BAB II

LANDASAN TEORI

II.1. Sistem Informasi

Sistem informasi bukan merupakan hal yang baru, yang baru adalah komputerisasinya. Sebelum ada komputer, teknik penyaluran informasi yang memungkinkan manajer merencanakan serta mengendalikan operasi telah ada. Komputer menambahkan satu atau dua dimensi, seperti kecepatan, ketelitian dan penyediaan data dengan volume yang lebih besar yang memberikan bahan pertimbangan yang lebih banyak untuk mengambil keputusan.

Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi untuk dapat menyediakan laporan-laporan yang diperlukan oleh pihak luar tertentu (Tata Sutabri, 2012 : 38).

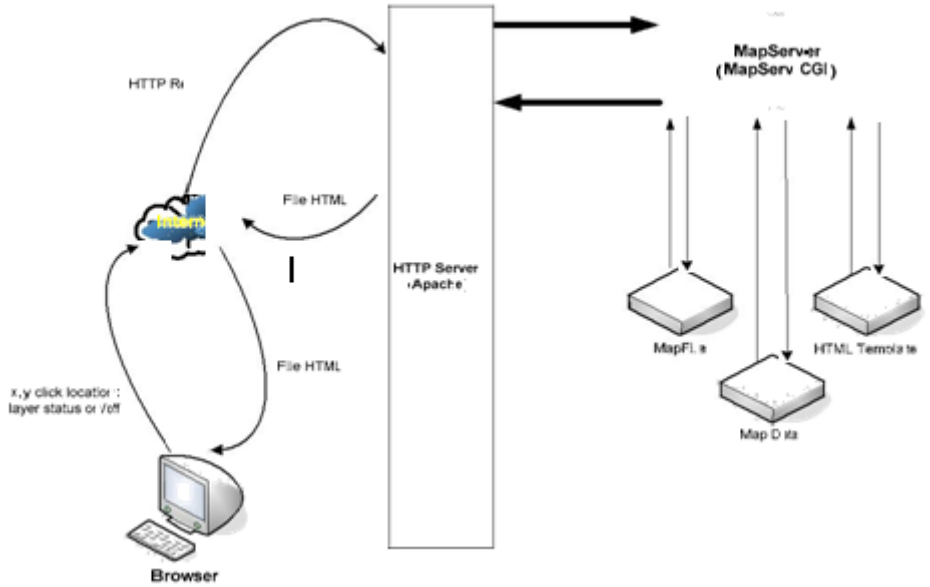
II.2. Sistem Informasi Geografis

Sistem Informasi Geografis (SIG) atau *Geographic Informasi System* (GIS) adalah sebuah sistem yang didesain untuk menangkap, menyimpan, memanipulasi, menganalisa, mengatur dan menampilkan seluruh jenis data geografis. Akronim GIS terkadang dipakai sebagai istilah untuk *geographical information scirnce* atau *geospatial information studies* yang merupakan ilmu studi atau pekerjaan yang berhubungan dengan *Geographic Information System*. Dalam artian sederhana sistem informasi geografis dapat kita simpulkan sebagai

gabungan kartografi, analisis statistic dan teknologi sistem basis data (database). (Edy Irwansyah ; 2013 : 1).

II.2.1. Arsitektur Sistem

Secara umum diagram pengembangan sistem dimulai dengan user melakukan action pada web browser yaitu meminta layanan ke web browser yaitu meminta layanan mapserver ke webserver dan diterjemahkan dalam CGI yang diteruskan oleh mapscript untuk meminta Dataset GIS pada program external sehingga didapatkan output yang ditampilkan pada web browser. Sistem aplikasi yang dibangun dengan MapServer memiliki arsitektur yang ditunjukkan pada Gambar I.1 (Hamidi ; 2013 : 3).

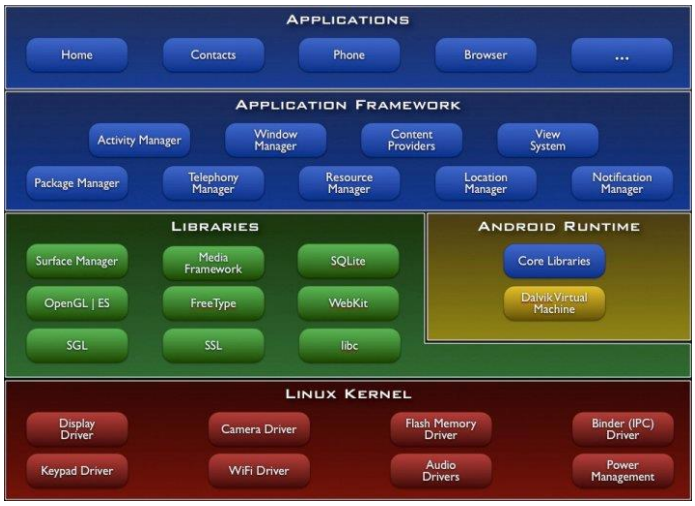


Gambar II.1. Arsitektur Sistem
(*Sumber : Hamidi ; 2013 : 3*)

II.3. Android

Android adalah sistem operasi berbasis kernel linux. Google mengibaratkan Android sebagai tumpukan software dimana setiap tumpukan berisi program yang mendukung fungsi spesifik dari sistem operasi. Adapun susunan lapisan tersebut dari bawah ke atas adalah sebagai berikut :

1. Linux sebagai kernel.
2. Android runtime dan libraries berisi Dalvik Virtual Machine dan kode-kode librari dalam bahasa C/C++.
3. Aplication framework berisi program untuk mengatur fungsi-fungsi dasar smartphone.
4. Application.



Gambar II.2. Lapisan Sistem Operasi Android
(*Sumber : Muhammad Athoillah ; 2013 : 2*)

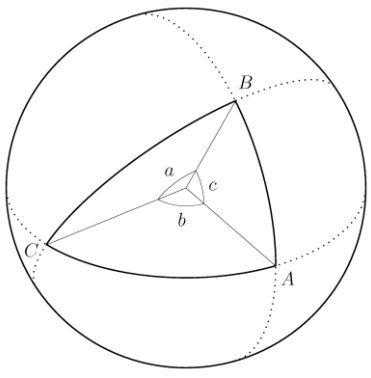
Android diciptakan oleh sebuah perusahaan kecil bernama Android Inc pada tahun 2000, yang kemudian perusahaan tersebut dibeli oleh Google Inc, untuk mengembangkan android lebih lanjut, dibentuklah Open Handset Alliance

(OHA) yang terdiri dari 34 perusahaan sofware, hardware dan telekomunikasi diantaranya yaitu Google, HTC, Intel, Motorola, T-Mobile dll.

Sampai saat ini Android telah banyak berkembang hingga beberapa versi. Android versi 1.1 adalah versi pertama yang dirilis pada tahun 2009, kemudian berturut – turut muncul versi yang lain yang merupakan perbaikan demi perbaikan dari versi yang sebelumnya diantaranya ialah versi 1.5 (Cupcake), versi 1.6 (Donut), versi 2.0/2.1 (Eclair), versi 2.2 (Froyo), versi 2.3 (Gingerbread), versi 3.0/3.1 (Honeycomb), versi 4.0 (ICS), hingga yang terbaru saat ini adalah versi 4.1 (Jellybean) (Muhammad Athoillah ; 2013 : 2).

II.4. Spherical Law of Cosines

Spherical Law of Cosines merupakan salah satu persamaan dasar dari spherical triangle. Salah satu pengaplikasian dari Spherical Law of Cosines adalah mengkalkulasi jarak diantara dua titik diatas permukaan Bumi. Untuk mengetahui bagaimana Spherical Law of Cosines digunakan perhatikan gambar dibawah



Gambar II.3. : Spherical triangle

Gambar diatas merupakan spherical triangle dengan titik A, B, C dan sisi melengkung a, b, dan c. Sisi melengkung tersebut merupakan jarak geodetik yang

bisa diketahui jaraknya. Apabila (lat1,long1) dan (lat2,long2) merupakan koordinat geografis dari titik B dan C, maka bisa didapat nilai b = $\pi/2 - \theta 1$, c = $\pi/2 - \theta 2$, and A = $\lambda 2 - \lambda 1$. Kemudian untuk mencari jarak antara B dan C dengan menggunakan rumus Spherical Law of Cosines,ekspresinya adalah sebagai berikut :

$$d = \text{acos}(\sin \theta 1 \sin \theta 2 + \cos \theta 1 \cos \theta 2 \cos A).R \quad \text{.....(1)}$$

$$d = \text{acos}(\sin \theta 1 \sin \theta 2 + \cos \theta 1 \cos \theta 2 \cos(\lambda 2 - \lambda 1)).R \quad \text{.....(2)}$$

d = acos (sin(lat1).sin(lat2) + cos (lat1).cos(lat2).cos(long2-long1).R (3)

Keterangan :

- *d* adalah jarak antara dua point
- lat, θ adalah latitude
- long, λ adaah longitude
- R adalah radius dari lingkaran bola (R = 637.100 : radius Bumi dalam meter) (Nurul Hikmah Agustin ; 2015 : 11)

II.5. Pengertian Database

D*atabase* adalah sekumpulan *file* data yang saling berhubungan dan diorganisasi sedemikian rupa sehingga memudahkan untuk mendapat dan memproses data. Lingkungan sistem *database* menekankan data yang tidak tergantung (*idenpendent data*) pada aplikasi yang akan menggunakan data. Data adalah kumpulan fakta dasar (mentah) yang terpisah.

Sebuah *database* harus dibuat dengan rapi agar data yang dimasukkan sesuai dengan tempatnya. Sebagai contoh, di sebuah perpustakaan, penyimpanan buku dikelompokkan berdasar jenis atau kategori-kategori tertentu, misalnya kategori buku komputer, buku pertanian, dan lain-lain. Kemudian dikelompokkan lagi berdasarkan abjad judul buku, ini dilakukan agar setiap pengunjung dapat dengan mudah mencari dan mendapatkan buku yang dimaksud (Wahana Komputer ; 2010 ; 1).

II.6. *MySQL*

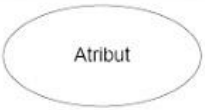
MySQL adalah suatu sistem manajemen basis data relasional (RDBMS-*Relational Database Management System*) yang mampu bekerja dengan cepat, kokoh, dan mudah digunakan. Contoh RDBMS lain adalah *Oracle, Sybase*. Basis data memungkinkan anda untuk menyimpan, menelusuri, menurutkan dan mengambil data secara efesien. *Server* MySQL yang akan membantu melakukan fungsionaliitas tersebut. Bahasa yang digunakan oleh MySQL tentu saja adalah SQL-*standar* bahasa basis data relasional di seluruh dunia saat ini.

MySQL dikembangkan, dipasarkan dan disokong oleh sebuah perusahaan Swedia bernama MySQL AB. RDBMS ini berada di bawah bendera GNU GPL sehingga termasuk produk *Open Source* dan sekaligus memiliki lisensi komersial. Apabila menggunakan MySQL sebagai basis data dalam suatu situs Web. Anda tidak perlu membayar, akan tetapi jika ingin membuat produk RDBMS baru dengan basis MySQL dan kemudian menjualnya, anda wajib bertemu mudah dengan lisensi komersial (Antonius Nugraha Widhi Pratama ; 2010 : 10).

II.7. Entity Relationship Diagram (ERD)

Entity Relationship Diagram atau ERD merupakan salah satu alat (tool) berbentuk grafis yang populer untuk *desain database.* Tool ini relatif lebih mudah dibandingkan dengan Normalisasi. Kebanyakan sistem analis memakai alat ini, tetapi yang jadi masalah, kalau di cermati secara seksama, tool ini mencapai 2NF (Yuniar Supardi, 2010 : 448).

Tabel II.1. Simbol ERD

Notasi	Keterangan
	**Entitas**, adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai.
	**Relasi**, menunjukkan adanya hubungan di antara sejumlah entitas yang berbeda.
	**Atribut**, berfungsi mendeskripsikan karakter entitas (atribut yg berfungsi sebagai key diberi garis bawah)
	Garis, sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

(*Sumber : Yuniar Supardi ; 2010 : 448*)

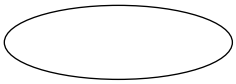
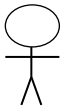
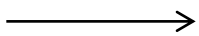
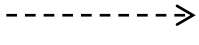
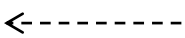
II.8. *UML*

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Usecase* Diagram

Usecase diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Usecase* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *usecase* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *usecase* diagram, yaitu :

Tabel II.2. Simbol *UseCase*

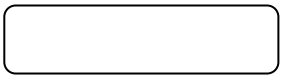
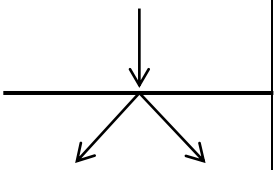
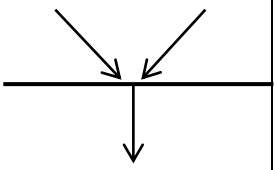
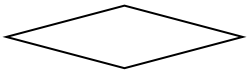
Gambar	Keterangan
	*Usecase* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *usecase*.
	Aktor adalah *abstraction*dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *usecase*, tetapi tidak memiliki control terhadap *usecase*.
	Asosiasi antara aktor dan *usecase*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *usecase* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	*Include*, merupakan di dalam *usecase* lain (*required*) atau pemanggilan *usecase* oleh *usecase* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *usecase* lain jika kondisi atau syarat terpenuhi.

(*Sumber : WinduGata ; 2013 : 4*)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow*(aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu :

Tabel II.3. Simbol *Activity Diagram*

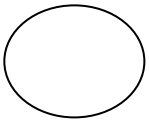
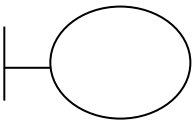
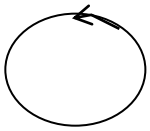
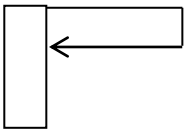
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*Endpoint*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork*(Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*DecisionPoints*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
	*Swimlane*, pembagian *activity*diagram untuk menunjukkan siapa melakukan apa.

(*Sumber : WinduGata ; 2013 : 6*)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *usecase* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.4. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*BoundaryClass*, berisi kumpulan kelas yang menjadi *interface*atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form*cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation*mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline*terdapat *activation*.

(*Sumber : WinduGata ; 2013 : 7*)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.5. *MultiplicityClass Diagram*

Multiplicity	**Penjelasan**
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(*Sumber : WinduGata ; 2013 : 9*)