BAB II
LANDASAN TEORI

II.1 Tentang PT. Raya Cipta Mandiri

PT. Raya Cipta Mandiri adalah perusahaan yang bergerak dalam bidang Distributor Ice Cream Wall's.

Nama Perusahaan : PT. Raya Cipta Mandiri

Lokasi : JL. Pandan Puti No.39 RT.03 RW.03 Kel. KTK Solok

II.2 Struktur Organisasi Perusahaan

Berikut ini adalah struktur organisasi PT. Raya Cipta Mandiri yang dapat dilihat pada Gambar II.1 Struktur Organisasi Perusahaan.

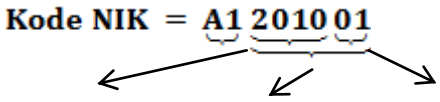


Gambar II.1 Struktur Organisasi Perusahaan

II.3 Struktur Kode NIK (Nomor Induk Karyawan)

Kode NIK atau kode Nomor Induk Karyawan merupakan pengkodean untuk menentukan nomor urut pegawai yang terdaftar pada PT. Raya Cipta Mandiri. Dalam hal ini kode NIK diperlukan bagi karyawan sebagai "surat jalan" dalam melakukan tugas sebagai canvasser. Kode NIK terdiri delapan digit dua digit pertama merupakan kode jabatan, empat digit selanjutnya merupakan tahun masuk karyawan dan dua digit terakhir merupakan no urut karyawan.

Adapun contoh dari kode NIK beserta penjelasannya adalah sebagai berikut :



Kode jabtan tahun masuk no urut

II.4 Sistem Informasi

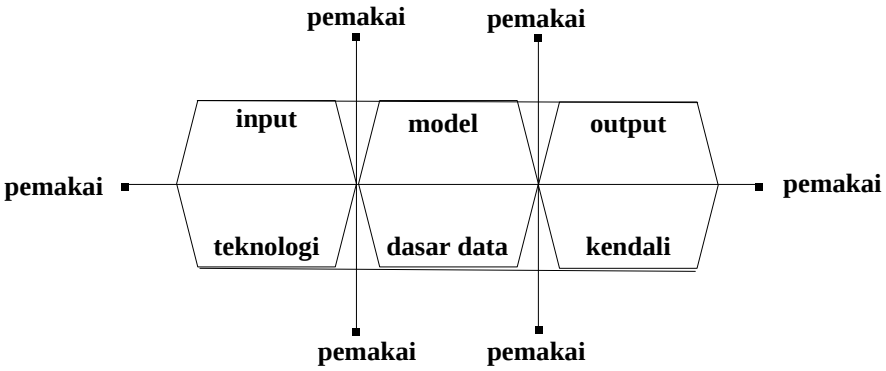
Sistem adalah kumpulan dari komponen-komponen peralatan model *requirements*, *function* dan *interface*. Sistem informasi merupakan kombinasi teratur apapun dari sumber daya manusia, *hardware*, *software*, data, dan jaringan. Daur hidup suatu sistem meliputi [2]:

1. Investigasi
2. Analisis
3. Desain
4. Implementasi
5. Pemeliharaan

II.4.1 Komponen Sistem Informasi

Sistem informasi terdiri dari komponen-komponen yang disebutnya dengan istilah blok bangunan (*building block*), yaitu blok masukan (*input block*),

blok model (*model block*), blok keluaran (*output block*), blok teknologi (*technology block*), blok basis data (*database block*) dan blok kendali (co*ntrols block*). Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya [3].



Gambar II.2 Blok Sistem Informasi yang Berinteraksi [3]

Blok Masukan

Input mewakili data yang masuk ke dalam sistem informasi. Input disini termasuk metode-metode dan media untuk menangkap data yang akan dimasukkan, yang dapat berupa dokumen-dokumen dasar.

Blok Model

Blok ini terdiri dari kombinasi prosedur, logika dan model matematika yang akan memanipulasi data input dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

Blok Keluaran

Produk dari sistem informasi adalah keluaran yang yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta semua pemakai sistem.

Blok Teknologi

Teknologi merupakan “kotak alat” (*tool-box*) dalam sistem informasi. Teknologi digunakan untuk menerima input, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem secara keseluruhan. Teknologi terdiri dari 3 bagian utama, yaitu teknisi (*humanware* atau *brainware*), perangkat lunak (*software*) dan perangkat keras (*hardware*). Teknisi dapat berupa orang-orang yang mengetahui teknologi dan membuatnya dapat beroperasi. Misalnya teknisi adalah operator komputer, pemrogram, operator pengolah data, spesialis telekomunikasi, analis sistem, penyimpanan data dan lain sebagainya.

Blok Basis Data

Basis data (*database*) merupakan kumpulan dari data yang saling berhubungan satu dengan yang lainnya, tersimpan di perangkat keras komputer dan digunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan di dalam basis data untuk keperluan penyediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikian rupa, supaya informasi yang dihasilkan berkualitas. Organisasi basis data yang baik juga berguna untuk efisiensi kapasitas penyimpanan. Basis data diakses atau dimanipulasi dengan menggunakan perangkat lunak paket yang disebut dengan DBMS (*Database management System*).

Blok Kendali

Banyak hal yang dapat merusak sistem informasi, seperti halnya bencana alam, api, temperatur, air, debu, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, kesalahan-kesalahan, ketidak efisienan, sabotase dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah ataupun bila terlanjur terjadi kesalahan-kesalahan dapat cepat langsung diatasi.

II.5 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, *middleware*, dan aplikasi. Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan *platform* terbuka bagi para pengembang untuk membuat aplikasi mereka sendiri. Pada awalnya dikembangkan oleh *Android Inc*, sebuah perusahaan pendatang baru yang membuat perangkat lunak untuk ponsel yang kemudian dibeli oleh *Google Inc*. Untuk pengembangannya, dibentuklah Open Handset Alliance (OHA), konsorsium dari 34 perusahaan perangkat keras, perangkat lunak, dan telekomunikasi termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia [4].

II.5.1 Android Life Cycle

Aplikasi android terdiri dari beberapa fungsi dasar seperti mengedit catatan, memutar file musik, membunyikan alarm, atau membuka kontak telepon. Fungsi-fungsi tersebut dapat diklasifikasikan ke dalam empat komponen android yang berbeda seperti ditunjukkan pada , klasifikasi tersebut berdasarkan kelas-kelas dasar java yang digunakan [5].

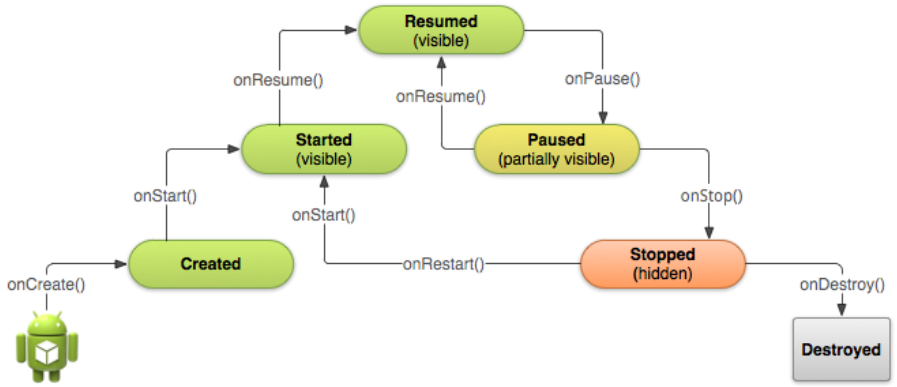
Tabel II.1 Komponen Aplikasi Android

Functionality	Java Base Class	Examples
Focused thing a user can do	Activity	Edit a note, play a game
Background process	Service	Play music, update weather icon
Receive messages	BroadcastReceiver	Trigger alarm upon event
Store and retrieve data	ContentProvider	Open a phone contact

Setiap aplikasi pasti menggunakan minimal satu dari komponen tersebut, akan tetapi terdapat beberapa komponen yang mengharuskan mencantumkan specified permission sebelum digunakan seperti komponen Service, BroadcastReceiver. ContentProvider [5].

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa di mana aplikasi yang dijalankan pada fungsi main(), sistem

android menjalankan kode dalam method Activity dengan menerapkan metode callback tertentu yang sesuai dengan tahap tertentu dari siklus hidup. Setiap aplikasi yang berjalan dalam sistem operasi android memiliki siklus hidup yang berbeda dengan aplikasi desktop atau web. Hal ini dikarenakan aplikasi mobile memiliki tingkat interupsi proses yang lumayan tinggi seperti ketika handling panggilan masuk aplikasi diharuskan menghentikan proses sementara. Penerapan siklus hidup juga berguna untuk memastikan aplikasi tidak menghabiskan sumber daya baterai pengguna [6].



Gambar II.3 Siklus Hidup Android [6]

Terdapat beberapa state dalam siklus hidup android yang terjadi seperti diilustrasikan pada Gambar II.4 Siklus Hidup Android [6], akan tetapi hanya beberapa dari state tersebut yang menjadi statis diantaranya :

1. Resumed

Resumed terjadi ketika aplikasi berjalan setelah state *paused* . State ini akan menjalankan perintah program yang ditulis pada method onResume() (Google Inc, 2014).

2. Paused

Dalam keadaan ini aktivitas yang terjadi dihentikan secara sementara tetapi masih terlihat oleh pengguna karena terdapat proses yang memiliki prioritas lebih tinggi seperti panggilan telepon. Aplikasi tidak dapat menjalankan perintah apapun ataupun menampilkan apapun dalam state ini (Google Inc, 2014).

3. Stopped

Dalam keadaan ini, aplikasi benar-benar tidak ditampilkan dan tidak terlihat oleh pengguna tetapi masih meninggalkan service dibackground (Google Inc, 2014).

State lain seperti Created dan Started bersifat sementara dan sistem dengan cepat menjalankan state berikutnya dengan memanggil metode *life cycle callback* berikutnya. Artinya, setelah sistem OnCreate() dipanggil, dengan cepat sistem akan memanggil method OnStart(), krmudian diikuti oleh onResume() (Schwarz, Dutson, Stelee, & To, 2013).

II.5.2 Fitur

Android memiliki beberapa fitur utama yang sering digunakan dalam proses pembangunan aplikasi diantaranya adalah:

1. Multi-proses dan App Widgets

Sistem operasi android tidak melarang prosesor menjalankan lebih dari satu aplikasi dalam satu waktu. Sistem operasi android dapat mengatur aplikasi dan thread yang berjalan secara multitasking. Keuntungan yang didapat adalah ketika aplikasi berjalan dan berinteraksi dengan pengguna di layer depan sistem operasi, proses dari aplikasi lain dapat berjalan untuk melakukan pembaruan informasi. Sebagai contoh misalnya ketika pengguna memainkan game, proses lain dapat berjalan di belakang aplikasi seperti memeriksa harga saham dan memunculkan peringatan [6].

App Widgets adalah mini aplikasi yang dapat embedded dalam aplikasi seperti home screen. App widgets dapat menjalankan proses request seperti musik streaming atau mendeteksi suhu ruangan secara background [6].

Multi-proses dapat memberikan manfaat berupa user experience yang lebih banyak, namun penggunaan fitur tersebut dapat menghabiskan banyak energi baterai jika penggunaan tidak benar [7].

2. Touch Gestures dan Multi-touch

Touchscreen adalah user interface intuitif yang digunakan banyak smartphone di dunia. Dengan fitur ini interaksi dapat dibuat lebih mudah karena cukup dengan menggunakan jari tangan. Multi-touch adalah kemampuan yang dapat melakukan tracking lebih dari satu tangan dalam satu waktu, Fitur ini sering digunakan untuk interaksi memperbesar atau memutar objek. Selain itu pengembang dapat membuat interaksi baru dengan memanfaatkan fitur tersebut [7].

3. Hard dan Soft Keyboard

Salah satu fitur pada perangkat smartphone adalah tombol fisik dan non fisik, tombol fisik digunakan untuk navigasi pendukung dalam pengoperasian android.. Pengembang aplikasi tidak perlu secara manual untuk mengintegrasikan tombol tersebut dalam aplikasi. Tombol non fisik adalah tombol yang dibuat oleh sistem operasi seperti keyboard virtual, dan tombol navigasi aplikasi [7].

II.5.3 Prinsip Desain

Android memiliki beberapa prinsip desain yang dapat menjadi acuan dalam membuat desain aplikasi android diantaranya adalah

1. Multiple Assets

Android mendukung jutaan smartphone, tablet dan perangkat lain dalam berbagai ukuran layar dan ukuran, untuk itu Multiple Assets sangat disarankan digunakan untuk mengatasi fragmentasi pada android. Seperti ilustrasi pada Gambar II.5 Klasifikasi Ukuran Ikon, android menciptakan beberapa klasifikasi ukuran icon yaitu MDPI, HDPI, XHDPI, XXHDPI dan XXXHDPI. MDPI dan HDPI dikhususkan untuk icon yang akan digunakan pada device berukuran smartphone sedangkan untuk XHDPI, XXHDPI dan XXXHDPI digunakan pada device berukuran tablet [8].



Gambar II.4 Klasifikasi Ukuran Ikon

2. Touch Feedback

Touch Feedback dalam android digunakan sebagai respon setiap objek yang ditekan pengguna. Hal ini bertujuan untuk memberi tahu pengguna objek mana yang berinteraksi dengan pengguna [8].



Gambar II.5 Touch Feedback

3. Pattern Gesture

Touch gesture berguna untuk meningkatkan experience pengguna dalam menggunakan aplikasi. Terdapat beberapa gesture yang didukung oleh android diataranya adalah [8]:

1. Touch

Touch adalah gesture default sebagai event untuk item tertentu yang paling sering digunakan dalam perangkat android, hampir seluruh kontrol utama aplikasi ini menggunakan touch.

2. Long Press

Biasanya digunakan untuk seleksi data , dengan gesture ini dimungkinkan untuk memilih satu atau lebih item dalam sebuah tampilan dan menjalankan suatu fungsionalitas tertentu [8].

3. Swipe Or Drag

Swipe adalah menyentuh sebuah titik pada layar dan menggerakkan jari yang tetap tersentuh pada layar ke titik lain pada layar. Swipe biasanya dilakukan untuk kebutuhan berpindah layar atau tergantung dari aplikasi. Swipe dapat dilakukan dari dan ke arah mana saja [8].

4. Double Touch

Pada Smartphone dan tablet android, melakukan dua kali tapping secara berturut-turut pada satu obyek, fungsinya berbeda dengan double klik mouse komputer. Pada Android, teknik ini biasanya dipakai untuk melakukan zoom in dan zoom out atau memperbesar / memperkecil sebuah obyek gambar [10].

5. Pinch Open

Teknik lain yang biasa digunakan adalah dengan menggunakan dua jari, di mana kedua jari tersebut menyentuh dua titik pada layar yang terpisah di mana ujung dari dua jari tersebut tidak bersentuhan, kemudian kedua jari tersebut sambil tetap menyentuh layar bergerak saling mendekati. Teknik ini digunakan untuk membuka aplikasi tertentu [8].

6. Pinch Close

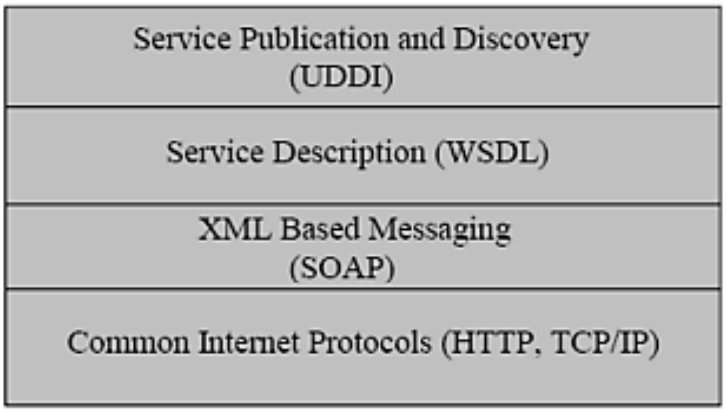
Teknik Pinch Close adalah kebalikan dari Pinch Open di mana spread dilakukan dengan berawal dari dua jari yang bersentuhan pada ujungnya dan ditempelkan sebuah titik yang sama pada layar, kemudian kedua jari tersebut bergerak memisahkan atau menjauhi satu sama lain Gerakan ini untuk menutup layar [8].

II.6 Web Services

Web service merupakan suatu komponen software yang merupakan selfcontaining, aplikasi modular self-describing yang dapat dipublikasikan, dialokasikan, dan dilaksanakan pada web. Web service adalah teknologi yang mengubah kemampuan internet dengan menambahkan kemampuan transactional web, yaitu kemampuan web untuk saling berkomunikasi dengan pola program-to-program (P2P). Fokus web selama ini didominasi oleh komunikasi program-to-user dengan interaksi business-to-consumer (B2C), sedangkan transactional web akan didominasi oleh program-to-program dengan interaksi business-to-business.

Gambar 1 merupakan blok bangunan web service yang mana menyediakan fasilitas komunikasi jarak jauh antara dua aplikasi yang merupakan layer arsitektur web service.

1. Layer 1 : Protokol internet standar yang digunakan sebagai sarana transportasi adalah HTTP dan TCP/IP.
2. Layer 2 : Simple Object Access Protocol (SOAP) berbasiskan XML dan digunakan untuk pertukaran informasi antar sekelompok layanan.
3. Layer 3 : Web service Definition Language (WSDL) digunakan untuk mendiskripsikan attribute layanan.
1. Layer 4 : Universal Description, Discovery and Integration, yang mana merupakan direktori pusat untuk deskripsi layanan.



***Gambar II.6 Blok Bangunan* Web Service**

II.7 JavaScript Object Notation (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 [9].

JSON terbuat dari dua struktur [9] :

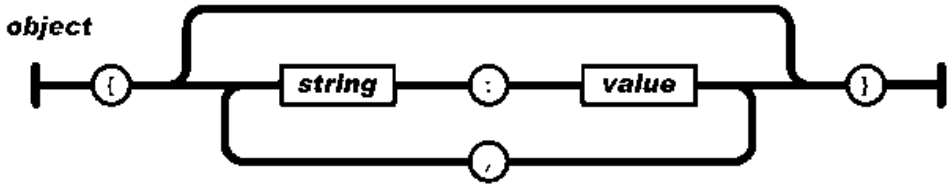
1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau associative array.
2. Daftar nilai terurutkan (*an ordered list of values*). 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 [9].

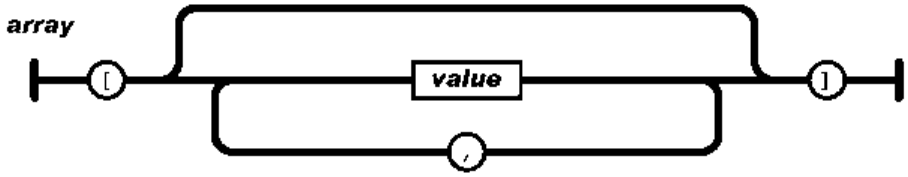
JSON menggunakan bentuk sebagai berikut [9] :

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).



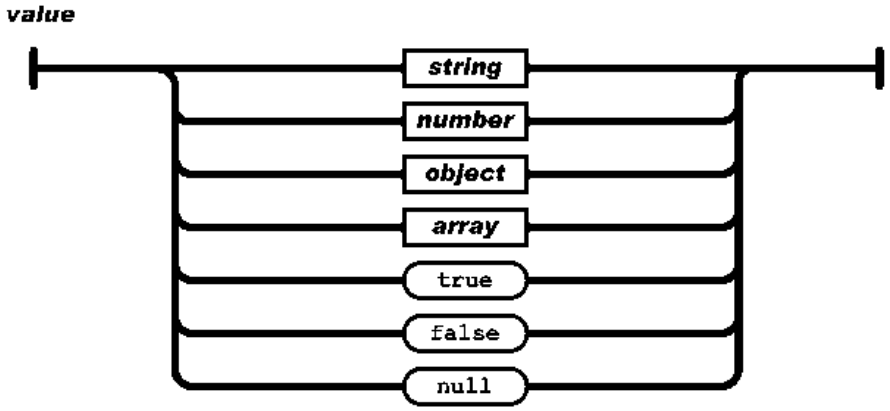
Gambar II.7 Objek JSON

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



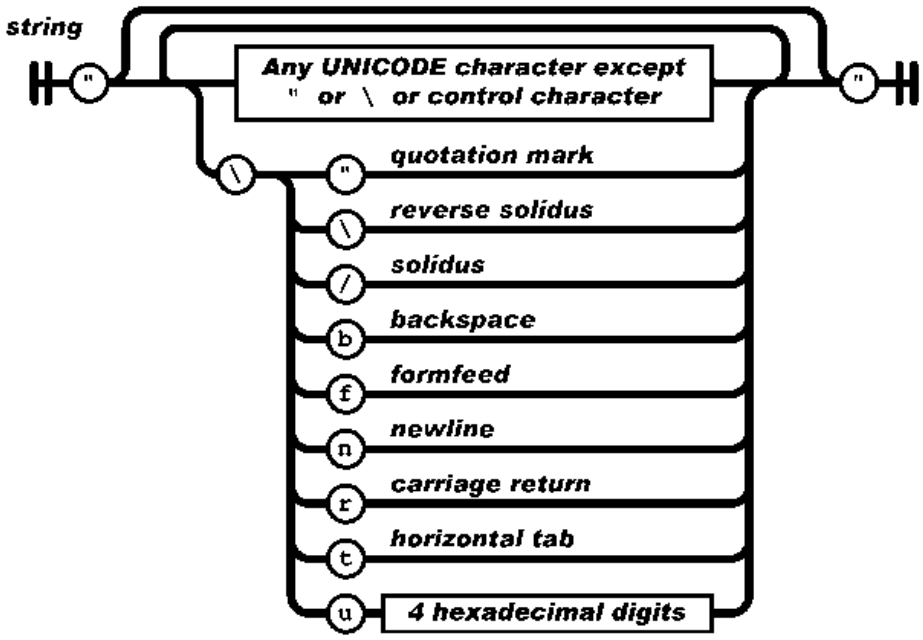
Gambar II.8 Array JSON

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.



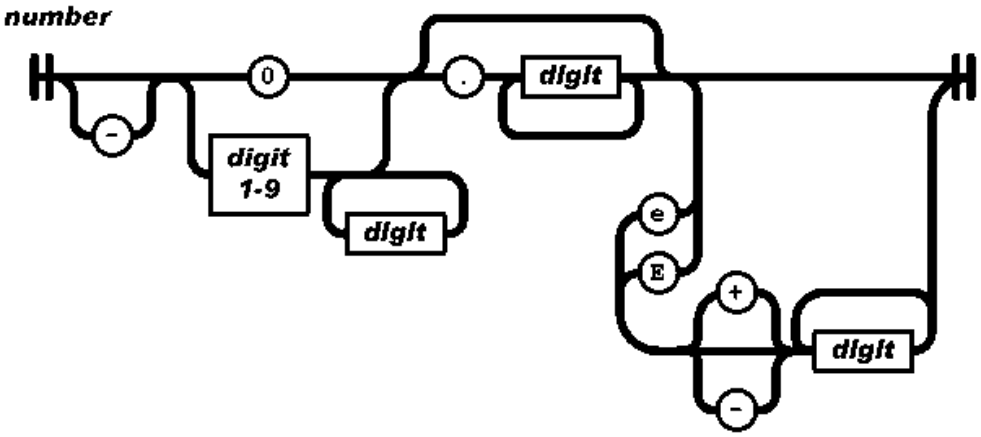
Gambar II.9 Value JSON

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 II.10 String JSON

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



Gambar II.11 Number JSON

Spasi kosong (whitespace) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil encoding yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

II.8 Global Positioning System (GPS)

Global Positioning System adalah sistem navigasi berbasis satelit terdiri dari jaringan 24 satelit ditempatkan ke orbit oleh Departemen Pertahanan Amerika Serikat yang pertama kali diperkenalkan mulai tahun 1978. Layanan GPS dahulu hanya dipergunakan untuk keperluan militer namun mulai terbuka untuk publik. 24 satelit GPS tersebut berada sekitar 12.000 mil di atas bumi bergerak mengelilingi bumi 12 jam dengan kecepatan 7.000 mil per jam. Satelit GPS berkekuatan energi sinar matahari, memiliki baterai cadangan untuk menjaga agar tetap berjalan pada saat gerhana matahari atau pada saat tidak ada energi matahari dan memiliki roket penguat kecil pada masing-masing satelit agar dapat mengorbit tepat pada tempatnya.

Satelit-satelit GPS harus selalu berada pada posisi orbit yang tepat untuk menjaga akurasi data yang dikirim ke GPS *receiver*, sehingga harus selalu dipelihara agar posisinya tepat. Posisi satelit-satelit tersebut selalu dipantau oleh stasiun pengendali. Stasiun-stasiun pengendali di bumi ada di Hawaii, Ascension Islan, Diego Garcia, Kwajalein dan Colorado Spring. Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberi nama GPS *receiver* yang

berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. GPS *receiver* mengambil informasi tersebut dan melakukan perhitungan triangulation untuk menentukan lokasi pengguna dengan tepat. GPS receiver membandingkan waktu sinyal dikirim dengan waktu sinyal tersebut diterima untuk mengetahui jarak satelit. Dengan mengetahui jarak tersebut, GPS receiver dapat melakukan perhitungan dan menentukan posisi pengguna dan menampilkan dalam peta elektronik. Setelah menentukan posisi pengguna, selanjutnya GPS dapat menghitung informasi lain, seperti kecepatan, arah yang dituju, jalur, tujuan perjalanan, jarak tujuan, matahari terbit dan matahari terbenam dan masih banyak lagi.

Satelit GPS sangat presisi dalam mengirim informasi karena satelit tersebut menggunakan jam atom. Jam atom yang ada pada satelit merupakan partikel atom yang diisolasi, sehingga dapat menghasilkan jam yang akurat dibandingkan dengan jam biasa. Keistimewaan GPS adalah mampu bekerja dalam berbagai kondisi cuaca, siang atau malam. Keakuratan sebuah perangkat GPS bisa mencapai 15 meter, bahkan model terbaru yang dilengkapi teknologi Wide Area Augmentation System (WAAS) keakuratannya sampai 3 meter. Karena GPS bekerja mengandalkan satelit, maka penggunaannya disarankan di tempat terbuka. Penggunaan di dalam ruangan, atau di tempat yang menghalangi arah satelit (di angkasa), maka GPS tidak akan bekerja secara akurat dan maksimal. Perhitungan waktu yang akurat sangat menentukan akurasi perhitungan untuk menentukan informasi lokasi. Semakin banyak sinyal satelit yang dapat diterima maka akan semakin presisi data posisi yang dihasilkan. Selain itu, ketinggian juga mempengaruhi proses kerja GPS, karena semakin tinggi maka semakin bersih atmosfer, sehingga gangguan semakin sedikit [10].

II.9 Google Geocoding

Google Maps sudah memiliki fitur yang cukup lengkap, termasuk mencari lokasi kota pada peta (*geocoding*). *Geocoding* adalah proses menemukan koordinat geografis, yaitu bujur dan lintang, dari data geografis lain, misalnya

alamat. Fitur *geocoding* dapat dimanfaatkan untuk menambahkan informasi *latitude* dan *longitude* pada foto yang tidak memiliki data *geotagging* [10].

II.10 Structure Analysis and Design

Structured analysis and design adalah pendekatan sistem analisis dan desain sistem informasi secara hierarki. SADM diproduksi untuk CCTA, kantor pemerintah Inggris yang berkaitan dengan penggunaan teknologi di pemerintahan dari tahun 1980 dan seterusnya [11].

II.10.1 Diagram Konteks

Diagram Konteks adalah data flow diagram tingkat atas (DFD Top Level), yaitu diagram yang paling tidak detail, dari sebuah sistem informasi yang menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar entitas-entitas eksternal. (CD menggambarkan sistem dalam satu lingkaran dan hubungan dengan entitas luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem) [12].

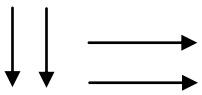
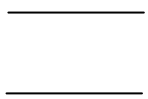
II.10.2 Data Flow Diagram

Data Flow Diagram (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut. DFD terdiri dari empat komponen utama diantaranya adalah :

1. Entitas.
2. Proses.
3. Data Store.
4. Data Flow.

DFD dapat dikatakan merupakan breakdown dari diagram konteks. Analisis mengenai sistem yang lebih detail dapat dilakukan dengan melakukan breakdown pada DFD level 1, 2 dan seterusnya. Berikut adalah Tabel II.7 Simbol Diagram Konteks dan DFD [12].

Tabel II.2 Simbol Diagram Konteks dan DFD [12]

Simbol	Nama Simbol	Fungsi
	Entitas Luar	Menggambarkan entitas eksternal yang berhubungan dengan sistem
	Sistem (Konteks) / Proses (DFD)	Menggambarkan proses yang ada dalam suatu sistem
	Aliran Data/Informasi	Menggambarkan aliran data antar proses, data store dan entitas luar
	Data Storage	Mengambarkan tempat penyimpanan data di dalam sistem

II.10.3 Spesifikasi Proses

Spesifikasi Proses adalah tabel yang berisi keterangan atau deskripsi dari semua proses yang terdapat pada DFD. Logika proses yang ditulis pada spesifikasi proses harus dituliskan secara jelas baik menggunakan bahasa deskriptif atau pseudecode tetapi tidak boleh mengkombinasikan kedua bahasa tersebut [14]. Berikut adalah Tabel II.8 Contoh Spesifikasi Proses [12]

Tabel II.3 Contoh Spesifikasi Proses [12]

No	Proses	Keterangan
1.	No Proses	Menyatakn no proses
	Nama Proses	Menyatakan nama proses
	Source (Sumber)	(Menyatakan sumber data input menuju proses)
	Input	(menyatakan isi data yang masuk ke proses)
	Output	(menyatakan informasi yang keluar dari proses)

No	Proses	Keterangan
	Destination	(menatakan tujuan informasi output dari proses)
	Logika Proses	(menyatakan algoritma dari proses)

II.10.4 Kamus Data

Kamus data adalah tabel yang berisi deskripsi dari data yang mengalir pada DFD, Penjelasan struktur data dapat berupa field tiap data harus sama dengan yang sudah dimodelkan di ERD [12]. Berikut adalah Tabel II.9 Contoh Kamus Data [12].

Tabel II.4 Contoh Kamus Data [12]

Nama aliran data	Data Siswa Nama dari data yang digunakan
Where used/How used	Daftar pada poin proses apa data digunakan
Keterangan	Uraian singkat dari data yang digunakan
Struktur Data	Daftar komponen data yang ada pada data yang digunakan
Deskripsi	Jenis data dalam representasi komputer untuk masing-masing data.

II.11 Object Oriented Analysis Design

Konsep OOAD mencakup analisis dan desain sebuah sistem dengan pendekatan objek, yaiut analisis berorientasi objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memerika requirement (syarat/keperluan) yang harus dipenuhi sebuah sistem) dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam ruang lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur software yang didasarkan pada manipulasi objek-objek sistem atau subsistem [13].

II.11.1 Unified Modelling Language (UML)

Unified Modeling Language (UML) adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain

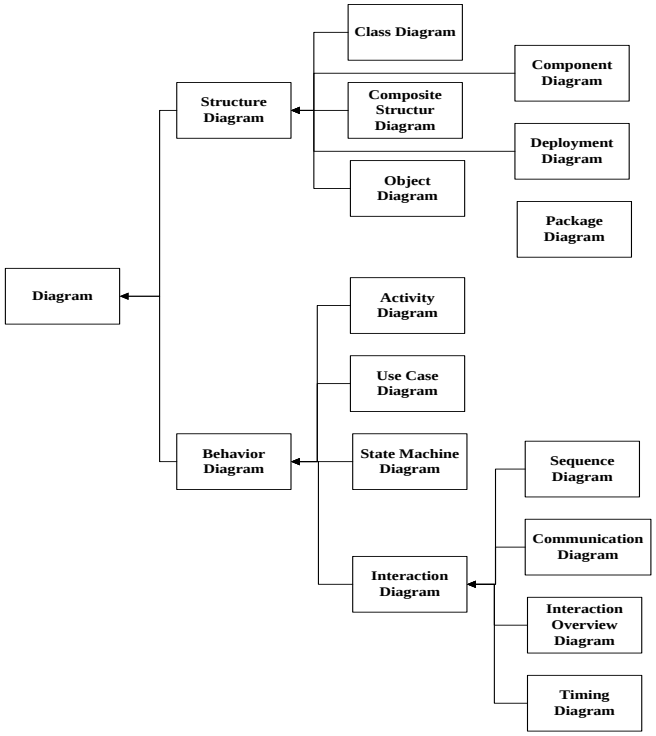
sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek (OO).

UML merupakan standar yang relatif terbuka yang dikontrol oleh Object Management Group (OMG), sebuah konsorsium terbukayang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar-standar yang mendukung interoperabilitas, khususnya interoperabilitas sistem berorientasi objek. OMG mungkin lebih dikenal dengan standar-standar CORBA (Common Object Request Broker Architecture) [14].

UML terdiri dari 13 jenis diagram resmi seperti tertulis dalam Tabel II.10 Jenis Diagram Resmi UML [14]

Tabel II.5 Jenis Diagram Resmi UML [14]

Diagram	**Kegunaan**
Activity	Behavior prosedural dan paralel
Class	Class, fitur, dan hubungan-hubungan
Communication	Interaksi antar objek; penekanan pada jalur
Component	Struktur dan koneksi komponen
Composite structure	Dekomposisis runtime sebuah class
Deployment	Pemindahan artifak ke node
Interaction overview	Campuran sequence dan activity diagram
Object	Contoh konfigurasi dari contoh-contoh
Package	Struktur hirarki compile-time
Sequence	Interaksi antar objek; penekanan pada sequence
State machine	Bagaimana event mengubah objek selama aktif
Timing	Interaksi antar objek; penekanan pada timing
Use case	Bagaimana pengguna berinteraksi dengan sebuah sistem



Gambar II.12 Klarifikasi Jenis Diagram UML [14]

Adapun penjelasan dari Class Diagram, Use Case Diagram dan Sequence Diagram adalah sebagai berikut [14]:

1. Class Diagram

Class diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai macam hubungan statis yang terdapat di antaranya. Class diagram juga menunjukan properti dan operasi sebuah class dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut. Kotak-kotak yang terdapat di dalam diagram merupakan class, yang dibagi menjadi tiga bagian : nama class (cetak tebal), atributnya dan operasinya.

a. Properti

Properti mewakili fitur-fitur struktural dari sebuah class. Properti merupakan sebuah konsep tunggal, tetapi tampak seperti dua notasi yang sedikit berbeda: atribut dan asosiasi.

b. Atribut

Notasi atribut mendeskripsikan properti dengan sebaris teks di dalam kotak class tersebut.

c. Operasi

Operasi merupakan suatu kegiatan yang dimengerti sebuah class untuk dilakukan. Operasi secara jelas berhubungan dengan metode dalam sebuah class.

2. Use Case Diagram

Use case diagram menampilkan aktor, use case dan hubungan antar keduanya :

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

3. Sequence Diagram

Sebuah sequence diagram secara khusus menjabarkan *behaviour* sebuah skenario tunggal. Diagram tersebut menunjukkan sejumlah objek contoh dan pesan-pesan yang melewati objek-objek ini di dalam use case.

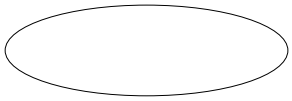
4. Activity Diagram

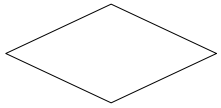
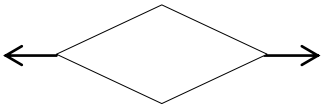
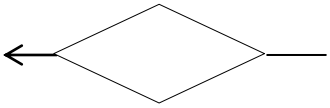
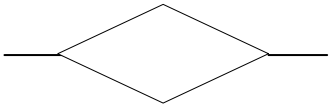
Activity diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jalur kerja. dalam beberapa hal, diagram ini seperti sebuah diagram alir, tetapi perbedaan prinsip antar diagram ini dan notasi diagram alir adalah diagram ini mendukung *behaviour* paralel.

II.12 Entity Relationship Diagram

Entity Relationship Diagram (ERD) adalah sekumpulan cara atau peralatan untuk mendeskripsikan data-data atau objek-objek yang dibuat berdasarkan dan berasal dari dunia nyata yang disebut entitas (entity) serta hubungan (relationship) antar entitas-entitas tersebut dengan menggunakan beberapa notasi [17]. Komponen-komponen pembentuk ERD dapat di lihat pada Tabel II.11 Komponen-komponen ERD [15].

Tabel II.6 Komponen-komponen ERD [15]

Notasi	Komponen	Keterangan
	Entitas	Individu yang mewakili suatu objek dan dapat dibedakan dengan objek yang lain
	Atribut	Properti yang dimiliki oleh suatu entitas, dimana dapat mendeskripsikan karakteristik darri entitas tersebut

Notasi	Komponen	Keterangan
	Relasi	Menunjukkan hubungan diantara sejumlah entitas yang berbeda
	Relasi 1 : 1	Relasi yang menunjukkan bahwa setiap entitas pada himpunan entitas pertama berhubungan dengan paling banyak satu entitas pada himpunan entitas kedua
	Relasi 1 : N	Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak atau sebaliknya. Setiap entitas dapat berelasi dengan banyak entitas pada himpunan entitas yang lain
	Relasi N : N	Hubungan ini menunjukkan bahwa setiap entitas yang pertama dapat berhubungan dengan banyak entitas pada himpunan entitas yang kedua, demikian juga sebaliknya

II.13 Java

Java menurut definisi dari Sun adalah nama untuk sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer standalone ataupun pada lingkungan jaringan. Java dikembangkan pada bulan Agustus 1991, dengan nama semula Oak. Pada Januari 1995, karena nama Oak dianggap kurang komersial, maka diganti menjadi Java. Pada Desember 1998, Sun memperkenalkan nama “Java 2” (J2) sebagai generasi kedua dari java platform. Konvensi nama baru ini diterapkan untuk semua edisi Java yaitu Standard Edition (J2SE), Enterprise Edition (J2EE), dan Micro Edition (J2ME) [16].

Ada tiga platform Java yang masing-masing diarahkan untuk tujuan tertentu dan untuk lingkungan komputasi yang berbeda-beda:

1. Standard Edition (J2SE) : J2SE merupakan inti dari bahasa pemrograman Java. J2SE didesain untuk jalan pada komputer desktop dan komputer workstations.

2. Enterprise Edition (J2EE): Dengan built-in mendukung untuk servlets. JSP, dan XML, edisi ini ditujukan untuk aplikasi berbasis server.
3. Micro Edition (J2ME) : Didesain untuk piranti dengan memori terbatas, layar display terbatas dan power pemrosesan yang juga terbatas.

II.13.1 Kelebihan Java

Adapun beberapa kelebihan dari bahasa pemrograman Java adalah sebagai berikut [16] :

1. Sederhana dan Ampuh

Java dirancang untuk mudah dipelajari, terutama bagi programmer yang telah mengenal C/C++ akan mudah sekali untuk berpindah ke Java. Pemakai dapat belajar membuat program dengan Java secara cepat jika telah memahami konsep dasar pemrograman berorientasi objek. Java tidak memiliki hal-hal yang mengejutkan dan aneh. Java memberi anda kemampuan untuk menuangkan semua ide, karena bahasa pemrograman ini bukan merupakan scripting language (bahasa naskah) yang menghilangkan kemampuan kita untuk berinovasi, tetapi dengan cara berorientasi objek yang mudah dan jelas.

2. Aman

Java dirancang sebagai bahasa pemrograman yang handal dan aman. Aplikasi-aplikasi yang dibangun dengan bahasa Java sangat handal dengan manajemen memori yang bagus. Aplikasi Java juga dikenal sangat secure, yaitu kasus-kasus seperti buffer everflow yang umumnya menjadi lubang keamanan aplikasi-aplikasi berbasis C/C++ tidak terjadi di Java, karena pengaturan securitynya yang bagus.

3. Berorientasi Objek

Paradigma pemrograman berorientasi objek merupakan paradigma pemrograman masa depan. Java merupakan bahasa pemrograman berorientasi objek. Java bukan turunan langsung dari bahasa pemrograman manapun, juga

sama sekali tidak kompetibel dengan semuanya. Java memiliki keseimbangan, Java secara cepat jika telah memahami konsep dasar pemrograman berorientasi objek. Java tidak memiliki hal-hal yang mengejutkan dan aneh. Java memberi anda kemampuan untuk menuangkan semua ide, karena bahasa pemrograman menyediakan mekanisme peng-class-an sederhana, dengan model antar muka dinamik yang intuitif hanya jika diperlukan.

4. Kokoh

Java membatasi anda dari beberapa hal kunci supaya anda dapat menemukan kesalahan lebih cepat saat mengembangkan program. Java langsung memeriksa program saat anda menuliskannya, dan sekali lagi ketika program di jalankan. Karena Java adalah bahasa yang sangat ketat dalam hal tipe data dan deklarasi, banyak kesalahan umum terjadi saat kompilasi. Hal ini akan lebih menghemat waktu jika dibandingkan dengan keharusan menjalankan program terlebih dahulu dan memeriksa semua bagian program untuk melihat ketidakcocokan dinamis selama program berjalan. Ini adalah contoh di mana Java lebih luwes dan kokoh dari beberapa bahasa lain, tetapi dengan imbalan yang layak untuk kelebihan itu.

5. Interaktif

Java memiliki beberapa kemampuan yang memungkinkan program melakukan beberapa hal pada saat bersamaan, tanpa harus kesulitan menangani proses yang akan terjadi selanjutnya. Jalinan program-program Java yang mudah digunakan memungkinkan kita untuk memikirkan pembuatan perilaku khusus, tanpa harus mengintegrasikan perilaku tersebut dengan model pemrograman global yang mengatur perulangan kejadian.

6. Netral terhadap Berbagai Arsitektur

Java telah mengambil beberapa keputusan yang sulit dalam pembuatan bahasa Java dan bagaimana program dijalankan, jadi anda dapat sepenuhnya percaya “tulis sekali, jalan di mana saja, kapan saja, selamanya”.

7. Terintrepentasi dan Berkinerja Tinggi

Java dilengkapi keajaiban lintas platform yang luar biasa dengan kompilasi ke dalam representasi langsung yang disebut kode-byte Java (Java byte-code), yang dapat diterjemahkan oleh sistem manapun yang memilki program Java didalamnya. Java, bagaimanapun dirancang untuk tetap berkinerja baik pada CPU yang tidak terlalu kuat. Walaupun Java merupakan bahasa terinterpretasi, kodekode Java telah dirancang dengan hati-hati sehingga mudah diterjemahkan ke dalam bahasa asli suatu mesin untuk menghasilkan kinerja yang tinggi.

Sistem program Java yang melakukan optimasi tepat waktu tersebut tidak kehilangan keuntungan dari program yang netral terhadap platform. “lintas platform berkinerja tinggi” bukan sekedar omong-kosong. Dalam aplikasi Java (*.class) merupakan Java bytecode yang berjalan di atas jvm (Java Virtual Machine), yang kemudian jvm-lah yang akan menginterpresentasikan kode-kode tersebut ke kode native atau kode mesin dari arsitektur yang bersangkutan. Hal sangat menarik karena urusan arsitektur mesin bukan jadi masalah bagi programmer tapi menjadi urusan kompiler pada bahasa pemrograman Java.

II.14 Processor Hypertext Protocol (PHP)

PHP adalah bahasa server-side –scripting yang menyatu dengan HTML untuk membuat halaman web yang dinamis. Karena PHP merupakan server-side-scripting maka sintaks dan perintah-perintah PHP akan diesksekusi diserver kemudian hasilnya akan dikirimkan ke browser dengan format HTML.

Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh user sehingga keamanan halaman web lebih terjamin. PHP dirancang untuk membuat halaman web yang dinamis, yaitu halaman web yang dapat membentuk suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman web [17].

II.15 MySQL

MySQL adalah salah satu jenis database server yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan database sebagai sumber dan pengolahan datanya. MySQL dikembangkan oleh perusahaan swedia bernama MySQL AB yang pada saat ini bernama Tcx DataKonsult AB sekitar tahun 1994-1995, namun cikal bakal kodenya sudah ada sejak tahun 1979. Awalnya Tcx merupakan perusahaan pengembang software dan konsultan database, dan saat ini MySQL sudah diambil alih oleh Oracle Corp.

Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses databasenya sehingga mudah untuk digunakan, kinerja query cepat, dan mencukupi untuk kebutuhan database perusahaan-perusahaan yang berskala kecil sampai menengah, MySQL juga bersifat open source (tidak berbayar). MySQL merupakan database yang pertama kali didukung oleh bahasa pemrograman script untuk internet (PHP dan Perl). MySQL dan PHP dianggap sebagai pasangan software pembangun aplikasi web yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman script PHP [17].

II.16 Google Maps Directions API

Google Maps Directions API adalah layanan yang menghitung arah antar lokasi menggunakan permintaan HTTP.

Anda bisa mencari arah untuk beberapa moda transportasi, termasuk angkutan umum, mengemudi, berjalan atau bersepeda. Arah bisa menetapkan tempat asal, tujuan dan titik jalan baik sebagai string teks (misalnya "Chicago, IL" atau "Darwin, NT, Australia") atau sebagai koordinat garis lintang/garis bujur. Directions API bisa mengembalikan arah multi-bagian menggunakan serangkaian titik jalan.

Layanan ini biasanya didesain untuk menghitung arah alamat statis (sudah diketahui sebelumnya) untuk penempatan konten aplikasi pada peta; layanan ini

tidak didesain untuk merespons input pengguna secara real-time, misalnya. Untuk perhitungan arah dinamis (misalnya, dalam elemen antarmuka pengguna), lihat dokumentasi untuk Google Maps JavaScript API Layanan Directions[18].

a. Permintaan Arah

Permintaan Google Maps Directions API mengambil bentuk berikut:

https://maps.googleapis.com/maps/api/directions/output?parameters

dalam hal ini, output bisa berupa salah satu nilai berikut:

- json (disarankan) menunjukkan output dalam JavaScript Object Notation (JSON)
- xml menunjukkan output berupa XML

Untuk mengakses Google Maps Directions API melalui HTTP, gunakan:

http://maps.googleapis.com/maps/api/directions/output?parameters

HTTP disarankan untuk aplikasi yang berisi data pengguna sensitif, seperti lokasi pengguna, dalam permintaan.

URL Google Maps Directions API dibatasi sekitar 2000 karakter, setelah Pengkodean URL. Karena sebagian URL Google Maps Directions API bisa melibatkan banyak lokasi sepanjang lintasan, berhati-hatilah dengan batas ini saat membangun URL Anda.

b. Parameter Permintaan

Beberapa parameter tertentu diperlukan sementara yang lainnya bersifat opsional. Sebagaimana standar dalam URL, semua parameter dipisah menggunakan karakter ampersand (&). Daftar parameter dan kemungkinan nilainya disebutkan di bawah ini.

c. Parameter yang diperlukan

- origin — Alamat, nilai garis lintang/garis bujur tekstual, atau ID tempat asal yang ingin Anda hitung arahnya.
- Jika Anda meneruskan sebuah alamat sebagai string, layanan Directions akan melakukan geocode atas string itu dan mengubahnya menjadi koordinat garis lintang/garis bujur untuk menghitung arah. Koordinat ini mungkin berbeda dengan yang dikembalikan oleh Google Maps Geocoding API, misalnya pintu masuk bangunan dan bukan pusatnya.
- Jika Anda meneruskan koordinat, itu akan digunakan tanpa diubah untuk menghitung arah. Pastikan tidak ada spasi di antara nilai garis lintang dan garis bujur.
- ID Tempat harus diawali dengan place_id:. ID tempat hanya bisa ditetapkan jika permintaan menyertakan kunci API atau ID klien Google Maps API for Work. Anda bisa mendapatkan ID tempat dari Google Maps Geocoding API dan Google Places API (termasuk Place Autocomplete). Untuk contoh menggunakan ID tempat dari Place Autocomplete, lihat Place Autocomplete dan Arah. Untuk informasi selengkapnya tentang ID tempat, lihat ikhtisar ID tempat.
- destination — Alamat, nilai garis lintang/garis bujur tekstual, atau ID tempat tujuan yang ingin Anda hitung arahnya. Opsi untuk parameter destination sama dengan opsi untuk parameter origin yang dijelaskan di atas.
- key — Kunci API aplikasi Anda. Kunci ini mengidentifikasi aplikasi Anda untuk keperluan manajemen kuota. Pelajari cara mendapatkan kunci.

d. Mode Perjalanan

Saat menghitung arah, Anda bisa menetapkan mode transportasi yang akan digunakan. Secara default, arah dihitung sebagai arah driving. Mode perjalanan berikut ini telah didukung:

- driving (default) menunjukkan arah mengemudi standar yang menggunakan jaringan jalan raya.
- walking meminta arah berjalan melalui jalur pejalan kaki & trotoar (bila tersedia).
- bicycling meminta arah bersepeda melalui lintasan sepeda & jalan yang disukai (bila tersedia).
- transit meminta arah melalui rute angkutan umum (bila tersedia). Jika mengatur mode ke transit, Anda juga bisa menetapkan departure_time atau arrival_time. Jika tidak ada waktu yang ditetapkan, default departure_time adalah sekarang (yaitu, waktu keberangkatan default adalah waktu saat ini). Anda juga bisa menyertakan transit_mode dan/atau transit_routing_preference.

e. Titik jalan

Saat menghitung rute menggunakan Google Maps Directions API, Anda juga bisa menetapkan titik jalan untuk arah mengemudi, berjalan atau bersepeda (titik jalan tidak tersedia untuk arah angkutan umum). Titik jalan memungkinkan Anda menghitung rute melalui lokasi tambahan, di mana rute yang dikembalikan menyertakan persinggahan di setiap titik jalan yang diberikan.

Titik jalan ditetapkan dalam parameter waypoints dan terdiri dari satu atau beberapa alamat atau lokasi yang dipisah dengan karakter pipa (|).

Misalnya, URL berikut membuat permintaan Directions untuk rute antara Boston, MA dan Concord, MA dengan persinggahan di Charlestown dan Lexington, dalam urutan itu:

https://maps.googleapis.com/maps/api/directions/json?origin=Boston,MA&destination=Concord,MA&waypoints=Charlestown,MA|Lexington,MA&key=YOUR_API_KEY

Untuk setiap titik jalan dalam permintaan, respons arah menyertakan entri tambahan dalam larik legs guna memberikan detail yang sesuai untuk segmen perjalanan itu.

Jika ingin memengaruhi rute menggunakan titik jalan tanpa menambahkan persinggahan, tambahkan awalanvia: pada titik jalan tersebut. Titik jalan yang diberi awalan via: tidak akan menambahkan entri ke larik legs, namun akan mengarahkan perjalanan melalui titik jalan yang diberikan.

URL berikut memodifikasi permintaan sebelumnya yang perjalanannya diarahkan melewati Lexington tanpa berhenti:

https://maps.googleapis.com/maps/api/directions/json?origin=Boston,MA&destination=Concord,MA&waypoints=Charlestown,MA|via:Lexington,MA&key=YOUR_API_KEY

Awalan via: sangat efektif saat membuat rute sebagai respons terhadap pengguna yang menyeret titik jalan pada peta. Hal itu memungkinkan pengguna melihat bagaimana tampilan rute akhir secara real-time dan membantu memastikan titik-titik jalan tersebut ditempatkan di lokasi yang bisa diakses oleh Google Maps Directions API.

f. Optimalkan titik jalan Anda

Secara default, layanan Directions menghitung rute melalui titik jalan yang diberikan dalam urutan tertentu. Atau, Anda bisa meneruskan optimize:true sebagai argumen pertama dalam parameter waypoints untuk memungkinkan layanan Directions mengoptimalkan rute yang disediakan dengan menata ulang titik jalan dalam urutan yang lebih efisien. (Optimalisasi ini merupakan aplikasi dari Travelling Salesman Problem.)

Jika Anda memerintahkan layanan Directions untuk mengoptimalkan urutan titik jalan, maka urutannya akan dikembalikan dalam bidang waypoint_order di dalam objek routes. Bidang waypoint_order mengembalikan nilai yang berbasis nol.

Contoh berikut menghitung rute perjalanan darat dari Adelaide, Australia Selatan ke setiap wilayah penghasil wine utama Australia Selatan dengan menggunakan optimalisasi rute.

https://maps.googleapis.com/maps/api/directions/json?origin=Adelaide,SA&destination=Adelaide,SA&waypoints=optimize:true|Barossa+Valley,SA|Clare,SA|Connawarra,SA|McLaren+Vale,SA&key=YOUR_API_KEY

Pemeriksaan rute yang dihitung akan menunjukkan rute tersebut dihitung menggunakan urutan titik jalan berikut:

```
"waypoint_order": [ 1, 0, 2, 3 ]
```

g. Pembatasan

Arah bisa dihitung dengan mengikuti pembatasan tertentu. Pembatasan ditunjukkan melalui penggunaan parameter avoid, dan sebuah argumen ke parameter itu yang menunjukkan pembatasan untuk avoid. Pembatasan berikut ini telah didukung:

- avoid=tolls
- avoid=highways
- avoid=ferries

Bisa saja meminta rute yang menghindari kombinasi tol, jalan raya dan feri dengan meneruskan kedua pembatasan itu ke parameter avoid. Misalnya: avoid=tolls|highways|ferries.

h. Sistem Satuan

Hasil arah berisi text dalam bidang distance yang bisa ditampilkan kepada pengguna untuk menunjukkan jarak "langkah" tertentu pada rute tersebut. Secara default, teks ini menggunakan sistem satuan pada negara atau wilayah asal.

Misalnya, rute dari "Chicago, IL" ke "Toronto, ONT" akan menampilkan hasil dalam mil, sedangkan rute terbalik akan menampilkan hasil dalam kilometer. Anda bisa mengesampingkan sistem satuan ini dengan menetapkannya secara eksplisit dalam parameter units permintaan, dengan meneruskan salah satu dari nilai berikut:

- metric menetapkan penggunaan sistem metrik. Jarak tekstual dikembalikan menggunakan kilometer dan meter.
- imperial menetapkan penggunaan sistem Imperial (Inggris). Jarak tekstual dikembalikan menggunakan mil dan kaki.

i. Mencondongkan Wilayah

Anda juga bisa mengatur layanan Directions untuk mengembalikan hasil yang dicondongkan ke wilayah tertentu dengan menggunakan parameter region. Parameter ini mengambil argumen ccTLD (top-level domain kode negara) yang menetapkan kecondongan wilayah. Umumnya kode ccTLD identik dengan kode ISO 3166-1, dengan beberapa pengecualian. Misalnya, ccTLD Kerajaan Inggris adalah "uk" (.co.uk) sementara kode ISO 3166-1 untuknya adalah "gb" (secara teknis untuk entitas "Kerajaan Bersatu Britania Raya dan Irlandia Utara").

Anda bisa memanfaatkan domain apa saja tempat aplikasi Google Maps utama menjalankan arah mengemudi.

Misalnya, permintaan arah dari "Toledo" ke "Madrid" akan mengembalikan hasil bila region diatur ke es karena "Toledo" ditafsirkan sebagai kota Spanyol:

```
https://maps.googleapis.com/maps/api/directions/json?origin=To
ledo&destination=Madrid&region=es&key=YOUR_API_KEY
```

```
{
  "status": "OK",
  "routes": [ {
    "summary": "AP-41",
    "legs": [ {
        ...
    } ],
    "copyrights": "Map data ©2010 Europa Technologies, Tele
Atlas",
    "warnings": [ ],
    "waypoint_order": [ ]
  } ]
}
```

Arah untuk "Toledo" ke "Madrid" yang dikirim tanpa parameter region tidak akan mengembalikan hasil, karena "Toledo" ditafsirkan sebagai kota di Ohio:

```
https://maps.googleapis.com/maps/api/directions/json?origin=To
ledo&destination=Madrid&key=YOUR_API_KEY
```

```
{
```

```
  "status": "ZERO_RESULTS",

  "routes": [ ]

}
```

j. Respons Arah

Respons arah dikembalikan dalam format yang ditunjukkan oleh flag output dalam path permintaan URL.

k. Respons Contoh

Contoh permintaan HTTP ditampilkan di bawah ini, yang menghitung rute dari Chicago, IL ke Los Angeles, CA melewati dua titik jalan di Joplin, MO dan Oklahoma City, OK.

https://maps.googleapis.com/maps/api/directions/json?origin=Chicago,IL&destination=Los+Angeles,CA&waypoints=Joplin,MO|Oklahoma+City,OK&key=YOUR_API_KEY

Contoh di atas meminta output JSON. Bisa juga meminta output XML. Klik tab di bawah ini untuk melihat contoh respons JSON dan XML.

Karena hasil arah bisa sangat detail, maka elemen berulang dalam respons telah dihilangkan agar lebih jelas.

```
JSON

{
  "status": "OK",
  "geocoded_waypoints" : [
     {
        "geocoder_status" : "OK",
```

```
         "place_id" : "ChIJ7cv00DwsDogRAMDACa2m4K8",
         "types" : [ "locality", "political" ]
      },
      {
         "geocoder_status" : "OK",
         "place_id" : "ChIJ69Pk6jdlyIcRDqM1KDY3Fpg",
         "types" : [ "locality", "political" ]
      },
      {
         "geocoder_status" : "OK",
         "place_id" : "ChIJgdL4flSKrYcRnTpP0XQSojM",
         "types" : [ "locality", "political" ]
      },
      {
         "geocoder_status" : "OK",
         "place_id" : "ChIJE9on3F3HwoAR9AhGJW_fL-I",
         "types" : [ "locality", "political" ]
      }
   ],
   "routes": [ {
     "summary": "I-40 W",
     "legs": [ {
       "steps": [ {
         "travel_mode": "DRIVING",
         "start_location": {
           "lat": 41.8507300,
           "lng": -87.6512600
         },
         "end_location": {
           "lat": 41.8525800,
           "lng": -87.6514100
         },
         "polyline": {
           "points": "a~l~Fjk~uOwHJy@P"
         },
         "duration": {
```

```
          "value": 19,
          "text": "1 min"
        },
        "html_instructions": "Head
\u003cb\u003enorth\u003c/b\u003e on \u003cb\u003eS Morgan
St\u003c/b\u003e toward \u003cb\u003eW Cermak
Rd\u003c/b\u003e",
        "distance": {
          "value": 207,
          "text": "0.1 mi"
        }
      },
      ...
      ... additional steps of this leg
    ...
    ... additional legs of this route
      "duration": {
        "value": 74384,
        "text": "20 hours 40 mins"
      },
      "distance": {
        "value": 2137146,
        "text": "1,328 mi"
      },
      "start_location": {
        "lat": 35.4675602,
        "lng": -97.5164276
      },
      "end_location": {
        "lat": 34.0522342,
        "lng": -118.2436849
      },
      "start_address": "Oklahoma City, OK, USA",
      "end_address": "Los Angeles, CA, USA"
    } ],
    "copyrights": "Map data ©2010 Google, Sanborn",
```

```
    "overview_polyline": {
      "points":
"a~l~Fjk~uOnzh@vlbBtc~@tsE`vnApw{A`dw@~w\\|tNtqf@l{Yd_Fblh@rxo
@b}@xxSfytAblk@xxaBeJxlcBb~t@zbh@jc|Bx}C`rv@rw|@rlhA~dVzeo@vrS
nc}Axf]fjz@xfFbw~@dz{A~d{A|zOxbrBbdUvpo@`cFp~xBc`Hk@nurDznmFfw
Mbwz@bbl@lq~@loPpxq@bw_@v|{CbtY~jGqeMb{iF|n\\~mbDzeVh_Wr|Efc\\
x`Ij{kE}mAb~uF{cNd}xBjp]fulBiwJpgg@|kHntyArpb@bijCk_Kv~eGyqTj_
|@`uV`k|DcsNdwxAott@r}q@_gc@nu`CnvHx`k@dse@j|p@zpiAp|gEicy@`om
FvaErfo@igQxnlApqGze~AsyRzrjAb__@ftyB}pIlo_BflmA~yQftNboWzoAlz
p@mz`@|}_@fda@jakEitAn{fB_a]lexClshBtmqAdmY_hLxiZd~XtaBndgC"
    },
    "warnings": [ ],
    "waypoint_order": [ 0, 1 ],
    "bounds": {
      "southwest": {
        "lat": 34.0523600,
        "lng": -118.2435600
      },
      "northeast": {
        "lat": 41.8781100,
        "lng": -87.6297900
      }
    }
  } ]
}
```


Biasanya, hanya dihasilkan satu entri dalam larik routes untuk pencarian arah, meskipun layanan Directions bisa mengembalikan beberapa rute jika Anda meneruskan alternatives=true.

II.17 Andorid Studio

Android Studio adalah *Integrated Development Environment* (IDE) yang resmi untuk pengembangan aplikasi Android, berdasarkan *IntelliJ IDEA*. Dengan kemampuan pengeditan kode dan alat – alat pengembang, Android Studio

menawarkan lebih banyak fitur yang meningkatkan produktivitas ketika membangun aplikasi Android, seperti [19]:

a. Berbasis Gradle yang fleksibel untuk membangun sistem
b. Sebuah emulator cepat dan kaya fitur
c. Lingkungan terpadu yang dimana dapat mengembangkan untuk semua perangkat Android
d. *Instant Run* untuk mendorong perubahan pada aplikasi berjalan tanpa membangun APK baru
e. Template kode dan integrasi *GitHub* untuk membantu Anda membangun fitur aplikasi umum dan contoh kode impor
f. Alat pengujian ekstensif dan kerangka kerja
g. *Lint tools* untuk menangkap kinerja, kegunaan, kompatibilitas dengan versi android, dan masalah lainnya.
h. C ++ dan dukungan NDK.
i. *Built-in* dukungan untuk *Google Cloud Platform*, sehingga mudah untuk mengintegrasikan *Google Cloud Messaging* dan *App Engine*