**BAB II**

**LANDASAN TEORI**

1. **Sistem Informasi**

Sistem adalah sekelompok elemen – elemen yang terintegrasi dengan maksud yang sama untuk mencapai suatu tujuan(Leod, 1995). Dan informasi adalah data yang telah diproses, atau data yang memiliki arti(Leod, 1995). Kemudian Kertahadi(1995) mendefinisikan sebagai suatu alat yang menyajikan informasi dengan cara sedemikian rupa sehingga bermanfaat bagi penerimanya. Sementara sistem informasi terdiri dari komponen – komponen yang mendukungnya dengan istilah *Building Block* (blok bangunan) antara lain:

1. Blok masukan (*Input Block*);
2. Blok model (*Model Block*);
3. Blok keluaran (*Output Block*);
4. Blok teknologi (*Technology Block*);
5. Blok kendali (*Control Block*).
6. **Sistem Informasi Geografis (SIG)**

Sistem Informasi Geografis (SIG) adalah suatu sistem berbasis komputer yang memiliki kemampuan dalam .

1. **Global Positioning System (GPS)**

*Global Positioning System* (GPS) adalah suatu sistem radio navigasi penentuan posisi menggunakan satelit. GPS dapat memberikan posisi suatu objek di muka bumi dengan akurat dan cepat (koordinat tiga dimensi x, y, z) dan memberikan informasi waktu serta kecepatan bergerk secara kontinyu di seluruh dunia (Riyanto, 2010).

1. **GoogleMaps**

*Google Maps* adalah layanan gratis yang diberikan oleh Google dan sangat popular untuk pemetaan digital berbasiskan web. Aplikasi ini diperkenalkan pada Februari 2005 dan merupakan revolusi penyajian peta dalam bentuk digital. Saat ini *Google* *Maps* untuk perangkat android telah mencapai *Google* *Maps* v3, yang tentu saja berbeda dengan sebelumnya yatu *Google* *Maps* v1 dan *Google* *Maps* v2. *Google* *Maps* v3 ini diluncurkan pada Maret 2013 dan menawarkan lebih banyak fungsionalitas dari pada versi sebelumnya seperti pembuatan peta 3D (Google, 2005).

*Google Maps* menyediakan layanan berupa pemetaan jalan, rute, dan navigasi untuk berbagai rute perjalanan sehingga dapat digunakan untuk mempercepat pencarian sebuah lokasi dalam waktu yang singkat, dan juga mununjukkan jalan mana saja yang harus ditempuh untuk mencapai tujuan. Teknologi *Google* *Maps* menggunakan citra satelit untuk melakukan pemetaan objek yang ada dipermukaan bumi secara *realtime*, dalam hal ini peta yang ada pada *Google* *Maps* diupdate dalam kurun waktu tertentu.

*Google Maps* dapat ditambahkan dalam web, blog maupun aplikasi *mobile* dengan menggunakan *Google Maps.* Dalam implementasinya Google Maps juga dapat disesuaikan dengan kebutuhan aplikasi yang akan dibuat (Google, 2005).

1. **Google Maps APIs**

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 terlihat 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 dpat fokus hanya pada data-data yang akan ditampilkan. Dengan kata lain, kita hanya membuat suatu data sedangkan peta yang akan ditampilkan adalah milih google sehingga kita tidak dipusingkan dengan membuat peta suatu lokasi, bahkan dunia.

Dalam pembuatan program google maps 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. Menginstall peta dalam tag body html dengan even onload.

Pada google maps api terdapat 4 jenis pilihan model peta yang disediakan oleh google, di antaranya adalah :

1. Roadmap diguakan untuk menampilkan peta biasa 2 dimensi.
2. Satellite, untuk menampilkan foto satelit.
3. Terrain, untuk meunjukkan relief fisik permukaan bumid an meunjukkan seberapa tingginya suatu lokasi, contohnya akan meunjukkan gunung dan sungai.hybrid, akan menunjukkan foto satelit yang di atasnya tergambar pula apa yang tampil pada roadmap.
4. **Pariwisata**

Munurut undang-undang Nomor 9 Tahun 1990 tentang Kepariwisataan BAB I Pasal 1 dinyatakan bahwa wisata adalah kegiatan perjalanan atau sebagian dari kegiatan tersebut yang dilakukan secara sukarela serta bersifat sementara untuk menikmati obyek dan daya tarik wisata. Fasilitas yang dikuasai oleh negara/pemerintah termasuk tempat wisata dibiayai oleh APBN atau APBD dan atau atas swasembada kerelaan masyarakat dibawah pengurusan lembaga-lembaga negara dalam arti yang luas, tidak termasuk barang ata kekayaan yang dimiliki oleh BUMN/BUMD, yang pemanfaatannya ditujukan secara khusus untuk kegiatan-kegiatan yang berhubungan dengan penyelenggaraan negara dan pemerintahan serta kepentingan seluruh masyarakat. (Kepmen Keuangan No. 225/MK/V/4/1971 tentang Pedoman Pelaksanaan Inventarisasi Barang-barang Milik Negara).

Tempat wisata merupakan aset yang dikelola oleh negara, dibiayai sebagian atau seluruhnya oleh anggaran dan belanja negara yang pemakaiannya atau peruntukkannya oleh pemerintah atau negara (*bestemming* atau *bestimmung*) serta bagi umum.

Secara garis besar, pariwisata atau objek wisata dapat di kelompokkan menjadi tiga kategori sebagai berikut :

1. Wisata Alam

Yaitu objek wisata yang berpotensi pada sisi alamnya, baik potensi alam yang bersifat alamiah atau asli maupun buatan manusia. Misalkan Pantai Boom, Goa Kancing, Pemandian Bektiharjo.

1. Wisata Sejarah

Wisata sejarah adalah sebuah perjalanan yang dilakukan pada tempat-tempat yang memiliki nilah sejarah, seperti Museum Kambang Putih, Makam Ronggolawe.

1. Wisata Budaya

Yaitu objek wisata yang berpotensi pada sisi peninggalan budaya dan peninggalan sejarah. Objek wisata budaya dan sejarah biasanya berbentuk suatu benda yang merupakan peninggalan dari masa lampau yang kemudian dikumpulkan dan dipelihara keasliannya pada suatu ruangan yang memang difungsikan sebagai cagar budaya dan rumah sejarah. Misalnya Batik Gedog, Legen.

1. Wisata Religi

Yaitu objek wisata yang berlatar belakang keagamaan. Objek wisata religi biasanya merupakan penginggalan benda, tempat atau kisah yang dibukukan yang kemudian dikumpulkan dan dipelihara. Misalnya Klenteng Kwan Sing Bio, Makam Sunan Bonang, Masjid Agung Tuban.

1. Wisata Kuliner

Wisata kuliner adalah perjalan ke tempat tertentu untuk menikmati jenis masakan khas suatu daerah yang unik, seperti Ayam Bakar Wong Solo, KFC Tuban, Resto Kayu Manis.

1. **Bahasa Pemrograman Java**

Java merupakan bahasa berorientasi objek dan serbaguna. Kode Java dikompilasi dalam format yang disebut *bytecode*. *Bytecode* ini dapat dijalankan di semua komputer yang telah dilengkapi dengan program *Java Interpreter* dan *Java Virtual Machine*.

Java menyediakan sarana untuk membuat program (*applet*) yang berjalan pada *web* *browser*. Bahasa ini juga mendukung koneksi ke database, menyediakan sarana untuk membuat aplikasi berbasis Windows dan juga dapat dipakai untuk pemrograman jaringan (Gosling, 1994).

1. **Android**

Android merupakan subset perangkat lunak untuk perangkat *mobile* yang meliputi sistem operasi, *middleware* dan aplikasi inti yang dirilis oleh *Google*. Sedangkan Android SDK (*Software Development Kit*) menyediakan *Tools* dan API yang diperlukan untuk mengembangkan aplikasi pada *platform* Android dengan menggunakan bahasa pemrograman Java.

Dikembangkan bersama antara *Google*, HTC, Intel, Motorola, Qualcomm, T-Mobile, Nvidia yang tergabung dalam OHA (*Open Handset Alliance*) dengan tujuan membuat sebuah standar terbuka untuk perangkat bergerak (*mobile device*).

Featuresyang tersedia pada *platform* Android saat ini antara lain :

1. Framework Aplikasi yang mendukung penggantian komponen dan

*reusable*.

1. Mesin *virtual Dalvik* berjalan di atas *Linux kernel* yang dioptimalkan untuk perangkat *mobile*.
2. *Integrated browser* berdasarkan *open source engine WebKit*.
3. Grafis yang dioptimalkan dan didukung oleh *library* grafis 2D yang terkustomisasi, grafis 3D berdasarkan spesifikasi openGL ES 1.0 (OpsionalAkselerasi *Hardware*).
4. SQLite untuk penyimpanan data.
5. Media *support* yang mendukung audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
6. GSM Telephoning(tergantung *hardware)*
7. Bluetooth*,* EDGE, 3G dan WiFi (tergantung hardware).
8. Dukungan perangkat tambahan android dapat memanfaatkan kamera, layar sentuh, *accelerometers, magnetometers,* GPS, akselerasi 2D (dengan perangkat orentasi, *scalling,* konversi format piksel) dan akselerasi grafis 3D.
9. Multi*-touch,* kemampuan layaknya *handset modern* yang dapat menggunakan dua jari atau lebih untuk berinteraksi dengan perangkat.
10. Lingkungan Developmentyang lengkap dan kaya termasuk perangkat emulator, *tools* untuk *debugging,* profil dan kinerja memori dan plugin untuk Eclipse IDE.
11. Market, seperti kebanyakan *Handphone* yang memiliki tempat penjualan aplikasi, market pada android merupakan katalog aplikasi yang dapat didownload dan diinstall pada *Handphone* melalui internet.
12. **Versi Android**

Sistem operasi android atau [os android](https://www.riaume.com/daftar-nama-os-android-menurut-versi.html) terdiri dari beberapa versi ,setiap versi android terbaru memiliki nama-nama unik tersendiri dan memiliki beberapa jenis kelebihan mulai dari tampilan hingga optomasi keamana,berikut daftar nama [“os android”](https://www.riaume.com/daftar-nama-os-android-menurut-versi.html) menurut versi .mulai dari nama os android pertama kali di keluarkan sampai dengan os android versi terbaru yang baru di keluarkan di tahun 2016 ini :

* 1. OS android pertama versi 1.0 di rilis pada tanggal 23 bulan September tahun 2008 dan android versi 1.1 Dirilis pada tanggal 9 bulan Februari tahun 2009 di namakan dengan Android.
  2. OS android versi 1.5 Dirilis pertama pada tanggal 30 bulan April tahun 2009 di namakan dengan Android Cupcake.
  3. OS android versi 1.6 Dirilis pada tanggal 15 bulan September tahun 2009 di namakan dengan Android Donut.
  4. OS android versi 2.0 Dirilis pada tanggal 26 bulan Oktober tahun 2009 dan beberapa versi lainya seperti android v2.0.1 dan android V2.1 di namakan dengan Android Eclair
  5. OS android versi 2.2 dan versi v2.2.3 Dirilis pada tanggal 10 bulan mei tahun 2010 di namakan dengan Android Froyo
  6. OS android versi 2.3 sampai dengan versi v2.3.7 Dirilis pertama pada tanggal 6 bulan Desember tahun 2010 di namakan dengan Android Gingerbread . android Gingerbread menjadi salah satu os android tersukses di tahun 2010 , hingga saat ini masih banyak sekali ponsel android yang menggunakan versi ini.
  7. OS android versi v3.0 sampai dengan versi v3.2 Dirilis pertama pada tanggal 22 bulan Februari tahun 2011 di namakan dengan Android Honeycomb ,os android Honeycomb merupakan os android paling sempurna di bandingkan dengan os android versi sebelumnya,os ini terkenal akan kinerja os yang sangat baik dan memiliki tampilan mewah.jadi tidak heran apabila di tahun 2011 ponsel yang menggunakan android Honeycomb yang paling banyak di cari.
  8. OS android versi v4.0 sampai dengan v4.0.2 Dirilis pertama pada tanggal 19 bulan Oktober tahun 2011 di namakan dengan Android Ice Cream Sandwich . ini merupakan versi Os Android tersukses dan paling banyak cari,tampilan pada Android Ice Cream Sandwich sudah tidak terlalu jauh dengan android jelly bean tersebut,dan sampai saat ini android Ice Cream Sandwich masih banyak di gunakan pada ponsel kelas menengah kebawah dan beberapa samsung galaxy tab terdahulu.kelebihan utama dari android ice crean sandwich ini adalah sudah mendukung Flash player sehingga bisa memudahkan kita browsing di internat dll
  9. OS android versi v4.1 sampai dengan v4.3 Dirilis pertama pada tanggal 9 bulan juli tahun 2012 di namakan dengan Android Jelly Bean.
  10. Android Jelly Bean merupakan salah satu os yang di perbaharui dari versi ice cream sandwich,dari segi tampilan os android jelly bean paling sempurna di banding beberapa os terdahulu,tidak hanya memiliki tampilan yang lebih dinamis ,android jelly bean juga telah di optimasi pada bagian kinerja os super cepat.di tambah dengan beberapa peningkatan kinerja pada sisi kamera pada smartphone dan tablet pc,penambahan navigasi Gesture dan banyak lagi,sampai di tahun 2014 ini android jelly bean masih menjadi salah satu os yang paling banyak di gunakan dan masih banyak sekali ponsel terbaru android yang masih menggunakan os android jelly bean satu ini,
  11. OS android versi v4.4 sampai dengan v4.4.4 Dirilis pertama pada tanggal 31 bulan Oktober tahun 2013 di namakan dengan Android kitkat .os android kitkat memiliki tampilan 100% lebih dinamis dan berbeda total dengan android jelly bean,android kitkat di optimasi pada sisi konsumsi baterai dan kinerja os lebih cepat ketika di jalankan pada perangkat memiliki spesifikasi lebih rendah,seperti kita tahu jika andorid jelly bean memiliki kelebihan pada sisi konsumsi baterai yang lebih tinggi dan ketika di jalankan di perangkat yang memiliki versi rendah os ini tidak maksimal.
  12. OS android versi v5.0 dirilis pertama pada tahun 2014 dikenal dengan nama “Android Lollipop”, android lollipop memiliki peningkatan sistem keamanan dan tampilan lebih dinamis ,os versi terbaru dari android ini sepertinya bisa di gunakan berbagai perangkat elektronik seperti tv dll.
  13. OS Android versi 6.0 bernama Marshmallow di kenal juga dengan nama Android M ,os android Marshmallow terbaru ini merupakan pemuktakhiran untuk os android lollipop v5.0 sebelum nya.dari segi tampilan os android versi 6.0 ini tidak terlalu jauh berbeda di banding dengan os android lollipop sebelumnya, hanya saja ada terdapat pembaharuan keamanan atau tingkat keamanan os android 6.0 ini di tingkatkan dan di tambah beberapa fitur unggulan lainnya.

OS Android versi 7.0 bernama Nougat. Pada versi ini Google telah menyempurnakan OS ini menjadi sangat nyaman dan mudah digunakan. Diantara fitur yang ditambahkan dari versi sebelumnya adalah : *direct reply notification, multi window support, quick setting menu,* dan *doze mode* 2.0*.*

1. **Arsitektur Android**

Berikut ini adalah arsitektur sistem operasi android :

1. Linux Kernel

Android bukan Linux, akan tetapi android dibangun diatas *Linux Kernel* yaitu versi 2.6 sehingga kehandalannya bisa dipercaya. Untuk inti sistem servis linux yang digunakan seperti keamanan, manajemen memori, proses manajemen, *network*, dan *driver* model. Seperti yang terlihat di gambar, linux kernel menyediakan Driver Layar, Kamera, Keypad, WiFi, Flash Memory, Audio dan IPC *(Interprocess Communication)* untuk mengatur aplikasi dan keamanan. Kernel juga bertidak sebagai lapisan antara hardware dan software stacknya.

1. Libraries

Android menyertakan libraries C/C++ yang digunakan oleh berbagai komponen dari sisterm Android. Kemampuan ini disediakan kepada Developer aplikasi melalui Framework Aplikasi Android. Beberapa init libraries tercantum di bawah ini:

1. System C *Library* – Variasi dari Implementasi BSD-berasal pelaksanaan system standar C *library* *(libc),* sesuai untuk perangkat *embedded* berbasis Linux.
2. Media *Libraries* – *PacketVideo* berdasarkan *OperCORE*; *library* video, serta file gambar, termasuk MPEG4, H.264, AAC, AMR, JPG dan PNG
3. *Surface* *Manager* – Mengelola akses ke subsistem layar, lapisan komposid 2D dan grafis 3D dari beberapa aplikasi.
4. *LibWebCore*–Mesin web modern yang *powerfull* yang baik browser Android *embedded* web.
5. SGL – Mesi grafis 2D.
6. 3D *Libraries* – sebuah pelaksanaan berdasarkan opengl ES 1.0 API ; perpustakaan baik menggunakan perangkat keras akselerasi 3D (apabila tersedia) atau yang disertakan, sangat optimal 3D *software* *resterizer*.
7. *FreeType* – Bitmap dan vector font rendering.
8. *SQLite* – Mesin database yangkuat dan ringan, dan penghubung tersedia untuk semua aplikasi.
9. Android**-**Runtime

Android terdirid aru satu set core libraries yang menyediakan sebagian besar fungsi yang sama denga yang terdapat pada core libraries baha pemrograman Java. Setiap aplikasi menjalankan prosesnya sendir dalam Android, dengan masing-masing instan dari virtual Dalvik *(Dalvik VM).* Dalvik dirancang agar perangkat dapat menjalankan multiple VMs secara efisien.

Mesin virtual Dalvik mengeksekusi file dalam Dalvik executable *(.dex),* sebuah format yang dioptimalkan untuk memori yang kecil. Delvik VM berbasis, berjalan dan dikompilasi oleh compiler Bahasa Java yang telah ditransformasikan ke dalam .dex format oleh tool ‘’dx’’ yang telah disertakan. Dalvik VM bergantung pada kernel Linux untuk berfungsi, seperti threading dan manajemen memori tingkat rendahnya.

1. Framework-Aplikasi

Pengembang memiliki akses penuh menuju API framework yang sama, dengan yang digunakan oleh aplikasi inti. Arsitektur aplikasi dirancang agar komponen dapat digunakan kembali (*reuse*) dengan mudah. Setiap aplikasi dapat memanfaatkan kemampuan ini dan aplikasi yang lain mungkin akan memanfaatkan kemampuan ini (sesuai dengan bantuan keamanan yang didefinisikan oleh framework). Mekanisme yang sama memungkinkan komponen untuk diganti oleh pengguna. Semua aplikasi merupakan rangkaian set layanan dan sistem termasuk:

1. *Views* yang kaya dan extensible yang dapat digunakan untuk membangun aplikasi, termasuk list, grids, kotak, teks, tombol, dan bahkan sebuah embedded web.
2. *Content* *Provider* yang memungkinkan aplikasi untuk mengakses data (seperti data dari daftar kontak telephon) atau membagi data tersebut.
3. *Resource* *Manager* yang menyediakan akses ke kode non sumber local seperti *string*, gambar, dan tata letak file.
4. *Notification* *Manager* yang memungkinkan semua aplikasi menampilkan alert yang bisa dikustomisasi di dalam status bar.
5. *Activity* *Manager* yang mengelola siklus hidup aplikasi dan menyediakan navigasi umum *backstack*.
6. Applications

Android telah menyertakan aplikasi ini seperti email client, SMS, kalender, peta, browser, kontak, dan lainnya. Semua aplikasi tersebut ditulis dengan menggunakan bahasa pemrograman Java. Pada layer inilah developer atau kita menempatkan aplikasi yang dibuat. Yang isitmewa adalah pada *android* semua aplikasi baik aplikasi inti (*native*) maupun aplikasi pihak ketiga berjalan pada layer aplikasi dengan menggunakan library API yang sama. Ini berarti semua aplikasi yang dibuat untuk *android* akan memiliki hak akses yang setara dalam mengakses seluruh aplikasi inti atau aplikasi pihak ketiga. Dalam kata lain dengan platform android ini, Programmer atau Developer secara penuh akan bisa mengkustomisasi perangkat *android*-nya.

1. **Fundamental Aplikasi**

Aplikasi android ditulis dalam bahasa pemrograman Java, yaitu kode Java yang terkompilasi – bersama-sama dengan data dan file resources yang dibutuhkan oleh aplikasi – yang digabungkan oleh aapt tools menjadi paket *Android*, sebuah file yang ditandai dengan suffix .apk. File inilah yang didistribusikan sebagai aplikasi dan diinstall pada perangkat mobile, file ini yang didownload oleh pengguna ke perangkat mobile mereka. Semua kode dijadikan satu file .apk, dan kemudian kita sebut sebagai sebuah aplikasi.

Dalam banyak cara, masing-masing aplikasi android aktif dalam lingkungannya sendiri :

1. Secara default, semua aplikasi berjalan sendiri dalam proses Linux. *Android* memulai proses ketika salah satu dari kode aplikasi harus dijalankan, dan menutup proses saat tidak lagi diperlukan dan sumber daya sistem tersedia untuk aplikasi lain.
2. Setiap proses memiliki *Virtual Machine* (VM) sendiri, maka kode aplikasi yang dijalankan diisolasi dari semua kode aplikasi lain.
3. Secara default, setiap aplikasi diberikan sebuah user ID Linux yang unik. Perizinan (*Permission*) diatur agar aplikasi dari file tersebut terlihat hanya untuk user yang memiliki izin untuk aplkasi itu sendiri, meskipun ada cara untuk mengekspornya ke aplikasi lain juga, dan bisa berjalan dengan baik.

Sangat mungkin mengatur dua aplikasi yang sama untuk berbagai user ID, dalam hal ini masing-masing mereka akan dapat melihat file mereka satu sama lain. Untuk menjaga sumber daya sistem, aplikasi dengan ID yang sama juga dapat diatur untuk menjalankan proses Linux yang sama, dan berbagi VM yang sama.

1. **Android Studio**

Android Studio adalah salah satu ADT yang digunakan pada bidang pengembangan aplikasi anroid, berdasarkan *IntelliJ IDEA.* Selain merupakan editor kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menwarkan fitur lebih banyak untuk meningkatkan produktifitas anda sat membuat aplikasi Android, misalnya :

1. Sistem versi berbasis *Gradle* yang fleksibel.
2. Emulator yang cepat dan memiliki banyak fitur.
3. Tool yang menyatu untuk pengembangan bagi semua perangkat android.
4. Instant *Run* untuk menjalankan perubahan ke aplikasi yang berjalan tanpa membuat APK baru.
5. Kumpulan kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
6. Alat *Lint* yntuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya.
7. Dukungan C++ dan NDK.

Dukungan bawaan untuk *Google Cloud Platform,* mempermudah pengintegrasian *Google Cloud Messaging* dan *App Engine*.

1. **XML**
2. **Pengertian XML**

XML (*eXtensible Markup Language*) merupakan bahasa web turunan dari SGML (*Standart Generalized Markup Language*) yang ada sebelumnya. XML hampir sama dengan HTML, dimana kedua – duannya diturunkan dari SGML. Secara sederhana XML adalah suatu bahasa yang digunakan untuk mendeskripsikan dan memanipulasi dokumen secara terstruktur. Secara teknis XML didefinisikan sebagai suatu bahasa *meta-markup* yang menyediakan format tertentu untuk dokumen – dokumen data terstruktur. Bahasa *markup* adalah mekanisme untuk mengenal suatu struktur didokumen (Supriyanto, 2007).

1. **Tujuan XML**

Pada saat kita memakai XML, aplikasi A dapat menerima XML-tagged data dari aplikasi B dan sebaliknya. Kedua – duanya tidak perlu mengetahui bagaimana cara kerja sistem. Bila organisasi aplikasi A mempunyai kesepakatan untuk berbisnis dengan organisasi aplikasi Bmaka organisasi A tidak perlu menulis kode untuk menukar informasi dengan sistem aplikasi B. Tetapi yang penting adalah bagaimana memvalidasi dokumen yang dipertukarkan. Sehingga tujuan desain XML dapat dirinci sebagai berikut:

1. XML harus mendukung secara luas berbagai aplikasi
2. XML harus kompatibel dengan SGML sebagai pendahulunya
3. Program yang memproses dokumen XML harus mudah dibuat

Singkatnya tujuan dari XML secara umum adalah memungkinkan SGML untuk membantu, menerima dan memproses pada aplikasi. XML dibuat untuk memudahkan pelaksanaan dan interoperability dengan SGML dan HTML. Suatu XML menggambarkan sebuah kelas dari obyek data yang disebut dokumen XML yang disimpan pada komputer, dan menggambarkan sebagian tentang program yang memproses obyek tersebut (Supriyanto, 2007).

1. **Elemen XML**

Elemen XML merupakan inti dari dokumen XML, seperti Tempat, alamat, koordinat dan keterangan. Mereka menulis konsep utama dari dokumen XML. Elemen terdiri dari *tag* pembuka, isi dan *tag* penutup. Misalnya :

<tempat>Goa Akbar</tempat>

Nama *tag* dapat dipilih hampir secara bebas, walau ada beberapa aturan. Yang paling penting adalah bahwa karakter pertama harus huruf, garis bawah atau titik dua dan nama tidak boleh dimulai dengan string “xml” dalam setiap kombinasi huruf besar apapun (seperti “Xml” dan “xML”).

Isi dapat berupa teks atau elemen lain atau kosong. Misalnya :

<tempat>

<nama>Masjid Agung</nama>

<keterangan>Masjid Agung Tuban</keterangan>

</tempat>

Jika tidak terdapat isi, maka elemen ini disebut kosong. Elemen kosong seperti:

<tempat></tempat>

Dapat disingkat menjadi:

<tempat/> (Antoniou, 2004).

1. **Atribut XML**

Sebuah elemen kosong belum tentu tidak berarti, karena mungkin bisa mempunyai properti dalam bentuk atribut. Suatu atribut adalah pasangan nama dan nilai di dalam *tag* pembuka sebuah elemen.

<tempat nama=”Masjid Agung” keterangan=”Masjid Agung Tuban”/>

Ini adalah contoh atribut dalam elemen yang tidak kosong.

<kategori nama=”Alam”>

<tempat nama=”Goa Suci” keterangan=”Goa Bekas Tambang Balanda”/>

</kategori>

Informasi yang sama dapat ditulis seperti berikut, mengganti atribut dengan elemen bersarang:

<kategori>

<nama>Alam</nama>

<tempat>

<nama>Goa Suci</nama>

<keterangan>Goa Bekas Tambang Belanda</keterangan>

</tempat>

</kategori>

Kapan harus menggunakan elemen dan kapan harus menggunakan atribut sering kali adalah merupakan masalah selera programer. Namun, perhatikan bahwa atribut tidak dapat dibuat bersarang (Antoniou, 2004).

1. **Pembacaan Dokumen XML**

Syarat agar dokumen XML dapat dibaca harus memenuhi dua level kebenaran, meliputi:

1. *Well-Formed* : dokumen XML memiliki syntax yang benar, antara lain kesesuaian penulisan elemen, *start tag* (< >) harus diikuti dengan *end tag* (</ >).
2. Valid : isi elemen – elemen dalam dokumen XML sesuai dengan aturan semantik, atau tipa data yang dimaksud, misal isi elemen yang tidak terdefinisi nilainya dianggap tidak valid.

Terdapat 2 cara untuk memvalidasi sutau dokumen XML. Cara pertama melalui *document rule* (*Document Type Definition* - DTD). Cara kedua adalah *XML Scheme*. DTD adalah koleksi dari sumber daya internal (internal pada isi dokumen XML) dan eksternal yang secara kolektif memberikan *formal grammer* pada dokumen XML. *XML Scheme* merupakan definisi bahasa yang digunakan untuk menjelaskan struktur XML (Supriyanto, 2007).

Proses pembacaan dokumen XML disebut juga *parsing*. Ada beberapa pustaka atau perangkat lunak untuk membaca *file* XML, antara lain MSXML dari Microsoft dan org.w3c.dom untuk pemrograman java (Rahman, 2011).

Hubungan istilah valid dan *well formed* adalah semua dokumen yang valid juga disusun dengan baik. Namun sebaliknya dokumen yang disusun dengan baik belum tentu valid. Sehingga dokumen XML valid adalah dokumen yang memiliki struktur dan isi elemennya secara formal dinyatakan dengan DTD. Dokumen yang disusun dengan baik (*well formed*) adalah dokumen yang jika tidak memerlukan pemakaian DTD dalam memahami struktur dan isi elemennya, tetapi mengikuti prinsip – prinsip umum XML, seperti *tag nested* (*tag* bersarang) secara tepat (Supriyanto, 2007).

1. **Flowchart**

Flowchart dapat diartikan sebagai gambaran dalam bentuk diagram alir dari algoritma-algoritma dalam suatu program yang menyatakan arah alur program tersebut (Pahlevy, 2010).

Berikut adalah daftar simbol flowchart yang ditunjukkan pada Tabel 2.1 :

Tabel 2.1 Simbol Flowchart (Pahlevy, 2010)

|  |  |  |  |
| --- | --- | --- | --- |
| **Simbol** | **Nama dan Keterangan** | **Simbol** | **Nama dan Keterangan** |
|  | **Flow Direction Symbol**  Digunakan untuk menghubungkan simbol satu dengan simbol yang lain. |  | **Symbol Manual Input**  Untuk Memasukkan Datan Secara Manual. |
|  | **Terminator Symbol**  Untuk permulaan (Start) atau akhiran (Finish) dari suatu kegiatan. |  | **Symbol Preparation**  Untuk mempersiapkan penyimpanan yang akan digunakan sebagai storage. |
|  | **Connector Symbol**  Untuk keluar masuk atau penyambungan proses pada lembar / halaman yang berbeda. |  | **Symbol Display**  Untuk menyatakan peralatan output. |
|  | **Processing Symbol**  Untuk melakukan suatu pengolahan oleh komputer. |  | **Symbol Disk**  Untuk menyatakan input dari suatu diks atau menyimpan ke disk. |
|  | **Symbol Manual Operation**  Untuk melakukan suatu pengolahan bukan komputer. |  | **Symbol Magnetic Tape Unit**  Untuk menyatakan input dari suatu pita magnetik atau menyimpan ke pita magnetik. |
|  | **Symbol Decision**  Untuk proses memilih kondisi yang ada. |  | **Symbol Punch Crad**  Untuk menyatakan input dari kartu atau menyimpan ke kartu. |
|  | **Symbol Input-Output**  Untuk melakukan suatu proses input atau output. |  | **Symbol Document**  Untuk menyatakan input dari suatu dokumen atau menyimpan ke dokumen. |

1. **UML (Unified Modelling Language)**

UML adalah bahasa grafis untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak tergantung pada bahasa dan teknologi, pemaduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak. Standar UML dikelola oleh OMG (Object Management Group) (Nugroho, 2010).

UML adalah bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan artifak- artifak dari sistem.

Diagram-diagram yang terdapat pada pemodelan UML sebagai berikut:

1. *Usecase Diagram*

Usecase diagram merupakan salah satu diagram untuk memodelkan aspek perilaku sistem. Masing-masing diagram usecase menunjukan sekumpulan usecase, aktor dan hubungannya. Diagram usecase digunakan untuk memvisualisasikan, menspesifikasikan dan mendokumentasikan kebutuhan perilaku sistem. Diagram-diagram usecase merupakan pusat permodelan perilaku sistem, subsistem dan kelas.

1. *Class Diagram*

Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Class diagram menggambarkan keadaan (atribut atau properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metode atau fungsi). Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain. *Class diagram* memiliki tiga area pokok, yaitu sebagai berikut :

1. Nama (dan stereotype)
2. Atribut
3. Metoda
4. *Statechart Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada system sebagai akibat dari stimuli yang diterima. Pada umumnya *statechart diagram menggambarkan class tertentu (satu class* dapat memiliki lebih dari satu statechart diagram).

1. *Activity Diagram*

Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alur berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-*trigger* oleh selesainya state sebelumnya (internal processing). Oleh karena itu activity diagram tidak menggambarkan perilaku internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

1. *Sequence Diagram*

*Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

*Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang menjadi trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

1. *Collaboration Diagram*

Collaboration diagram juga menggambarkan interaksi antar objek seperti sequence diagram, tetapi lebih menekankan pada peran masing masing objek dan bukan pada waktu penyampaian *message*. Setiap message memiliki sequence number, di mana message dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

1. *Component Diagram*

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (dependency) di antaranya. Komponen piranti lunak adalah modul berisi code, baik berisi source code maupun binary code, baik *library* maupun *executable,* baik yang muncul pada *compile time,* link time, maupun run time. Umumnya komponen terbentuk daribeberapa class dan, atau package, tapi dapat juga dari komponen-komponen yang lebih kecil.

1. *Deployment Diagram*

*Deployment* atau *physical diagram* menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisik. Sebuah node adalah server, workstation, atau piranti keras lain yang digunakan untuk men-deploy komponen dalam lingkungan sebenarnya. Hubungan antar node (misalnya TCP/IP) dan requirement dapat juga didefinisikan dalam diagram ini.

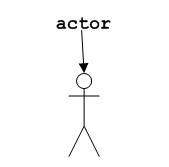
1. **Notasi UML**

Notasi UML diturunkan dari tiga notasi yang sudah ada sebelumnya yaitu Grady Booch OOD *(Object Oriented Design)*, Jim Rumbough OMT *(Object Modelling Technique)*, dan Ivar Jacobson *OOSE (Object Oriented Sofware Engineering).* Notasi UML yang digunakan sekarang merupakan penggabungan dari tiga notasi tersebut, yang terdiri dari :

1. Aktor **(***actor****)***

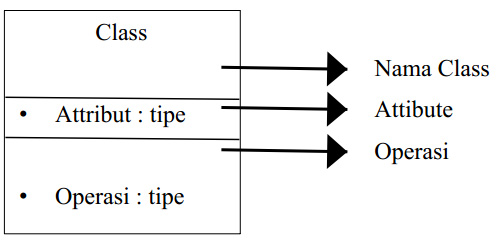
Dalam pemodelan sistem dengan UML, aktor adalah seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang kita kembangkan. Aktor berada diluar lingkup sistem atau perangkat lunak yang sedang kita kembangkan; bersifat eksternal (Nugroho, 2010).

Secara prinsip dapat kita kenali 3 jenis aktor untuk hamper semua sistem atau perangkat lunak yang kita kembangkan: para pengguna sistem, perangkat lunak lain yang berinteraksi dengan sistem, dan perangkat lunak yang kita kembangkan serta waktu. Jadi aktor ini bisa berupa orang, perangkat keras atau mungkin juga objek lain dalam sistem yang sama. Biasanya yang dilakukan oleh aktor adalah memberikan informasi pada sistem dan atau memerintahkan sistem untuk melakukan sesuatu.



Gambar 2.1 Notasi Aktor Sumber: (Nugroho, 2010)

Kelas seperti juga objek, adalah sesuatu yang membungkus (*encapsulate*) informasi dan perilaku dalam dirinya (Nugroho, 2006). Dalam pengembangan sistem tradisional, kita mengadakan pendekatan dengan cara memisahkan informasi-informasi pada sisi basis data dan perilaku yang mengaksesnya di sisi aplikasi pemasup atau pengakses. Pendekatan berorientasi objek berbeda, yaitu menggabungkan potongan-potongan informasi dengan perilaku yang akan mengaksesnya dalam apa yang dinamakan kelas.

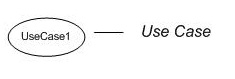


Gambar 2.2 Notasi Kelas Sumber : (Nugroho, 2010)

1. *Usecase*

*Usecase* adalah peringkat tertinggi dari fungsional yang dimiliki sistem (Nugroho, 2006). Dengan kata lain, usecase menggambarkan bagaimana seseorang akan menggunakan sistem. Usecase menjelaskan suatu urutan kegiatan yang dilakukan oleh aktor dan sistem untuk mencapai tujuan tertentu walaupun menjelaskan kegiatan namun usecase hanya menjelaskan apa yang dilakukan oleh aktor dan sistem, bukan bagaimana aktor dan sistem melakukan kegiatan tersebut.

Keunggulan dari cara memandang sistem sebagai kumpulan usecase adalah kemampuannya untuk memisahkan implementasi sistem dari alasan mengapa sistem harus ada. Ia akan membantu kita untuk berfokus pada apa yang paling penting, yaitu menentukan apa yang dibutuhkan serta apa harapan pengguna terhadap sistem atau perangkat lunak yang sedang dikembangkan.

Gambar 2.3 Notasi Usecase Sumber: (Nugroho, 2010)

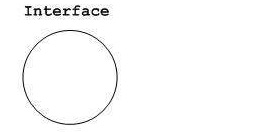
1. *Interaction*

Menurut Interaction digunakan untuk menunjukan baik aliran pesan atau informasi antar objek maupun hubungan antar objek. Biasanya interaction ini dilengkapi juga dengan teks bernama *operation signature* yang tersusun dari nama operasi, parameter yang dikirim dan tipe parameter yang dikembalikan (Nugroho, 2004).

Gambar 2.4 Notasi Interaction Sumber : ( Nugroho, 2010)

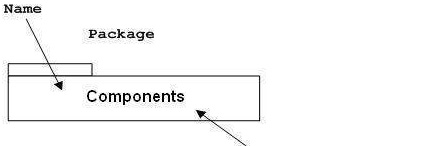
1. *Interface*

*Interface* merupakan kumpulan operasi tanpa implementasi dari suatu class. Implementasi operasi dalam interface dijabarkan dalam operasi dalam class. Oleh karena itu keberadaan interface selalu disertai oleh class yang mengimplementasikan operasinya (Nugroho, 2007). Interface ini merupakan salah satu cara mewujudkan prinsip enkapsulasi dalam objek.

Gambar 2.5 Notasi Interface Sumber : (Nugroho, 2010)

1. *Package*

Package adalah container atau wadah konseptual yang digunakan untuk mengelompokkan elemen-elemen dari suatu sistem yang sedang dibangun, sehingga bisa dibuat model yang lebih sederhana. Tujuannya adalah untuk mempermudah penglihatan (*visibility*) dari suatu model yang sedang dibangun.



Gambar 2.6 Notasi Package Sumber: (Nugroho, 2010)

1. *Note*

*Note* dibangun untuk memberikan keterangan dan komentar tambahan dari suatu elemen sehingga bisa langsung terlampir dalam model. Note ini bisa ditempelkan ke semua elemen notasi yang lain.

Gambar 2.7 Notasi Note Sumber: (Nugroho, 2010)

1. *Dependency*

Merupakan relasi yang menunjukan bahwa perubahan pada suatu elemen memberikan pengaruh pada elemen yang lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada di bagian tanpa ada tanda panah.

Terdapat dua *stereotype* dari *dependency,* yaitu *include* dan *extend*. *Include* menunjukan bahwa suatu bagian dari elemen (yang ada di garis tanpa panah) memicu eksekusi bagian dari elemen yang lain (yang ada di garis dengan panah), misalnya untuk notasi A → B operasi yang ada di class A memicu dieksekusinya operasi yang berada di class B.

Extend menunjukan bahwa suatu bagian dari elemen di garis tanpa panah bisa disiapkan ke dalam elemen yang ada di gari dengan panah, misalnya untuk notasi A→B suatu fungsi dari usecase A bisa disisipkan ke dalam usecase B atau dengan kata lain A optional untuk B. Ke dua *stereotype* ini di representasikan denganmenambahkan text include atau extend di notasi dependency.



Gambar 2.8 Notasi Dependency (Nugroho, 2010)

1. *Association*

*Association* menggambarkan navigasi antar class (*navigation*), berapa banyak objek lain yang bisa berhubungan dengan satu objek (multiplicity antar class) dan apakah suatu class menjadi bagian dari class lainnya (aggregation).

*Navigation* di lambangkan dengan penambahan tanda panah di akhir garis*. Bidirectional navigation* menunjukan bahwa dengan mengetahui salah satu class bisa didapatkan dari informasi lainnya. Sementara dengan unidirectional navigation hanya dengan mengetahui class di ujung garis association tanpa panah kita bisa mendapatkan informasi dari class di ujung dengan panah, tetapi tidak sebaliknya.

Gambar 2.9 Notasi Association Sumber: (Nugroho, 2010)

Pada penelitian kali ini penulis menggunakan Visual Paradigm untuk merancang aplikasi. Visual Paradigm adalah perangkat lunak yang digunakan untuk perancangan aplikasi dengan tools UML.