumi garis equator dapat dilihat pada gambar 2.1



Gambar 2. 1 Garis equator

Latitude di bedakan menjadi 2 wilayah, yaitu utara atau yang biasa kita sebut lintang utara dan selatan atau yang biasa kita sebut lintang selatan, dimana nilai koordinat di bagian utara selalu positif dan nilai koordinat di bagian selatan adalah negatif.

Berikut nilai-nilai yang dijadikan patokan ukuran garis lintang ini.

1. Garis paling atas (kutub utara) = 90 derajat

2. Garis paling tengah (equator) = 0 derajat, dan

3. Garis paling bawah (kutub selatan) = -90 derajat.

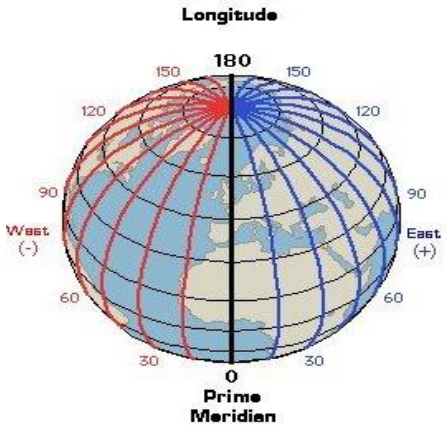
Dengan "mem-persamakan" derajat ke dalam bentuk satuan kilometer (km) maka ukurannya seperti ini :

1 derajat latitude = 111 km

1 menit latitude = 1.85 km

Sedangkan longitude adalah garis membujur yang menghubungkan antara sisi utara dan sisi selatan bumi (kutub). Garis bujur ini digunakan untuk mengukur sisi barat-timur koordinat suatu titik di belahan bumi.

Sama seperti equator pada latitude yang berada ditengah dan memiliki nilai 0 (nol) derajat, pada longitude, garis tengah yang bernilai 0 (nol) derajat disebut garis prime meridian (garis bujur). Sedangkan garis yang berada paling kiri memiliki nilai -90 derajat, dan yang paling kanan memiliki nilai 90 derajat, dapat dilihat pada gambar 2.2.



Gambar 2. 2 Garis bujur

Longitude juga dibedakan menjadi 2 wilayah, yaitu bujur timur dan bujur barat, dimana koordinat yang berada di timur selalu bernilai negatif, dan sebaliknya yang berada di barat selalu positif. Nilai satuan ukuran derajat menjadi kilometer pada longitude juga sama seperti pada latitude.

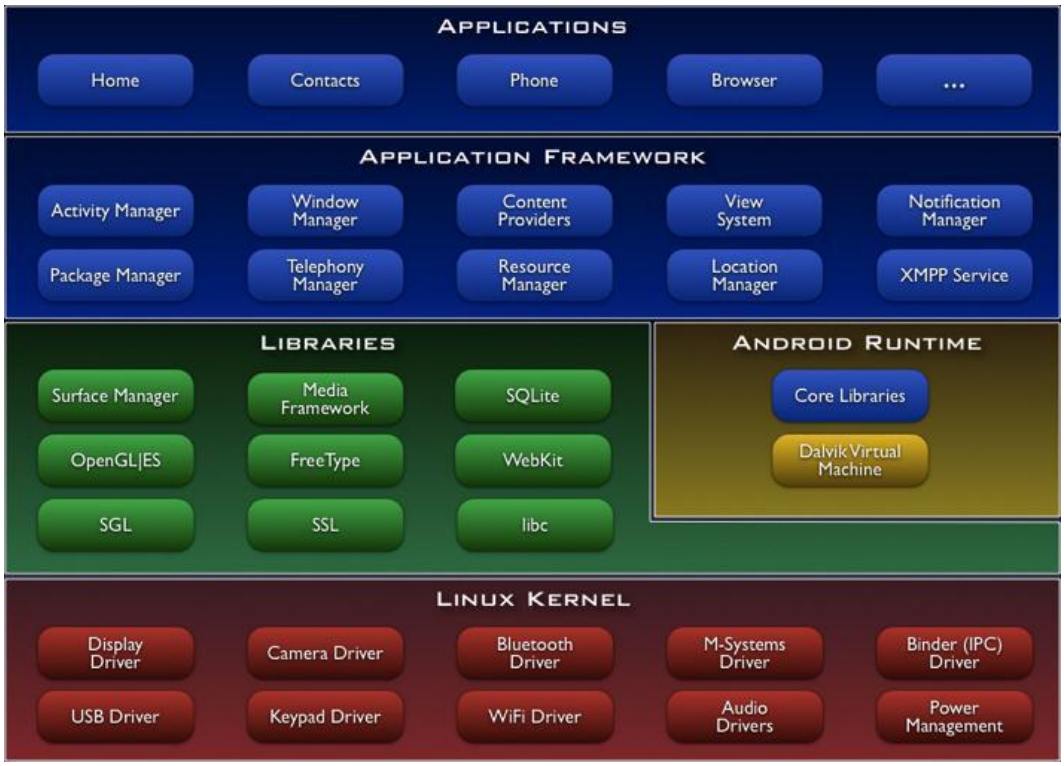
Jadi, dalam metode pengukuran koordinat, suatu titik terlebih dulu diukur derajatnya berdasarkan latitude dan longitude-nya, setelah itu barulah di translasikan kedalam bentuk satuan kilometer, baik itu dalam format degree (DDD) maupun degree-minutes-second (DMS). Berdasarkan pengalaman kemarin saya coba berbagi bagi anda yang mungkin belum mencoba mencari latitude dan longitude tersebut.

2.1.4 Android

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan platform terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam piranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah Open Handset Alliance, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia[7].

2.1.5 Arsitektur Sistem Operasi Android

Pada gambar 2.8 merupakan gambar dari arsitektur system operasi android



Gambar 2. 3 arsitektur sistem operasi android

2.1.5.1 *Linux Kernel*

Android dibangun di atas kernel Linux 2.6. Namun secara keseluruhan

android bukanlah linux, karena dalam android tidak terdapat paket standar yang dimiliki oleh linux lainnya. Linux merupakan sistem operasi terbuka yang handal dalam manajemen memori dan proses. Oleh karenanya pada android hanya terdapat beberapa servis yang diperlukan seperti keamanan, manajemen memori, manajemen proses, jaringan dan driver. Kernel linux menyediakan driver layar, kamera, keypad, WiFi, Flash Memory, audio, dan IPC (Interprocess Communication) untuk mengatur aplikasi dan lubang keamanan

2.1.5.2 *Libraries*

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar Berkeley Software Distribution (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya:

1. Media Library untuk memutar dan merekam berbagai macam format audio dan video.
2. Surface Manager untuk mengatur hak akses layer dari berbagai aplikasi.
3. Graphic Library termasuk didalamnya SGL dan OpenGL, untuk tampilan 2D dan 3D.
4. SQLite untuk mengatur relasi database yang digunakan pada aplikasi.
5. SSl dan WebKit untuk browser dan keamanan internet.

Pustaka-pustaka tersebut bukanlah aplikasi yang berjalan sendiri, namun hanya dapat digunakan oleh program yang berada di level atasnya. Sejak versi Android 1.5, pengembang dapat membuat dan menggunakan pustaka sendiri menggunakan Native Development Toolkit (NDK).

2.1.5.3 Android *Runtime*

Pada android tertanam paket pustaka inti yang menyediakan sebagian besar fungsi android. Inilah yang membedakan Android dibandingkan dengan

system operasi lain yang juga mengimplementasikan Linux. Android Runtime merupakan mesin virtual yang membuat aplikasi android menjadi lebih tangguh dengan paket pustaka yang telah ada. Dalam Android Runtime terdapat 2 bagian utama, diantaranya:

1. Pustaka Inti, android dikembangkan melalui bahasa pemrograman Java, tapi Android Runtime bukanlah mesin virtual Java. Pustaka inti android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus android.

2. Mesin Virtual Dalvik, Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland.

Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format Dalvik Executable (*.dex). Dengan format ini Dalvik akan mengoptimalkan efisiensi penyimpanan dan pengalamatan memori pada file yang dieksekusi. Dalvik berjalan di atas kernel Linux 2.6, dengan fungsi dasar seperti threading dan manajemen memori yang terbatas.

2.1.5.4 *Aplication Framework*

Kerangka aplikasi menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi android. Selain itu, juga menyediakan abstraksi generic untuk mengakses perangkat, serta mengatur tampilan *User* Interface dan sumber daya aplikasi. Bagian terpenting dalam kerangka aplikasi android adalah sebagai berikut:

1. *Activity Manager*, berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan ”Backstack“ untuk navigasi penggunaan.

2.*Content Providers*, berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.

3. R*esuource Manager*, untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya diluar kode program, seperti karakter, grafik, dan file layout.

4. *Location Manager*, berfungsi untuk memberikan informasi detail mengenai lokasi perangkat android berada.

5. *Notification Manager*, mencakup berbagai macam peringatan seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada status bar.

2.1.5.5 *Aplication Layer*

Puncak dari diagram arsitektur android adalah lapisan aplikasi dan widget. Lapisan aplikasi merupakan lapisan yang paling tampak pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika digunakan tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android *runtime* dengan menggunakan kelas dan *service* yang tersedia pada Framework aplikasi. Lapisan aplikasi android sangat berbeda dibandingkan dengan system operasi lainnya. Pada android semua aplikasi, baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan diatas lapisan aplikasi dengan menggunakan pustaka API (*Application Programming Interface*) yang sama.

2.1.6 Komponen Aplikasi Android

Fitur penting android adalah bahwa satu aplikasi dapat menggunakan elemen dari aplikasi lain (untuk aplikasi yang memungkinkan). Sebagai contoh, sebuah aplikasi memerlukan fitur *scroller* dan aplikasi lain telah mengembangkan fitur s*croller* yang baik dan memungkinkan aplikasi lain menggunakannya. Maka pengembang tidak perlu lagi mengembangkan hal serupa untuk aplikasinya, cukup menggunakan *scroller* yang telah ada. Agar fitur tersebut dapat bekerja, sistem harus dapat menjalankan aplikasi ketika setiap bagian aplikasi itu dibutuhkan, dan pemanggilan objek java untuk bagian itu.

Oleh karenanya android berbeda dari sistem-sistem lain, Android tidak memiliki satu tampilan utama program seperti fungsi main() pada aplikasi lain. Sebaliknya, aplikasi memiliki komponen penting yang memungkinkan sistem untuk memanggil dan menjalankan ketika dibutuhkan

2.1.6.1 *Activities*

Activity merupakan bagian yang paling penting dalam sebuah aplikasi, karena *Activity* menyajikan tampilan visual program yang sedang digunakan oleh pengguna. Setiap *Activity* dideklarasikan dalam sebuah kelas yang bertugas untuk menampilkan antarmuka pengguna yang terdiri dari *Views* dan respon terhadap *Event*. Setiap aplikasi memiliki sebuah *activity* atau lebih. Biasanya pasti akan ada *activity* yang pertama kali tampil ketika aplikasi dijalankan.

Perpindahan antara *activity* dengan *activity* lainnya diatur melalui sistem, dengan memanfaatkan *activity stack*. Keadaan suatu *activity* ditentukan oleh posisinya dalam tumpukan a*citivity*, LIFO (*Last In First Out*) dari semua aplikasi yang sedang berjalan. Bila suatu *activity* baru dimulai, *activity* yang sebelumnya digunakan maka akan dipindahkan ketumpukan paling atas. Jika pengguna ingin menggunakan *activity* sebelumnya, cukup menekan tombol *Back*, atau menutup *activity* yang sedang digunakan, maka *activity* yang berada diatas akan aktif kembali. *Memory Manager* android menggunakan tumpukkan ini untuk menentukan prioritas aplikasi berdasarkan *activity*, memutuskan untuk mengakhiri suatu aplikasi dan mengambil sumber daya dari aplikasi tersebut. Ketika *activity* diambil dan disimpan dalam tumpukkan *activity* terdapat 4 kemungkinan kondisi transisi yang akan terjadi :

1. *Active*, setiap *activity* yang berada ditumpukan paling atas, maka dia akan terlihat, terfokus, dan menerima masukkan dari pengguna. Android akan berusaha untuk membuat *activity* aplikasi ini untuk untuk tetap hidup dengan segala cara, bahkan akan menghentikan *activity* yang berada dibawah tumpukkannya jika diperlukan. Ketika *activity* sedang aktif, maka yang lainnya akan dihentikan sementara.

2. *Paused*, dalam beberapa kasus *activity* akan terlihat tapi tidak terfokus pada kondisi inilah disebut *paused*. Keadaan ini terjadi jika *activity* transparan dan tidak *fullscreen* pada layar. Ketika *activity* dalam keadaan *paused*, dia terlihat *active* namun tidak dapat menerima masukkan dari pengguna. Dalam kasus ekstrim, android akan menghentikan *activity* dalam keadaan *paused* ini, untuk menunjang sumber daya bagi *activity* yang sedang aktif.

3. *Stopped*, ketika sebuah *activity* tidak terlihat, maka itulah yang disebut *stopped*. *Activity* akan tetap berada dalam memori dengan semua keadaan dan informasi yang ada. Namun akan menjadi kandidat utama untuk dieksekusi oleh system ketika membutuhkan sumberdaya lebih. Oleh karenanya ketika suatu *activity* dalam kondisi *stopped* maka perlu disimpan data dan kondisi antar muka saat itu. Karena ketika *activity* telah keluar atau ditutup, maka dia akan menjadi *inactive*.

4. *Inactive*, kondisi ketika *activity* telah dihentikan dan sebelum dijalankan. *Inactive activity* telah ditiadakan dari tumpukan *activity* sehingga perlu *restart* ulang agar dapat tampil dan digunakan kembali.

Kondisi transisi ini sepenuhnya ditangani oleh manajer memori android. Android akan memulai menutup aplikasi yang mengandung *activity* inactive, kemudian *stopped activity*, dan dalam kasus luar biasa *paused activity* juga akan di tutup.

2.1.6.2 *Services*

Suatu *service* tidak memiliki tampilan antarmuka, melainkan berjalan di *background* untuk waktu yang tidak terbatas. Komponen *service* diproses tidak terlihat, memperbarui sumber data dan menampilkan notifikasi. *Service* digunakan untuk melakukan pengolahan data yang perlu terus diproses, bahkan ketika *Activity* tidak aktif atau tidak tampak.

2.1.6.3 *Intents*

Intens merupakan sebuah mekanisme untuk menggambarkan tindakan tertentu, seperti memilih foto, menampilkan halaman web, dan lain sebagainya. *Intents* tidak selalu dimulai dengan menjalankan aplikasi, namun juga digunakan oleh sistem untuk memberitahukan ke aplikasi bila terjadi suatu hal, misal pesan masuk. *Intents* dapat eksplisit atau implisit, contohnya jika suatu aplikasi ingin menampilkan URL, sistem akan menentukan komponen apa yang dibutuhkan oleh *Intents* tersebut.

2.1.6.4 *Broadcast Receiver*

Broadcast Receivers merupakan komponen yang sebenarnya tidak melakukan apa-apa kecuali menerima dan bereaksi menyampaikan pemberitahuan. Sebagian besar *Broadcast* berasal dari sistem misalnya, Batre sudah hampir habis, informasi zona waktu telah berubah, atau pengguna telah merubah bahasa default pada perangkat. Sama halnya dengan *service*, *Broadcast Receivers* tidak menampilkan antarmuka pengguna. Namun, *Broadcast Receivers* dapat menggunakan *Notification Manager* untuk memberitahukan sesuatu kepada pengguna.

2.1.6.5 *Content Providers*

Content Providers digunakan untuk mengelola dan berbagi *database*. Data dapat disimpan dalam file sistem, dalam *database SQLite*, atau dengan cara lain yang pada prinsipnya sama. Dengan adanya *Content Provider* memungkinkan antar aplikasi untuk saling berbagi data. Komponen ini sangat berguna ketika sebuah aplikasi membutuhkan data dari aplikasi lain, sehingga mudah dalam penerapannya

2.1.7 Google Maps

Google Maps adalah layanan gratis yang diberikan oleh Google dan sangat popular. Google Maps adalah suatu peta dunia yang dapat kita gunakan untuk

melihat suatu daerah. Dengan kata lain, Google Maps merupakan suatu peta yang dapat dilihat dengan menggunakan suatu browser. Kita dapat menambahkan fitur Google Maps dalam web yang telah kita buat atau pada blog kita yang berbayar maupun gratis sekalipun dengan Google Maps API. Google Maps API adalah suatu library yang berbentuk JavaScript.

Cara membuat Google Maps untuk ditampilkan pada suatu web atau blog sangat mudah hanya dengan membutuhkan pengetahuan mengenai HTML serta JavaScript, serta koneksi Internet yang sangat stabil. Dengan menggunakan Google Maps API, kita dapat menghemat waktu dan biaya untuk membangun aplikasi peta digital yang handal, sehingga kita dapat fokus hanya pada data-data yang akan ditampilkan. Dengan kata lain, kita hanya membuat suatu data sedangkan peta yang akan ditampilkan adalah milik Google sehingga kita tidak dipusingkan dengan mambuat peta suatu lokasi, bahkan dunia[8].

2.1.7.1 Elemen Google Maps API

Beberapa elemen dan objek yang terdapat pada Google Maps API yang diterapkan di aplikasi:

1. *Markers*

   Marker adalah fungsionalitas simbol yang menandakan suatu titik koordinat lokasi. marker digunakan untuk menandai lokasi seperti klub olahraga dan kejuaraan olahraga pada peta Google Maps.

2. *Directions*

   Adalah *utility* yang menangani masalah pencarian rute. *Directions* ini digunakan untuk memandu rute jalan dari pengguna menuju tempat lokasi klub olahraga dan kejuaraan olahraga pada peta Google Maps.

3. *GeoCoding and Reverse GeoCoding*

   Adalah *utility* untuk memproses konversi alamat yang dapat digunakan untuk menempatkan penanda atau posisi peta.

4. *Location Manager*

Location Manager adalah API yang paling penting, dalam memberikan akses ke layanan sistem lokasi dengan pemetaan dan GPS. Dalam hal ini *Location Manager* berfungsi untuk mendapatkan update periodik lokasi geografis perangkat dengan memetakan nilai latitude dan longitude kedalam Google Maps

5. *Overlays Overview*

   Adalah objek di Google Maps yang terkait dengan koordinat lintang dan bujur, sehingga objek ini bergerak saat ditarik atau zoom pada peta. *Overlays* mencerminkan benda yang ada pada peta untuk menunjukan titik, garis, area, atau markers benda.

6. Loading Google Maps API koneksi script yang dibuat keserver Google Map API dengan menggunakan API key yang didapatkan pada saat mendaftar ke Google Maps API.

7. *Map DOM Elements*

   Untuk menampilkan map pada halaman web, umumnya menggunakan div tag untuk menampungnya. Diharuskan membuat sebuah div tag dengan nama "map_canvas", dan mereferensikan element ini ke *Document Object Model* (DOM).

Dalam pembutan program Google Map API menggunakan urutan sebagai berikut:

1. Memasukkan Maps API JavaScript ke dalam HTML kita.

2. Membuat element div dengan nama map_canvas untuk menampilkan peta

.3. Membuat beberapa objek literal untuk menyimpan property-properti pada peta.

4. Menuliskan fungsi JavaScript untuk membuat objek peta.

5. Meng-inisiasi peta dalam tag body HTML dengan event onload.

Pada Google Maps API terdapat 4 jenis pilihan model peta yang disediakan oleh Google, diantaranya adalah:

1. *ROADMAP*, untuk menampilkan peta biasa 2 dimensi

2. *SATELLITE*, untuk menampilkan foto satelit

3. *TERRAIN*, untuk menunjukkan relief fisik permukaan bumi dan menunjukkan seberapa tingginya suatu lokasi, contohnya akan menunjukkan gunung dan sungai

4. *HYBRID*, akan menunjukkan foto satelit yang diatasnya tergambar pula apa yang tampil pada *ROADMAP* (jalan dan nama kota)

2.1.8 *Web Service*

Web Service adalah sekumpulan application logic beserta objek-objek dan metode-metode yang dimilikinya yang terletak di suatu server yang terhubung ke internet sehingga dapat diakses menggunakan protocol HTTP dan SOAP (*Simple Object Access Protocol*). Dalam penggunaannya, web service dapat digunakan dari hanya untuk memeriksa data *user* yang login ke sebuah website ataupun untuk digunakan pada transaksi perbankan online yang rumit.

Tujuan dari teknologi ini adalah untuk memudahkan beberapa aplikasi atau komponennya untuk saling berhubungan dengan aplikasi lain dalam sebuah organisasi maupun diluar organisasi menggunakan standar yang tidak terikat *plat*form (*plat*form-*neutral*) dan tidak terikat akan bahasa pemrograman yang digunakan (language-neutral). Hal tersebut dapat terjadi karena penggunaan XML standar yang didukung oleh banyak perusahaan besar di dunia, yang digunakan untuk bertukar data. Selain daripada itu, penggunaan SOAP menjadikan metode-metode dari objek-objek yang ada dalam sebuah *web service* dapat di akses dari aplikasi lain seperti halnya aplikasi tersebut mengakses metode *local*

2.1.9 JSON

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 Pemprograman JavaScript, Standar ECMA-262 Edisi ke-3 – Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python

dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

JSON terbuat dari dua struktur:

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 pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut:

1. Objek

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).

2. Larik

Larik adalah kumpulan nilai yang terurutkan. Larik dimulai dengan [(kurung kotak buka) dan diakhiri dengan] (kurung kotak tutup). Setiap nilai dipisahkan oleh , (koma).

3. Nilai

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.

4. String

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.

5. Angka

Angka adalah sangat mirip dengan angka di C atau Java, kecuali format oktal dan heksadesimal tidak digunakan[8].

2.1.10 Unified Modelling Language (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem[9].

2.1.10.1 *Use Case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem.

2.1.10.2 *Class Diagram*

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi). Class diagram menggambarkan struktur dan deskripsi class, package dan objek

beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain

2.1.10.3 *Behavior Diagram*

Behavior diagram dapat dikelompokan menjadi tiga diagram, yaitu :

a. Statechart Diagram

Statechart Diagram berfungsi untuk memodelkan prilaku dinamis satu kelas satu objek.

b. *Activity* Diagram

Activity Diagram memodelkan alur kerja (work flow) sebuah proses bisnis dan urutan aktifitas dalam suatu proses.

c. Interaction Diagram

Interaction Diagram dibagi menjadi dua model diagram yaitu :

1. Sequence Diagram menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram ini secara khusus bersosialisasi dengan use case. Sequence diagram, memperlihatkan tahap demi tahap apa yang seharusnya terjadi untuk menghasilkan sesuatu dalam use case.

2. Colaboration Diagram melihat pada interaksi dan hubungan terstruktur antar objek. Tipe diagram ini menekankan pada hubungan (relationship) antar objek, sedangkan sequence diagram menekankan pada urutan kejadian. Dalam collaboration diagram terdapat beberapa objek, link, dan message.

2.1.11 Eclipse

Eclipse adalah sebuah IDE (Integrated Development Environment) untuk mengembangkan perangkat lunak dan dapat dijalankan di semua platform (platform-independent). Berikut ini adalah sifat dari Eclipse:

a. Multi-platform: Target sistem operasi Eclipse adalah Microsoft Windows, Linux, Solaris, AIX, HP-UX dan Mac OS X.

b. Mulit-language: Eclipse dikembangkan dengan bahasa pemrograman Java, akan tetapi Eclipse mendukung pengembangan aplikasi berbasis bahasa pemrograman lainnya, seperti C/C++, Cobol, Python, Perl, PHP, dan lain sebagainya.

c. Multi-role: Selain sebagai IDE untuk pengembangan aplikasi, Eclipse pun bisa digunakan untuk aktivitas dalam siklus pengembangan perangkat lunak, seperti dokumentasi, test perangkat lunak, pengembangan web, dan lain sebagainya.

Eclipse pada saat ini merupakan salah satu IDE favorit dikarenakan gratis dan open source, yang berarti setiap orang boleh melihat kode pemrograman perangkat lunak ini. Selain itu, kelebihan dari Eclipse yang membuatnya popular adalah kemampuannya untuk dapat dikembangkan oleh pengguna dengan komponen yang dinamakan plug-in.

2.1.11.1 Arsitektur Eclipse

Sejak versi 3.0, Eclipse pada dasarnya merupakan sebuah kernel, yang mengangkat plug-in. Apa yang dapat digunakan di dalam Eclipse sebenarnya adalah fungsi dari plug-in yang sudah diinstal. Ini merupakan basis dari Eclipse yang dinamakan Rich Client Platform (RCP). Berikut ini adalah komponen yang membentuk RCP:

a. Core platform

b. OSGi

c. SWT (Standard Widget Toolkit)

d. JFace

e. Eclipse Workbench

Secara standar Eclipse selalu dilengkapi dengan JDT (*Java Development Tools*), plug-in yang membuat Eclipse kompatibel untuk mengembangkan program Java, dan PDE (*Plug-in Development Environment*) untuk

mengembangkan plug-in baru. Eclipse beserta plug-in-nya diimplementasikan dalam bahasa pemrograman Java.

Konsep Eclipse adalah IDE yang terbuka (*open*), mudah diperluas (*extensible*) untuk apa saja, dan tidak untuk sesuatu yang spesifik. Jadi, Eclipse tidak saja untuk mengembangkan program Java, akan tetapi dapat digunakan untuk berbagai macam keperluan, cukup dengan menginstal plug-in yang dibutuhkan. Apabila ingin mengembangkan program C/C++ terdapat plug-in CDT (C/C++ Development Tools).

Selain itu, pengembangan secara visual bukan hal yang tidak mungkin oleh Eclipse, plug-in UML2 tersedia untuk membuat diagram UML. Dengan menggunakan PDE setiap orang bisa membuat plug-in sesuai dengan keinginannya. Salah satu situs yang menawarkan plug-in secara gratis seperti Eclipse *downloads by project*.

2.1.12 Software Development Kit (SDK)

Software Development Kit (SDK) adalah suatu kit atau library dari bahasa pemrograman untuk pengembangan atau pembangunan suatu perangkat lunak dan biasanya SDK terdiri dari kumpulan tools yang dibutuhkan. Misalnya bahasa pemrograman java, mempunyai SDK yang berisi suatu library yang dapat digunakan untuk membuat suatu aplikasi berbasis java.[11]

2.1.13 PHP

PHP (*Hypertext Preprocessor*), merupakan bahasa pemograman web yang bersifat serverside. Hal tersebut mengandung arti bahwa PHP merupakan bahasa berbentuk scriptyang disimpan dan dijalankan di komputer server (*web server*) sedang hasilnya dikirimkan ke komputer client (*web browser*) dalam bentuk script HTML. Keuntungan penggunaan PHP, kode yang menyusun program tidak perlu dibagikan ke pemakai, yang berarti kerahasiaan kode dapat dilindungi.

Penggunaan PHP memungkinkan web dapat dibuat dinamis sehingga perawatan situs web tersebut menjadi lebih mudah dan efisien. Hal menarik yang didukung PHP adalah kenyataan bahwa PHP bisa digunakan untuk mengakses

berbagai *database* seperti *Access*, Oracle, MY\ySQL, dan lain-lain. PHP merupakan perangkat lunak *Open Source* yang disebarkan dan dilisensikan secara gratis serta dapat diunduh secara bebas dari situs resminya.

Fungsi yang dimiliki PHP sangat lengkap sehingga dapat dikatakan tidak perlu membuat fungsi sendiri. Hal tersebut dikarenakan daftar fungsi PHP yang lengkap menjadikan baris perintah semakin efisien. Selain itu juga, seperti yang telah disebutkan sebelumnya bahwa kelebihan PHP adaah mampu berintegrasi dengan berbagai macam database. Salah satu database yang selalu diintegrasikan adalah MySQL.