BAB II
TINJAUAN PUSTAKA

II.1. Bekam Ruqyah Center (BRC)

BRC (Bekam *Ruqyah Center*) merupakan Balai Pengobatan *thibbun nabawi* terbesar dengan memiliki lebih dari 48 cabang tersebar di seluruh indonesia, dan telah menangani sedikitnya 25.000 pasien per bulan. Saat ini BRC mempunyai banyak jenis pelayanan pengobatan yang dapat dirasakan oleh para pasien pada di setiap cabang khusus nya di kota Bandung. Di kota Bandung, BRC mempunyai 10 cabang dan tersebar di setiap penjuru kota Bandung. Jenis layanan dan alamat cabang dapat di lihat sebagai berikut :

II.1.1. Jenis Layanan dan Harga

1. Bekam : Rp 65.000 (30 Menit)
2. Ruqyah dan Diagnosa : Rp 40.000 (30 Menit)
3. Terapi Lintah : Rp 40.000 / ekor (30 Menit)
4. Terapi Detox : Rp 40.000 (15 Menit)
5. *Ear Candle* : Rp 65.000 (30 Menit)
6. Terapi Totok Wajah : Rp 80.000 (45 Menit)
7. Pijat Kebugaran : Rp 65.000 (30 Menit)
8. Bekam Estetika : Rp 65.000 (30 Menit)
9. *Body Spa* : Rp 80.000 (30 Menit)

II.1.2. Alamat Cabang dan Waktu Operasinoal

1. BRC Antapani (10 a.m. – 5.p.m)
   Jl. Puwarkata no.152 (100 m samping kanan Toserba Griya Antapani), Bandung. Telp. 0851 0055 3442

2. BRC Buah Batu (10 a.m. – 5.p.m)
   Jl.Terusan Buah Batu no. 280 (50 m samping kanan pintu tol Buah Batu), Bandung. Telp.0851 0081 0149
3. BRC Cibaduyut (10 a.m. – 5.p.m)
   Jl. Cibaduyut Raya no. 7 (100 m dari terminal Leuwi Panjang), Bandung. Telp. 0851 0505 7600
4. BRC Dago (10 a.m. – 5.p.m)
   Jl. Ir. H. Juanda no. 479A, Bandung (Dago Atas, 300 m dari terminal Dago) Telp. 022 2533 496
5. BRC Gerlong (10 a.m. – 5.p.m)
   Jl.Gegerkalong Girang no. 54, Bandung (100 m dari Masjid Daarut Tauhid / samping Indomart) Telp. 022 200 8187 / 0815 7378 2426
6. BRC Margacinta (10 a.m. – 5.p.m)
   Jl. Ciwastra no. 181 (5 m seberang Taman Pluto Margahayu Raya), Bandung Telp. 0851 0178 0415
7. BRC Padjajaran (10 a.m. – 5.p.m)
   Jl. Pajajaran Belakang no. 102B Rt 03 Rw 07, Bandung (seberang Bank Mandiri) Telp. 0853 2030 6259
8. BRC Riung Bandung (10 a.m. – 5.p.m)
   Jl. Raya Cipamokolan no. 28, Bandung (100 m dari Borma Riung Bandung) Telp. 0851 0307 3895
9. BRC Suci (10 a.m. – 5.p.m)
   Jl. PHH Mustofa (Suci) Komplek Surapati Core Blok C-23. Bandung (Belakang Radio Mora FM/300 m sebelum Terminal Cicaheum) Telp. 0853 2051 9977
10. BRC Ujung Berung (10 a.m. – 5.p.m)
    Komplek Sukup Baru No. 4, Ujung Berung-Bandung (Belakang Pegadaian Cijambe) Telp. 0851 0809 0972

II.2. Reservasi

Pemesanan dalam bahasa Inggris adalah *reservation* yang berasal dari kata *to reserve* yaitu menyediakan atau mempersiapkan tempat sebelumnya. Sedangkan *reservation* yaitu pemesanan suatu tempat fasilitas.

Pengertian reservasi adalah sebuah proses perjanjian berupa pemesanan sebuah produk baik barang maupun jasa dimana pada saat itu telah terdapat kesepahaman antara konsumen dengan produsen mengenai produk tersebut namun belum ditutup oleh sebuah transaksi jual beli. Pada saat reservasi berlangsung biasanya ditandai dengan adanya proses tukar-menukar informasi antara konsumen dan produsen agar kesepahaman mengenai produk dapat terwujud.

Alasan reservasi menjadi sebuah media yang sangat efektif baik bagi produsen maupun bagi konsumen adalah produsen akan dapat melakukan evaluasi terhadap produk yang akan mereka jual melalui tingkat tinggi rendahnya jumlah reservasi jauh sebelum produk tersebut dijual (barang) ataupun diselenggarakan (jasa), dimana hasil evaluasi tersebut akan membantu produsen untuk menentukan langkah pemasaran yang akan diambil terhadap produk yang akan dijual tersebut.

Sedangkan bagi konsumen melalui media reservasi dapat menimbang terlebih dahulu sebelum membeli produk sampai dengan waktu yang telah ditentukan (*time limit*). Sampai dengan *time limit* yang telah ditentukan produk yang telah dipesan tersebut tidak boleh dijual kepada konsumen lain, karena secara tertulis maupun tidak sampai dengan *time limit* produk tersebut telah dipriotaskan kepada konsumen yang telah melakukan reservasi. Hal ini akan sangat menguntungkan konsumen karena sampai dengan *time limit* yang ditentukan, konsumen dapat melakukan perbandingan dengan produk lain. Selain itu, jika terjadi sesuatu dan lain hal yang mengharuskan konsumen untuk batal membeli produk tersebut, konsumen yang bersangkutan tidak harus membeli produk tersebut.

Layaknya perjanjian lainnya, reservasi akan dinyatakan batal apabila terjadi beberapa hal berikut [2]:

1. Terjadi sesuatu di luar kendali manusia (bencana alam, perang, dll) sehingga produk tidak dapat dibuat atau diselenggarakan oleh produsen

2. Konsumen melakukan pembatalan sebelum batas waktu
3. Terdapat kesalahan pengisian data penting mengenai konsumen pada saat proses reservasi sehingga produsen harus membatalkan reservasi (catatan: tergantung kebijakan masing-masing perusahaan)
4. Tidak dilakukan transaksi terhadap produk yang dipesan sampai dengan *time limit* yang telah ditentukan
5. Baik produsen maupun konsumen melakukan hal-hal yang menyebabkan reservasi harus dibatalkan, tergantung perjanjian yang dibuat/ditentukan.

II.3. Aplikasi

Aplikasi berasal dari kata *application* yaitu bentuk sebuah kata benda dari kata kerja *to apply* yang dalam bahasa Indonesia berarti pengolah. Secara istilah, aplikasi komputer adalah suatu subkelas perangkat lunak komputer yang menggunakan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pemakai. Contoh utama perangkat lunak aplikasi adalah program pengolah kata, lembar kerja, pemutar media, dan program pendukung komputer lainnya

Kumpulan aplikasi komputer yang digabung menjadi suatu paket biasanya disebut paket atau *suite* aplikasi (*application suite*). Aplikasi-aplikasi dalam suatu paket biasanya memiliki antarmuka pengguna yang memiliki kesamaan sehingga memudahkan pengguna untuk mempelajari dan menggunakan tiap aplikasi. Umumnya aplikasi-aplikasi tersebut memiliki kemampuan untuk saling berinteraksi sehingga menguntungkan pemakai. Contohnya, suatu lembar kerja dapat dimasukkan dalam suatu dokumen pengolah kata walaupun dibuat pada aplikasi lembar kerja yang terpisah [3].

II.4. Sistem Informasi

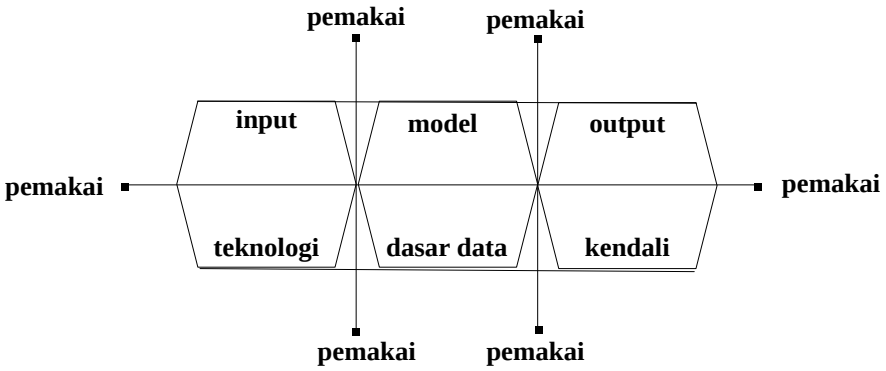
Sistem adalah kumpulan dari komponen-komponen peralatan model requirements, function dan interface. Sistem informasi merupakan kombinasi teratur apapun dari sumber daya manusia, hardware, software, data, dan jaringan. Daur hidup suatu sistem meliputi [4]:

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 (controls block). Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lainnya membentuk satu kesatuan untuk mencapai sasarannya [5].



Gambar II.1. Blok Sistem Informasi yang Berinteraksi [5]

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. Global Positioning Sistem (GPS)

GPS adalah singkatan dari *Global Positioning Sistem*, merupakan sistem satelit navigasi dan penentuan posisi yang dimiliki dan dikelola oleh Amerika Serikat. Sistem ini didesain untuk memberikan posisi dan kecepatan tiga dimensi serta informasi mengenai waktu, secara terus-menerus di seluruh dunia tanpa tergantung waktu dan cuaca.

Pada dasarnya GPS terdiri atas 3 segmen utama, yaitu segmen angkasa (*space segment*) yang terutama terdiri dari satelit-satelit GPS, segmen sistem control (*control sistem segment*) yang terdiri dari stasiun-stasiun pemonitor dan pengontrol satelit, dan segmen pemakai (*user segment*) yang terdiri dari pemakai GPS termasuk alat-alat penerima dan pengolah sinyal dan data GPS.

Pada saat ini, sistem GPS sudah sangat banyak digunakan orang di seluruh dunia dalam berbagai bidang aplikasi untuk penentuan posisi. Hal tersebut karena GPS memiliki banyak keunggulan seperti [6]:

1. GPS dapat digunakan setiap saat tanpa tergantung cuaca dan waktu. Dapat digunakan dengan baik pada kondisi cuaca buruk sekalipun ataupun dalam waktu siang atau malam hari.
2. Satelit-satelit GPS memiliki ketinggian orbit yang cukup tiggi, yaitu sekitar 20.000 km diatas permukaan Bumi dan jumlahnya yang cukup banyak yaitu 24 satelit sehingga GPS dapat meliput wilayah dengan cakupan yang luas. Selain itu GPS dapat digunakan oleh banyak orang secara bersamaan serta pemakaiannya menjadi tidak tergantung pada batas-batas politik dan batas alam.

3. Penggunaan GPS sangat efisien dan efektif untuk diaplikasikan pada survei dan pemetaan di wilayah yang relatif luas dengan kondisi topografi yang relatif sulit seperti daerah pegunungan dan daerah rawa-rawa. Hal ini dikarenakan penggunaan GPS dalam penentuan posisi relatif tidak terlalu terpengaruh dengan kondisi topografi daerah survei. GPS hanya memerlukan saling keterlihatan antara suatu titik dengan satelit.
4. GPS dapat memberikan ketelitian posisi yang spektrumnya cukup luas. Dari yang sangat teliti (orde milimiter) sampai yang biasa-biasa saja (orde puluhan meter).
5. Pemakaian sistem GPS tidak dikenakan biaya. Selama pengguna memiliki alat penerima (*receiver*) sinyal GPS beserta perangkat keras dan lunak untuk pemrosesan datanya.

Penerapan teknologi GPS pada penilitian ini yaitu untuk mendapatkan lokasi pasien berada, memasang *marker* lokasi cabang, menunjukan rute perjalanan dari lokasi pasien ke lokasi cabang yang dituju.

II.6. Application Programming Interface (API)

Antarmuka pemrograman aplikasi (*Application Programming Interface*/API) adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan oleh *programmer* saat membangun perangkat lunak untuk sistem operasi tertentu. API memungkinkan *programmer* untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi.

API dapat menjelaskan cara sebuah tugas (*task*) tertentu dilakukan. Dalam pemrograman prosedural seperti bahasa C, aksi biasanya dilakukan dengan media pemanggilan fungsi. Karena itu, API biasanya menyertakan penjelasan dari fungsi rutin yang disediakannya.

API menyediakan fungsi dan perintah dengan bahasa yang lebih terstruktur dan lebih mudah untuk dipahami oleh *programmer* bila dibandingkan dengan Sistem *Calls*, hal ini penting untuk aspek editing dan pengembangan, sehingga

programmer dapat mengembangkan sistem dengan mudah. API juga dapat digunakan pada sistem operasi mana saja asalkan sudah ada paket-paket API nya.

Dalam contoh program sederhana, dibutuhkan setidaknya ribuan sistem *calls* per detik. Oleh karena itu kebanyakan *programmer* membuat aplikasi dengan menggunakan *Application Programming Interface* (API). Dalam API itu terdapat fungsi-fungsi/perintah-perintah untuk menggantikan bahasa yang digunakan dalam sistem *calls* dengan bahasa yang lebih terstruktur dan mudah dimengerti oleh *programmer*. Fungsi yang dibuat dengan menggunakan API tersebut kemudian akan memanggil sistem *calls* sesuai dengan sistem operasinya. Tidak tertutup kemungkinan nama dari sistem *calls* sama dengan nama di API. Keuntungan memprogram dengan menggunakan API adalah [7]:

1. Portabilitas.

   Programmer yang menggunakan API dapat menjalankan programnya dalam sistem operasi mana saja asalkan sudah ter- install API tersebut. Sedangkan sistem *call* berbeda antar sistem operasi, dengan catatan dalam implementasinya mungkin saja berbeda.
2. Lebih Mudah Dimengerti.

   API menggunakan bahasa yang lebih terstruktur dan mudah dimengerti daripada bahasa sistem *call*. Hal ini sangat penting dalam hal editing dan pengembangan.

Pada penelitian ini API mempunyai peranan sebagai sebagai jembatan penghubung dari aplikasi dengan server dan *database*. Dari API dihasilkan JSON untuk pengurain (*parsing*) data

II.7. Java Script Object Notation (JSON)

Java Script Object Notation (JSON) 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, JSON merupakan format teks yang tidak bergantung pada bahasa pemprograman apapun karena menggunakan gaya bahasa yang umum

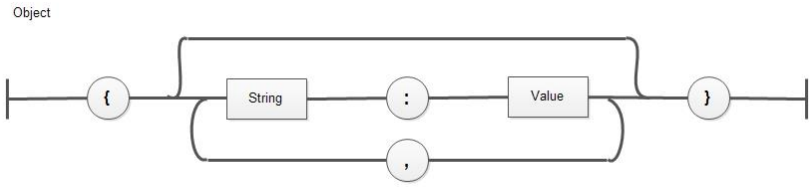
digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data.

JSON terbuat dari dua struktur:

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

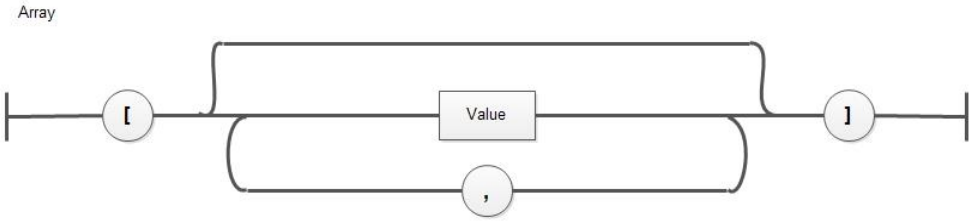
Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut [8]:

1. Objek adalah sepasang nama/nilai yang tidak terurutkan. Objek dimulai dengan "{" (kurung kurawal buka) dan diakhiri dengan "}" (kurung kurawal tutup). Setiap nama diikuti dengan ":" (titik dua) dan setiap pasangan nama/nilai dipisahkan oleh "," (koma).



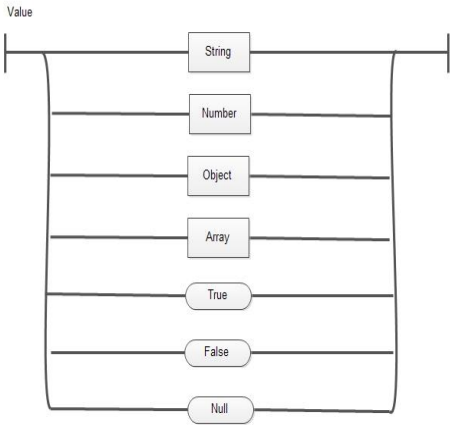
Gambar II.2. Objek

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



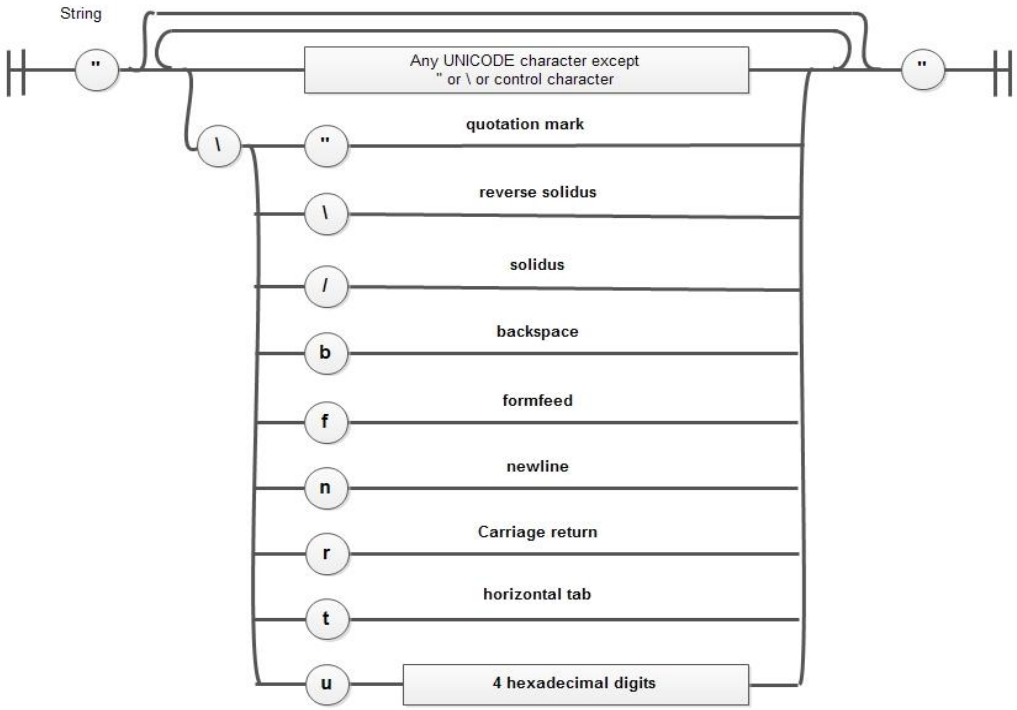
Gambar II.3. Larik

3. 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.



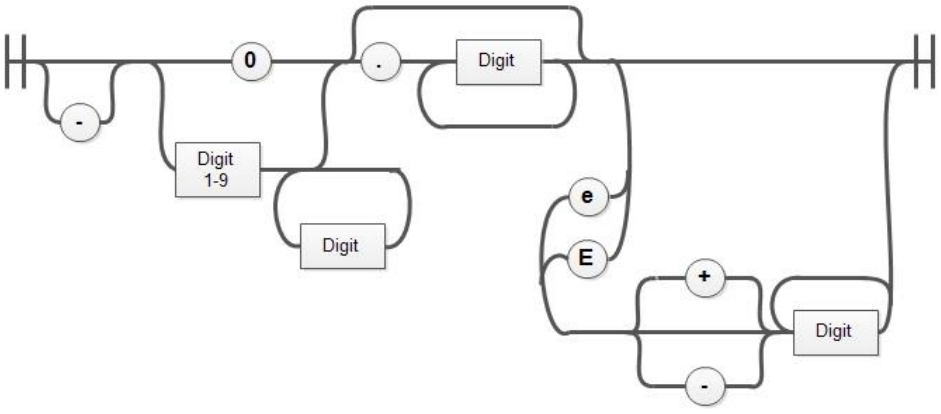
Gambar II.4. *Value*

4. 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.5. *String*

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



Gambar II.6. Angka

Pada penelitian ini, JSON mempunyai peranan sebagai pengurain (*parsing*) data supaya antara server dan aplikasi dapat melakukan komunikasi pengiriman dan penerimaan data.

II.8. Web Service

Web services adalah kumpulan dari protokol yang dapat digunakan untuk pertukaran data antar aplikasi atau sistem yang berbeda. *Web services* mudah untuk diimplementasikan dengan teknologi yang sudah ada. *Web services* mengijinkan penggunanya untuk memperoleh informasi yang mereka tuju tanpa informasi asing yang disajikan secara umum di halaman *website*.

Ada beberapa alasan untuk menggunakan *web services*, kebanyakan dapat dengan mudah dikatakan seseorang lainnya telah mendapatkan informasi yang Anda butuhkan, atau Anda memiliki informasi yang diharapkan dapat digunakan orang lain. Ketika merencanakan sebuah *web services* baru, ingat bahwa server dapat menyediakan atau menerima informasi. Ingat bahwa *web services* dapat menyediakan beberapa level dari keamanan dari sebuah *open source* secara keseluruhan menjadi *private source*.

Berikut ini adalah beberapa hal yang dapat membantu Anda bahwa *web services* adalah sebuah jalur untuk mencari hal-hal berikut [9].

1. *Web services* membuat penggunaan teknologi yang telah disebarkan. Anda telah memiliki sebuah *web* server, *database* dan sebagainya maka anda dapat memberi akses terhadap orang yang membutuhkan data tersebut.
2. *Web services* sesuai untuk skema keamanan yang telah dipakai. Penyebaran secara hati-hati akan menjaga tingkat keamanan tetap pada jalur regular pengaksesan *web*.
3. Secara teknis memahami pengguna akan menemukan sebuah jalan untuk mengakses informasi yang dituju. Tanpa *web services* maka mereka akan membutuhkan usaha yang lebih besar untuk mendapatkan informasi yang diinginkan.
4. *Web services* akan memberikan akses kepada client untuk mengintegrasikan sistem mereka dengan sistem yang dibangun

Pada Penelitian ini, *web service* mempunyai peranan untuk menangani bagian yang akan di *handle* secara otomatis oleh sistem. Contoh nya ketika ada pasien yang sudah melakukan reservasi pada sebuah cabang tetapi pada waktu nya dia tidak datang, maka secara otomatis sistem akan menghapus pasien dari data reservasi.

II.9. Mobile Application

Sistem aplikasi *mobile* merupakan aplikasi yang dapat digunakan walaupun pengguna berpindah-pindah tempat tanpa mengalami pemutusan komunikasi selama ada jaringan. Aplikasi ini bisa diakses melalui perangkat *nirkabel* seperti pager, telepon seluler, dll. Berikut ini adalah karakteristik dari perangkat *mobile* [10].

1. Perangkat *mobile* memiliki ukuran dan bentuk yang bisa dibawa kemana saja. Konsumen membutuhkan perangkat yang bisa digunakan kapan saja dan dimana saja.
2. Perangkat *mobile* memiliki *memory* yang terbatas yaitu *primary* (RAM) dan *secondary* (*memory card*).
3. Daya proses sistem *mobile* tidak setangguh *desktop*.
4. Perangkat *mobile* membutuhkan daya lebih rendah dibandingkan dengan perangkat *desktop*.
5. Kuat dan dapat diandalkan, karena bisa dibawa kemana saja dan tahan terhadap benturan-benturan normal dan percikan air dalam batas wajar.
6. Konektivitas yang terbatas, perangkat mobile memiliki *bandwith* rendah

Aplikasi yang akan dibangun pada penelitian ini merupakan *mobile application.* Sehingga dapat di akses dimanapun dan kapanpun selama terhubung dengan akses internet.

II.10. Android

Aplikasi *Android* ditulis dalam bahasa pemrograman *Java*. Kode *Java* dikompilasi bersama dengan data *fileresource* yang dibutuhkan oleh aplikasi, dimana prosesnya dipaket oleh *tools* yang dinamakan “apt *tools*” ke dalam paket

Android sehingga menghasilkan *file* dengan ekstensi apk. *File* apk itulah yang *kit*a sebut dengan aplikasi, dan nantinya dapat dilakukan instalasi pada perangkat *mobile* [3].

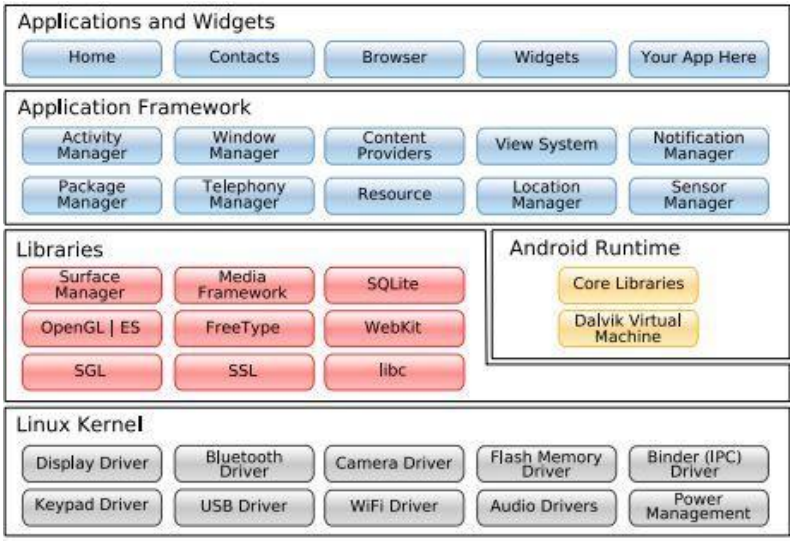
II.10.1. Sejarah Android

Android adalah sebuah sistem operasi untuk perangkat *mobile* berbasis Linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para pengembang untuk menciptakan aplikasi mereka. Awalnya, GoogleInc. Membeli AndroidInc. Yang merupakan pendatang bar yang membuat peranti lnak untuk *Smartphone*.

Pesatnya perkembangan Android adalah platform yang sangat lengkap baik itu sistem operasinya, Aplikasi dan *Tools* pengembangan, *Market* aplikasi Android, serta dukungan yang sangat tinggi dari komunitas *Open Source* di dunia.

II.10.2. Arsitektur Android

Secara garis besar Arsitektur Android dapat dijelaskan dan digambarkan sebagai berikut :



Gambar II.7. Arsitektur Android [3]

Android punya peranan sebagai *operating system* (OS) pada perangkat *mobile* yang akan dibangun aplikasi penilitian ini.

II.11. Metode Black-box

Black-box testing merupakan pendekatan pengujian dimana program dianggap sebagai suatu '*black-box*' ('kotak hitam') Program *test case* berbasiskan spesifikasi *test planning* dapat dimulai sejak awal proses pengembangan sistem.

Black-box testing berfokus pada persyaratan fungsional perangkat lunak. Metode ini memungkinkan *software developer* untuk mendapatkan serangkaian kondisi input yang mempergunakan semua persyaratan fungsional program [11].
Pengujian *blackbox* berusaha menemukan kesalahan dalam kategori :

1. Fungsi – fungsi yang tidak benar atau hilang
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses *database* eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

Equivalence partitioning yaitu metode pengujian *black-box* yang membagi domain input dari suatu program ke dalam kelas data dari mana *test case* dapat dilaksanakan. *Input* data dan *output* hasil terdapat di kelas yang berbeda yang sesuai dengan kelas *input* nya. Masing-masing kelas equivalensi *partition* diproses dimana program akan memproses anggota kelas-kelas tersebut secara equivalen. *Test cases* dipilih dari masing-masing partisi.

Partisi Ekivalensi (equivalensi *partition*) adalah *Input* data dan *output* hasil terdapat di kelas yang berbeda yang sesuai dengan kelas *input* nya. Masing-masing kelas equivalensi *partition* dimana program akan memproses anggota kelas-kelas tersebut secara equivalen.

Integration testing adalah pengujian keseluruhan sistem atau subsistem yang terdiri dari komponen yang terintegrasi. Tes integrasi menggunakan black box dengan *test case* ditentukan dari spesifikasi. Kesulitannya adalah menemukan atau melokasikannya.

Metode *Black-Box* digunakan untuk melakukan pengujian pada aplikasi yang akan dibangun untuk menemukan *bugs* dan *error* yang masih terdapat pada aplikasi.

II.12. 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* [12].

II.13. 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 [13].

II.13.1. 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) [14].

II.13.2. Data Flow Diagram

Data Flow Diagrams (DFD) adalah diagram yang deskripsi aliran data dalam perusahaan secara grafik. *Data Flow Diagram* ini digunakan untuk menggambarkan sistem yang sudah ada dan juga merencanakan dan mendesain sistem yang baru. Tidak ada cara yang ideal untuk membangun sebuah *Data Flow Diagram*, perrnasalahan berbeda akan menghasilan *Data Flow Diagram* yang berbeda pula [10]. DFD mempunyai empat elemen dasar yaitu [15]:

a. *Entity/Data Source* (Sumber dan tujuan data)

Sumber data menunjukkan suatu organisasi atau perseorangan yang memasukkan data ke sistem. Sedangkan tujuan data menunjukkan suatu organisasi atau perseorangan yang menerima data yang dihasilkan oleh sistem. Sumber dan tujuan data mempunyai satu simbol yang sama.



Gambar II.8. Entity

b. *Data Flow* (Aliran Data)

Data Flow menunjukkan aliran data diantara proses, data storage, dan data source. Simbol data flow memiliki panah yang menunjukkan ke mana arah data mengalir.



Gambar II.9. *Data Flow*

c. Proses

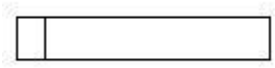
Proses menunjukkan transforrnasi dari data. Pada proses minimal ada satu data yang masuk ke proses dan satu data yang keluar dari proses, dimana data yang masuk tidak boleh sama dengan data yang keluar.



Gambar II.10. *Proses*

d. *Data Storage* (Penyimpanan Data)

Data Storage menunjukkan tempat untuk menyimpan data. Dalam mendokumentasikan sistem. DFD mempunyai level-level, mulai dari yang terkecil yaitu level 0 atau sering disebut *Context* Diagram. *Context* diagram merupakan gambaran paling umum dari sistem, yang hanya mempunyai satu proses saja untuk mewakili proses seluruh sistem. Semakin bertambahnya level dalam DFD akan semakin detail digambarkan proses-proses yang ada pada sistem, tetapi yang boleh bertambah hanya proses dan data flow saja sedangkan untuk data source dan data storage jumlahnya harus tetap dengan yang ada pada *context* diagram.



Gambar II.11. ***Data Storage***

DFD mempunyai peranan untuk menggambarkan aliran data yang terjadi pada web admin.

II.13.3. 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 [16]. Berikut adalah Tabel II.1. Contoh Spesifikasi Proses [14]

Tabel II.1. Contoh Spesifikasi Proses [14]

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)
	Destination	(menatakan tujuan informasi output dari proses)
	Logika Proses	(menyatakan algoritma dari proses)

II.13.4. 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 [14]. Berikut adalah Tabel II.2. Contoh Kamus Data [14].

Tabel II.2. Contoh Kamus Data [14]

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.14. Unified Modeling Language (UML)

UML merupakan seuatu kumpulan pemodelan sistem yang digunakan untuk menentukan atau menggambarkan sebuah sistem perangkat lunak yang memiliki hubungan langsung dengan objek, yang dikenal dengan konsep OOP (*Object Oriented Programming*).

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem OOP karena menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat blueprint dari program yang akan dibuat [17].

UML diagram mempunyai peranan untuk merancang sistem yang berjalan pada aplikasi *mobile.*

II.14.1. Use Case Diagram

Use case diagram menggambarkan fungsionalitas atau relasi yang diharapkan antara sistem internal, ekternal dan *user*. Sebuah *use case* cocok untuk merepresentasikan interaksi yang jelas antara sistem dan pengguna.

a. Use Cases

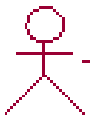
Mendeskripsikan fungsi dari sistem yang dilihat dari sudut pandang pengguna. Disimbolkan dengan bentuk elips.



Gambar II.12. *Use Case* Diagram – *Use Case*

b. Actors

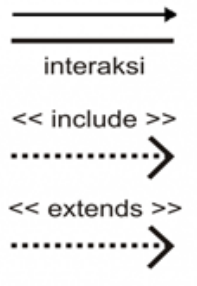
Sesuatu yang berinteraksi dengan sistem untuk saling bertukar informasi. *Actors* tidak harus berupa manusia



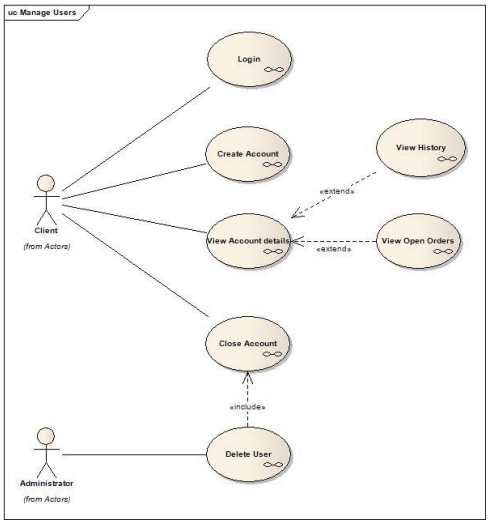
Gambar II.13. *Use case* Diagram – *Actor*

c. Relantionships

Sebuah relasi antara sistem dan user digambarkan dengan sebuah garis diantara keduanya. Arti relasi yang digambarkan bisa beragam tergantung pada interaksi seperti apa yang akan dibangun.



Gambar II.14. Use case Diagram – Relationships



Gambar II.15. Contoh *Use Case* Diagram

Peranan *use case* diagram untuk mengambarkan kelakuan aplikasi mobile yang akan di buat.

II.14.2. Sequence Diagram

Menggambarkan interaksi antara sejumlah objek melalui pesan eksekusi *use case* atau operasi. Diagram ini mengilustrasikan bagaimana sebuah pesan dikirim dan diterima antar objek sesuai urutannya, sehinggan diagram ini menjadi lebih detai menggambarkan aliran data.

Sebuah *sequence* diagram terbentuk dari beberapa notasi, antara lain :

a. Actor

Digambarkan dengan simbol *actor* pada *use case* diagram.

b. System

Digambarkan dengan sebuah kotak.

c. Lifelines

Masa hidup sistem/aktor digambarkan dengan garis putu-putus vertikal.

d. Activation bars

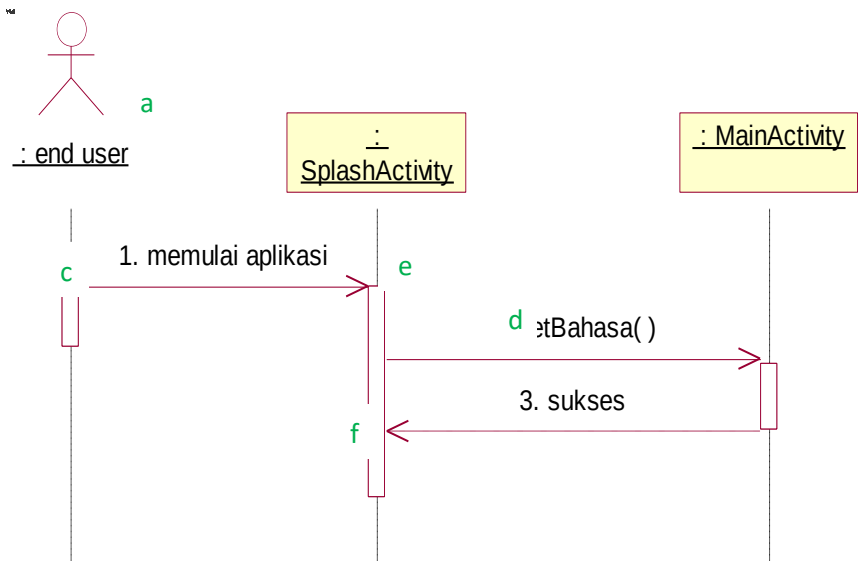
Menggambarkan masa waktu terjadinya interaksi aktif, disimbolkan dengan balok panjang yang diletakkan di atas *lifelines*.

e. Input message

Garis horizontal dengan panah ke kanan yang mengindikasikan pesan masuk.

f. Output message

Garis horizontal dengan panah ke kiri yang mengindikasikan pesan balik.



Gambar II.16. Contoh *Sequence Diagram*

Sequence diagram mempunyai peranan untuk menggambarkan arus pekerjaan, pesan yang di sampaikan dan bagaimana elemen-elemen di dalamnya bekerja sama dari waktu ke waktu untuk mencapai suatu hasil.

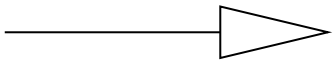
II.14.3. Class Diagram

Class diagram digunakan untuk menggambarkan struktur objek statis dalam sebuah sistem. Diagram ini dapat juga menunjukkan kelas-kelas dan hubngan apa saja yang terbentuk antar kelas tersebut dalam sistem. *Class* diagram terbentuk dari beberapa notasi, antara lain :

a. Attribute dan Behavior

b. Arrow Head

Melambangkan generalisasi atau spesialisasi dari sebuah relasi.



Gambar II.17. *Class* Diagram –*Arrow Head*

c. Solid Diamond

Melambangkan komposisi agregasi dari sebuah relasi.



Gambar II.18. *Class* Diagram – *Solid Diamond*

Class diagram mempunyai peranan menggambarkan kelas-kelas dalam sebuah sistem dan hubungannya antara satu dengan yang lain.

II.14.4. Activity Diagram

Activity diagram menggambarkan alur yang berurutan dari aktivitas *use case* atau proses bisnis. Diagram ini juga bisa digunakan untuk memodelkan berbagai aksi yang dilakukakn saat sebuah operasi dieksekusi, dan hasil dari aksi tersebut. Pada diagram ini akan terlihat ativitas sistem, dari saat mulai dan berakhir. *Activity* diagram dibentuk oleh beberapa notasi, antara lain :

a. Initial Node

Menggambarkan titik mulai suatu proses. Disimbolkan dengan lingkaran penuh.



Gambar II.19. *Activity* Diagram – *Initial Node*

b. Actions

Menggambarkan langkah-langkah yang dilakukan. Disimbolkan dengan notasi segiempat berudut tumpul.



Gambar II.20. *Activity* Diagram – Actions

c. Flow

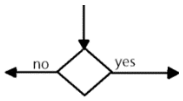
Mengindikasikan alur antar *action*. Disimbolkan dengan panah.



Gambar II.21. *Activity* Diagram – *Flow*

d. Decision

Memiliki bentuk seperti wajik dengan satu alur masuk dan dua atau lebih alur keluar, alur keluar ditentukan dengan kondisi tertentu.



Gambar II.22. *Activity* Diagram - *Decision*

e. Merge

Wajik dengan dua alur atau lebih masuk dan satu alur keluar untuk menggabungkan alur yang sebelumnya terpisah oleh *decision*.

f. Fork

Bar hitam dengan satu alur masuk dan dua atau lebih alur keluar. Aksi di bawah percabangan dapat terjadi dalam urutan apapun atau bahkan secara bersamaan.



Gambar II.23. *Activity* Diagram – *Fork*

g. Join

Bar hitam dengan dua atau lebih alur masuk dan satu alur keluar untuk menyatukan lagi alur aksi yang dipidahkan oleh *fork*.



Gambar II.24. *Activity* Diagram – *Join*

h. Activity final

Menggambarkan titik akhir proses. Berbentuk lingkaran penuh dengan satu lingkaran di luarnya.



Gambar II.25. Activity Diagram – Activity Final

i. Swimlanes

Membagi aktifitas-aktifitas pada *activity* diagram ke dalam beberapa kelompok yang merepresentasikan organisasi (*role* bisnis) yang bertanggung jawab untuk aktifitas tersebut. Setiap aktifitas atau aksi hanya dapat berada pada satu *swimlane*.

Activity diagram berperan untuk menjelaskan aliran kerja aktor pada aplikasi *mobile* yang akan dibangun.

II.15. Entity Relationship Diagram (ERD)

Sebuah ERD mendokumentasikan data sebuah perusahaan dengan cara menentukan data yang terdapat dalam tiap *entity* dan *relationship* (hubungan) antara sebuah *entity* dengan yang lainnya. Di bawah ini akan dijelaskan beberapa istilah yang digunakan dalam ERD [18] :

a. Entity

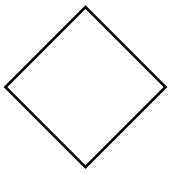
Berupa elemen di sekitar sistem yang berhubungan dengan sistem tersebut (*environmental element*), sumber daya yang berhubungan dengan sistem yang ada (*resource*), dan transaksi yang sangat penting bagi usaha dagang. *Entity* digambarkan dengan kotak persegi dengan sebuah narna *entity* berupa kata benda tunggal di dalamnya.



Gambar II.26. *Entity*

b. Relationship

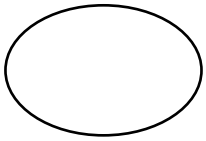
Hubungan yang tetjadi antara dua buah *entity* atau lebih.



Gambar II.27. *Relationship*

c. Attribute

Karakteristik *entity* tersebut yang menjabarkan hal-hal yang berkaitan di dalamnya.



Gambar II.28. *Attribute*

d. Primary Key

Atribut unik yang mengidentifikasikan sebuah *entity* yang bersifat unik dan tidak boleh kembar.

ERD mempunyai peran untuk mendeskripsikan data-data atau objek-objek yang dibuat berdasarkan entitas yang didapatkan pada penilitan.

II.16. PHP

PHP singkatan dari *Hypertext Preprocessor* yaitu bahasa pemrograman *web server-side* yang bersifat *open source*. PHP merupakan script yang terintegrasi dengan HTML dan berada pada server (*Server side HTML embedded scripting*). PHP adalah script yang digunakan untuk membuat halaman website yang dinamis [19].

PHP pertama kali dibuat oleh Rasmus Lerdorf tahun 1995, yang diberi nama FI (*Form Interpreted*) dan digunakan untuk mengelola form dari Web. Pada perkembangannya, kode tersebut dirilis ke umum sehingga mulai banyak dikembangkan oleh *programmer* di seluruh dunia.

Tahun 1997, PHP 2.0 dirilis. Pada versi ini sudah terintegrasi dengan bahasa pemrograman C dan dilengkapi dengan modulnya, sehingga kualitas kerja PHP meningkat secara signifikan.

Pada tahun 2004, Zend merilis PHP 5.0. Dalam versi ini, inti dari interpreter PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi obyek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi obyek [20].

Pada penerapan nya di penelitian ini, PHP berfungsi untuk membangung web admin dan API.

II.17. Flowmap

Disebut juga sebagai *Forms Flowchart* atau Diagram alir dokumen atau *mapping flow* chart atau *Paperwork Flowchart* atau diagram sistem prosedur kerja. *Flowmap* merupakan diagram alir yang menunjukkan arus dari dokumen, aliran data fisik, entitas, entitas sistem informasi dan kegiatan operasi yang digunakan dengan sistem informasi. Terdapat dua jenis entitas [21]:

a. Entitas dalam (internal): Pelaku proses yaitu personal, tempat/bagian, atau mesin seperti komputer dalam suatu sistem yang melakukan kegiatan pemrosesan/pengolahan (transformasi) atau kegiatan pemrosesan informasi.
b. Entitas luar (eksternal): Entitas atau satuan unit yang terletak di lingkungan atau di luar sistem yang mengirim data ke sistem tersebut, atau menerima data dari sistem tersebut .

Flowmap mempunyai peran untuk mendefinisikan hubungan antara bagian (pelaku proses),proses(manual/berbasis komputer) dan aliran data (dalam bentuk dokumen keluaran dan masukan).

II.18. 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 *database* nya 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 [22].

MySQL berperan untuk membangun *database* aplikasi yang dapat di akses oleh web admin dan API.

II.19. 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) [23].

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.19.1. Kelebihan Java

Adapun beberapa kelebihan dari bahasa pemrograman Java adalah sebagai berikut [23] :

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 Bekerja 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.20. Google Maps Direction

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[24].

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=Toledo&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=Toledo&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@vr
Snc}Axf]fjz@xfFbw~@dz{A~d{A|zOxbrBbdUvpo@`cFp~xBc`Hk@nurDznm
FfwMbwz@bbl@lq~@loPpxq@bw_@v|{CbtY~jGqeMb{iF|n\\~mbDzeVh_W
r|Efc\\x`Ij{kE}mAb~uF{cNd}xBjp]fulBiwJpgg@|kHntyArpb@bijCk_Kv~eGy
qTj_|@`uV`k|DcsNdwxAott@r}q@_gc@nu`CnvHx`k@dse@j|p@zpiAp|gEicy
@`omFvaErfo@igQxnlApqGze~AsyRzrjAb__@ftyB}pIlo_BflmA~yQftNboW
zoAlzp@mz`@|}_@fda@jakEitAn{fB_a]lexClshBtmqAdmY_hLxiZd~XtaBnd
gC"
    },
```

```
   "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.21. Android 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 [25]:

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*

Android studio berperan sebagai *editor source code* untuk pembangunan aplikasi *mobile* pada penelitian ini.