BAB 2

TINJAUAN PUSTAKA

2.1. Korps Sukarela

Relawan menurut Markas Besar PMI (Palang Merah Indonesia, 1998) yang juga disebut sebagai Korp Sukarelawan (KSR) dan Tenaga Sukarelawan (TSR) PMI adalah pribadi-pribadi yang secara sukarela meluangkan/ menyumbangkan tenaga, waktu, fikiran, dan keahlian/ ketrampian khusus yang dimilikinya baik yang diperoleh melalui tingkat formal pendidikan maupun secara non formal (kursus, dan sebagainya) dimana hal itu dapat membantu pengembangan Perhimpunan Palang Merah Indonesia. Sedangkan kegiatan operasional relawan dilapangan meliputi :

1. Siap siaga dalam membantu kegiatan dibidang penanggulangan bencana di wilayahnya.
2. Bersedia melakukan kegiatan terpadu dengan masyarakat setempat dalam melaksanakan kegiatan pelayanan social/ kesehatan masyarakat.
3. Berinisiatif mengambil langkah-langkah dalam keadaan darurat jika terjadi sesuatu kejadian luar biasa dimana setelah itu perlu diiformasikan/ dilaporkan kembali.

Dalam menghadapi kejadian darurat dapat melakukan koordinasi dengan instansi lain untuk mendapatkan bantuan [3].

Untuk mewujudkan peranan tersebut Korp Sukarela Palang Merah Indonesia harus mampu menganalisis kelebihan dan kelemahan serta peluang dan hambatan. Dalam upaya mencapai kondisi tersebut perlu adanya peningkatan kualitas dan kapasitas individu maupun instansi yang harus dipenuhi KSR PMI.

Peningkatan kualitas dan kapasitas individu dapat di capai dengan adanya sumber daya manusia yang memiliki rasa kekeluargaan, kekompakan, pola pikir kritis, kreatifitas, disiplin, tanggung jawab, loyalitas dan moralitas serta keimanan yang kuat.

Oleh karena itu, diperlukan adanya proses managerial yang baik serta didukung sarana prasarana yang memadai.

Dengan demikian kinerja KSR PMI ke depan dipengaruhi oleh kemampuan individu dalam mengelola organisasi, atas dasar pemahaman individu terhadap potensi diri serta memahami peluang dan hambatan yang mendukung tercapainya kualitas dan kapasitas individu maupun organisasi [1].

2.2. Pertolongan Pertama

Pertolongan pertama adalah tindakan pertolongan pada kecelakaan atau kejadian yang tidak diinginkan dapat terjadi dimana saja dan kapan saja. Kejadian ini dapat berupa suatu insiden kecil atau suatu bencana yang melibatkan penderita dalam jumlah besar. Bencana yang baru akan terjadi bila para korban tidak mendapat pertolongan yang baik dengan segera. Dalam suatu peristiwa yang membutuhkan penanganan medis biasanya orang pertama yang memberikan pertolongan adalah mereka yang berada ditempat kejadian atau anggota keluarga perderita tersebut. Mereka yang yang berupaya memberikan pertolongan ini memiliki berbagai tingkat pengetahuan mulai dari tidak ada sampai mereka yang mungkin sudah terlatih. Ada waktu antara pertolongan dilapangan sampai korban dapat memperoleh pertolongan oleh tenaga medis di fasilitas kesehatan sehingga masa tenggang inilah yang harus diisi. Prinsip kemanusiaan yang utama adalah mengurangi penderitaan dan memberikan bantuan kepada para penderita. Ini harus dilakukan sebaik-baiknya dan dalam waktu singkat. Pertolongan yang diberikan harus menjadi satu kesatuan pertolongan korban dari lapangan sampai perawatan lanjutan di rumah sakit [1].

Pertolongan ini dikenal dengan pelayanan gawat darurat, pelayanan ini dibagi menjadi dua fase :

2.3.1. Fase Pra Rumah Sakit

Pada fase ini dilakukan perawatan ditempat kejadian dengan atau tanpa melakukan transportasi penderita kefasilitas kesehatan. Konsep dasar dari pertolongan pertama adalah memberikan bantuan hidup dasar dan mempertahankan nyawa dengan melakukan tindakan pertolongan pertama secepatnya setelah kejadian.

2.3.2. Fase Perawatan Rumah Sakit

Para penderita tentunya akan dikirim kefasilitas kesehatan yang umumnya adalah rumah sakit atau puskesmas di daerah-daerah terpencil. Perawatan fase kedua ini seharusnya tidak dibedakan keduanya harus saling menunjang, fase pra rumah sakit dilakukan dengan baik sehingga rumah sakit tinggal melanjutkan apa yang sudah dilakukan dan tidak mundur kembali, jika diperlukan maka sistem rujukan rumah sakit harus diaktifkan [1].

2.3. Aplikasi

Secara istilah pengertian aplikasi adalah suatu program yang siap untuk digunakan yang dibuat untuk melaksanakan suatu fungsi bagi pengguna jasa aplikasi serta penggunaan aplikasi lain yang dapat digunakan oleh suatu sasaran yang akan dituju. Menurut kamus komputer eksekutif, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu teknik pemrosesan data aplikasi yang biasanya berpacu pada sebuah komputasi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan.

Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia “Aplikasi adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu” [4].

Sedangkang menurut para ahli, pengertian aplikasi adalah sebagai berikut :

1. Pengertian aplikasi menurut Jogiyanto(1999:12) adalah penggunaan dalam suatu computer, instruksi (*instruction*) atau pernyataan (*statement*) yang disusun sedemikian rupa sehingga computer dapat memproses input menjadi output.
2. Pengertian aplikasi menurut Kamus Besar Bahasa Indonesia (1998:52) adalah penerapan dari rancang sistem untuk mengolah data yang menggunakan aturan atau ketentuan bahasa pemrograman tertentu. Aplikasi adalah suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna.

3. Menurut Wikipedia aplikasi adalah suatu subkelas perangkat lunak computer yang memanfaatkan kemapuan computer langsung untuk melakukan suatu tugas yang dinginkan pengguna.
4. Menurut rachmad Hakim S aplikasi adalah perangkat lunak yang digunakan untuk tujuan tertentu, seperti mengolah dokumen, mengatur windows dan permaian (*game*) dan sebagainya.
5. Menurut Harip Santoso Aplikasi adalah suatu kelompok file (*form, class, repor*t) yang bertujuan untuk melakukan aktivitas tertentu yang saling terkait, misalnya aplikasi *payroll, aplikasi fixed asset*, dan lain-lain [5].

2.4.1. Aplikasi Berdasarkan Fungsi

Aplikasi dapat dibagi menjadi beberapa berdasarkan fungsinya, yaitu :

1. *Word Processing*

   Word Processing adalah aplikasi yang berfungsi untuk pengolahan kata seperti menulis dokumen atau surat. Contoh : *Microsoft Word, Google Docs*.
2. *Desktop Publishing*

   Desktop Publishing adalah aplikasi yang berfungsi untuk membatu penulisan naskah. Contoh: *Final Draft*, *Trelby*.
3. *Program Speadsheet*

   Program Speadsheet adalah aplikasi yang berfungsi untuk perhitungan matematika dengan *layout* baris dan kolom. Contoh: *Microsoft Excel, Google Sheets.*
4. *Database Management Sistem*

   Database Management Sistem adalah aplikasi yang berfungsi untuk pengolah basis data. Pengolahan ini dapat berupa penambahan, pengubahan, pembacaan dan penghapusan data. Contoh: *Microsoft Access, phpMyAdmin*.
5. *Graphics*

   Graphics adalah aplikasi yang berfungsi untuk pengolahan citra. Contoh : *Adobe Photoshop, Paint*.

6. *Program* Akuntansi

   Aplikasi yang berfungsi untuk mengolah data akutansi dan keuangan. Contoh : *ZipBooks, Zoho Books.*

7. *Program* Statistik

   Aplikasi yang berfungsi untuk mengolah data yang mampu menghasilkan informasi yang berguna. Contoh : SPSS, Sttata, *GraphPad*.

8. *Communication*

   Communication adalah aplikasi yang berfungsi membantu dalam hubungan komunikasi jarak jauh antar dua atau lebih orang dalam waktu yang bersamaan atau berbeda. Contoh: *Skype*, *Whatsapp*.

9. *Multi media*

   Multi media adalah aplikasi yang berfungsi untuk pekerjaan multimedia seperti suara, video, teks dan gambar. Contoh : *Adobe Premiere, Blender*.

10. *Game*

    Game adalah aplikasi yang berfungsi untuk melakukan permainan yang menghibur dengan media video. Contoh : *Dota 2, World of Warcraft*.

11. *Anti Virus*

    Anti Virus adalah aplikasi yang berfungsi untuk mendeteksi dan membersihkan virus pada komputer. Contoh : *Smadav, Windows Defender*.

2.4. *Android*

Andorid merupakan sistem operasi yang memang khusus dirancang untuk *smartphone* dan *tablet*. Sistem *android* memiliki basis *linux* yang mana dijadikan sebagai pondasi dasar dari sistem operasi android. *Linux* sendiri merupakan sistem operasi yang memang khusus dirancang untuk komputer. *Android* memang dirancang untuk dipasang pada perangkat - perangkat *mobile touchscreen* (*smartphone*, dan *tablet*). Sehingga sistem operasi yang berada di dalam *smartphone* saat ini memang menyesuaikan dari spesifikasi kelas *low-end* hingga *high-end.* Sehingga perkembangan sistem android memang cukup memang meningkat tajam. *Android* merupakan sistem operasi yang terbuka *(open source)* yang mana jika pihak *Google* memperbolehkan dan

membebaskan bagi pihak manapun untuk dapat mengembangkan sistem operasi tersebut.

Bahkan anda sendiri pun juga dapat mengembangkan sistem *android* yang memang sesuai dengan keinginan anda. Sistem *android* memiliki gudang aplikasi dan *game* yaitu *Google Play Store*, yang mana disini anda bisa men*download* serta menggunakan aplikasi atau *game* yang terdapat di *Google Play Store* sepuasnya dengan menggunakan perangkat seluler dengan sistem *android*. Dengan uniknya, *android* menggunakan nama-nama makanan untuk membedakan versi sistem *android* yang diluncurkannya. *Android* dengan menggunakan huruf depan dari nama makanan tersebut sebagai penanda peningkatan versi sistemnya. Mulai dari *Cupcake Andorid* 1.5 (C) pada 27 april 2009, *Donuts* Android 1.6 (D) pada 15 september 2009, *Éclair Android* 2.0-2.1 (E) pada 26 oktober 2009 atau *Marshmallow Andorid* (M) pada 5 oktober 2015 hinga yang terbaru sekarang versi *Pie Android* 9.0 (P) pada 6 agustus 2018 [6]. Berikut daftar versi android dapat dilihat pada Table 2.1 :

Tabel 2. 1 Daftar Versi Android

Versi	Nama Versi	Tanggal Rilis
1.0	(*No Codename*)	23 September 2008
1.1	(*Internally known as "Petit Four*")	9 February 2009
1.5	*Cupcake*	27 April 2009
1.6	*Donut*	15 September 2009
2.0	*Éclair*	26 Oktober 2009
2.2	*Froyo*	20 Mei 2010
2.3	*Gingerbread*	6 Desember 2010
3.0	*Honeycomb*	22 Februari 2011
4.0	*Ice Cream Sandwich*	18 Oktober 2011
4.1	*Jelly Bean*	9 Juli 2012
4.4	*Kitkat*	31 Oktober 2013
5.0	*Lollipop*	12 November 2014
6.0	*Marshmellow*	5 Oktober 2015
7.0	*Nougat*	22 Agustus 2016
8.0	*Oreo*	21 Agustus 2017
9.0	*Pie*	6 Agustus 2018

2.5.1. Arsitektur *Android*

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

2.5.1.1. *Aplication dan Widgets*

Aplication dan *Widgets* ini adalah layer dimana kita berhubungan dengan aplikasi saja, dimana biasanya kita *download* aplikasi kemudian kita lakukan instalasi dan jalankan aplikasi tersebut. Dilayer terdapat aplikasi inti termasuk klien *email*, *program* SMS, kalender, peta, *browser*, kontak dan lain-lain. Semua aplikasi ditulis menggunakan pemrograman *java*.

2.5.1.2. *Application Frameworks*

Android adalah "*Open Development Platform*" yaitu *android* menawarkan kepada pengembang atau memberi kemampuan kepada pengembang untuk membangun aplikasi yang bagus dan inovatif. Pengembang bebas untuk mengakses perangkat keras, akses informasi *resources*, menjalankan *service background*, dan sebagainya. Pengembang memiliki akses penuh menuju API *Framework* seperti yang dilakukan oleh aplikasi yang kategori inti. Arsitektur aplikasi dirancang supaya kita dengan mudah dapat menggunakan kembali komponen yang sudah digunakan (*reuse*).

Komponen – komponen yang termasuk didalam *applications Frameworks* adalah sebagai berikut ini :

1. *Viess*
2. *Content Provider*
3. *Resource Manager*
4. *Notification Manager*
5. *Activity Manager*
6. *Libraries*

2.5.1.3. *Libraries*

Libraries ini adalah layer dimana fitur – fitur *android* berada, biasanya para pembuat aplikasi mengakses *Libraries* untuk menjalankan aplikasinya. Berjalan diatas kernel, layer ini meliputi berbagai *library* C/C++ inti seperti Libc dan SSL, Serta :

1. *Libraries* media untuk pemutaran media *audio* dan *video*
2. *Libraries* untuk menajemen tampilan
3. *Libraries graphics* mencakup SGL dan *Open*GL untuk grafis 2D dan 3D
4. *Libraries SQLite* untuk dukungan *database*
5. *Libraries* SSL dan *WebKit* terintegrasi dengan *webserver* dan *security*
6. *Libraries LiveWebcore* mencakup modern *web browser* dengan *engine embedded web view*
7. *Libraries 3D* yang mencakup inplementasi *OpenGL ES 1.0 API's*

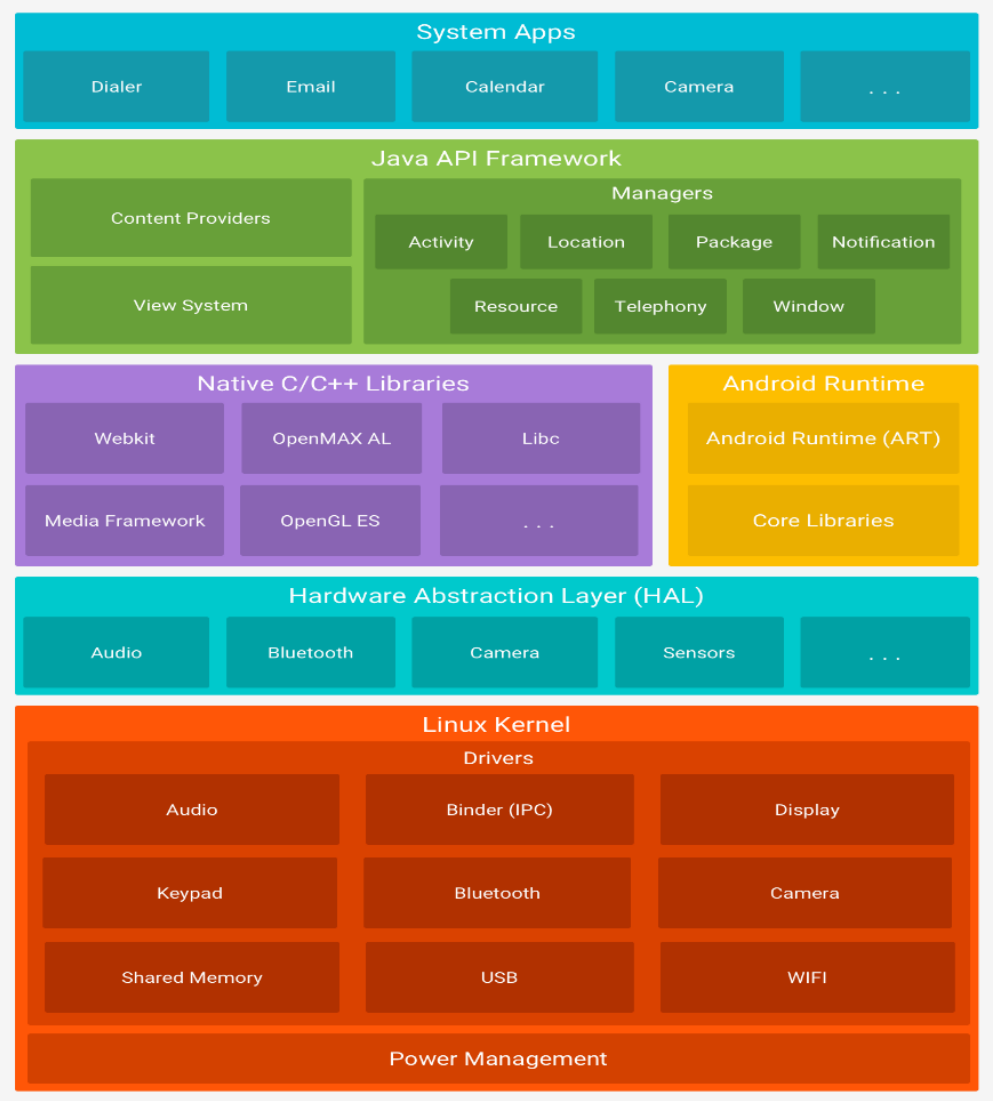
2.5.1.4. *Android Run Time*

Layer yang mebuat aplikasi android dapat dijalankan dimana dalam prosesnya menggunakan Implementasi Linux. *Dalvik Virtual Machine* (DVM) merupakan mesin yang membentuk dasar kerangka aplikasi android. Di dalam *Android Run Time* dibagi menjadi dua bagian yaitu :

1. *Core Libraries* : Aplikasi Android dibangun dalam bahasa java, sementara Dalvik sebagai *virtual* mesinnya bukan *virtual machine* java, sehingga diperlukan sebuah *Libraries* yang berfungsi untuk menterjemahkan bahasa java/c yang ditangani oleh *core Libraries.*
2. *Dalvik Virtual Machine* : *Virtual* mesin berbasis *register* yang dioptimalkan untuk menjalankan fungsi – fungsi secara efisien, dimana merupakan pengembangan yang mampu membuat linux kernel untuk melakukan *threading* dan manajemen tingkat rendah.

2.5.1.5. *Linux Kernel*

Linux Kernel adalah layer dimana inti dari *operating sistem* dari android itu berada. Berisi file – file sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem – sistem android lainnya. Linux kernel yang digunakan android adalah linux kernel *release* 2.6 [6].



Sumber : https://developer.android.com/guide/platform/

Gambar 2. 1 Tumpukan perangkat lunak

2.5. *Android Studio*

Android studio adalah sebuah IDE untuk *android Development* yang diperkenalkan *google* pada acara Google I/O 2013. *Android studio* merupakan pengembangan dari *Eclipse* IDE *Java* popular, yaitu *Intellij* IDEA. *Android Studio* merupakan IDE resmi untuk pengembangan aplikasi *Android* [7].

Berdasarkan website *https://developer.android.com* , android studio memiliki beberapa keunggulan :

1. *Android studio* didasarkan pada intellij DEA yang memberikan kemungkinan tercepat dalam melakukan coding dan *running workflow* sehingga meng-*coding* dan melakukan iterasi dapat dilakukan dengan cepat.
2. Struktur proyek *Android Studio* dan pembangunannya yang berbasis *gradle* memberikan fleksibilitas yang dibutuhkan untuk menghasilkan APK untuk semua jenis *device* sehingga konfigurasi pemangunan dapat dilakukan tanpa batas.
3. *Android Studio* menyediakan fitur-fitur yang membuat pengembang menjadi percaya diri dan dapat membuat kode terbaik.
4. *Android Studio* mengetahui bahwa tidak seluruh kode ditulis dengan bahasa *java* dan tidak semua kode dapat berjalan di *device* pengguna. Dengan fitur-fitur yang dimiliki Android Studio maka aplikasi menjadi kaya dan terkoneksi satu sama lain.
5. *Android Studio* menyediakan alat GUI untuk mempermudah pengembang dalam merancang tampilan aplikasi sehingga pengembang tidak lelah [8].

Dengan menggunakan android studio pengembang aplikasi dapat mengerjakan aplikasi yang akan dibangunya. *Interface* yang digunakan lebih elegan dan *"modern"*, lalu kelebihan-kelebihan lain seperti yang tertera diatas dapat membantu pengguna yang akan membangun aplikasi. Berbagai menu dan perintah yang ada memberikan kesan yang sesuai dengan tujuan dari *Android Studio* itu sendiri, yaitu membuat aplikasi *Android* secara nyaman, mudah, dan inovatif.

2.6. *Java*

Java menurut definisi *Sun* merupakan sekumpulan teknologi untuk membuat dan menjalankan perangkat lunak pada komputer *stand alone* ataupun lingkungan jaringan. Bahasa pemrograman *java* ini bersifat *case sensitive*, sehingga harus lebih memperhatikan bentuk tertentu, sehingga dalam menuliskan semua baris *code* tersebut dalam satu baris tersendiri [9].

Java adalah bahasa yang dapat dijalankan di sembarang *platform*, di beragam lingkungan : internet, consumen electronic products, dan computer applications. *The Java 2 Platform* tersedia dalam tiga edisi untuk keperluan berbeda berikut :

2.7.1. *Java 2 Standart Edition* (J2SE)

The Java 2 Platform, standart edition (J2SE) menyediakan lingkungan pengembangan yang kaya fitur, stabil, aman, dan *cross-platform*. Edisi ini mendukung konektivitas basis data, rancangan antarmuka pemakai, masukan/keluaran, dan pemrograman jaringan dan termasuk sebagai paket dasar bahasa *java*.

2.7.2. *Java 2 Enterprise Edition* (J2EE)

Enterprise Edtion (J2EE) menyediakan kakas untuk membangun dan menjalankan *multitier enterprise applications*. J2EE berisi paket-paket di J2SE ditambah paket-paket untuk mendukung pegembangan *Enterprise JavaBeans, Java Servlets, JavaServer Pages, XML* dan kendali transaksi yang fleksibel.

2.7.3. *Java 2 Micro Edition* (J2ME)

Micro Edition (J2ME) untuk beragam *consumer electronic product*, seperti *pager, smart card, cell phone, handled* PDA dan *set-op box*. J2ME sembari menyediakan bahasa java yang sama, unggul dalam portabilitas yaitu kemampuan dijalankan dimana pun dan *safe network delivery* seperti J2SE. J2ME menggunakan sekumpulan paket lebih kecil. J2ME berisi subset paket J2SE ditambah paket spesifik *Micro Edition javax.microedtion.io*. Aplikasi J2ME dapat diskala agar juga dapat bekerja dengan J2SE dan J2EE [10].

2.7. *Web Services*

Web service adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas dan interaksi antar sistem pada suatu jaringan. *Web service* digunakan sebagai suatu fasilitas yang disediakan oleh suatu *website* untuk menyediakan layanan (dalam bentuk informasi) kepada sistem lain, sehingga sistem lain dapat berinteraksi dengan sistem tersebut melalui layanan (*service*) yang disediakan oleh suatu sistem yang menyediakan *web service*. *Web service* menyimpan data informasi dalam format XML, sehingga data ini dapat diakses oleh sistem lain walaupun berbeda platform, sistem operasi, maupun bahasa compiler.

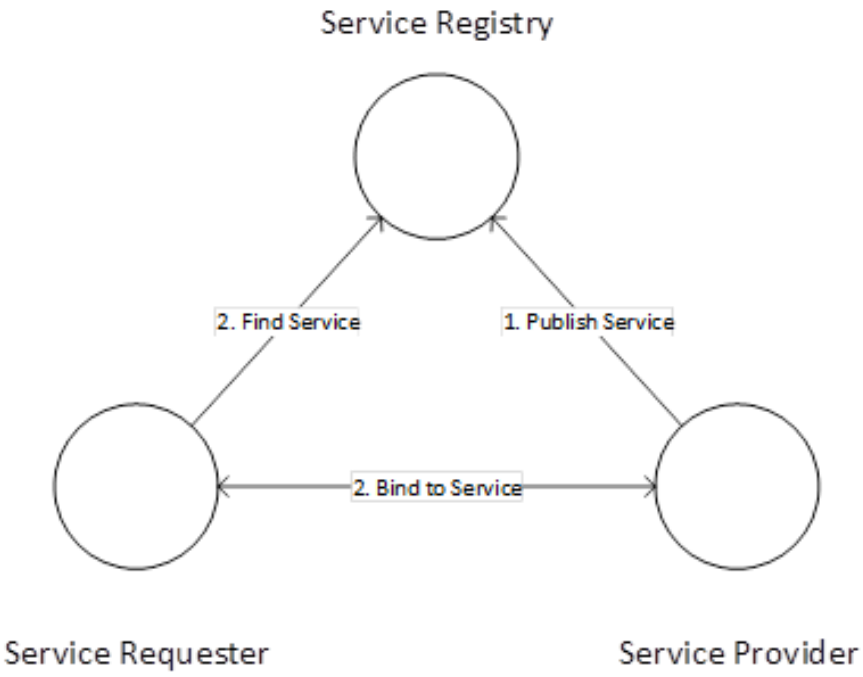
Web service bertujuan untuk meningkatkan kolaborasi antar pemrogram dan perusahaan, yang memungkinkan sebuah fungsi di dalam *Web Service* dapat dipinjam oleh aplikasi lain tanpa perlu mengetahui detil pemrograman yang terdapat di dalamnya. Terdapat 3 alasan mengapa digunakannya *web service* adalah sebagai berikut :

1. *Web service* dapat digunakan untuk mentransformasikan satu atau beberapa bisnis *logic* atau *class* dan objek yang terpisah dalam satu ruang lingkup yang menjadi satu, sehingga tingkat keamanan dapat ditangani dengan baik.
2. *Web service* memiliki kemudahan dalam proses deployment-nya, karena tidak memerlukan registrasi khusus ke dalam suatu sistem operasi. *Web service* cukup di-*upload* ke *web server* dan siap diakses oleh pihak-pihak yang telah diberikan otorisasi.
3. *Web service* berjalan di port 80 yang merupakan protokol standar HTTP, dengan demikian web service tidak memerlukan konfigurasi khusus di sisi *firewall*.

2.8.1. Arsitektur *Web Services*

Web service memiliki tiga entitas dalam arsitekturnya, yaitu:

1. *Service Requester* (peminta layanan)
2. *Service Provider* (penyedia layanan)
3. *Service Registry* (daftar layanan)



Sumber: Teknik Pemrograman Web Service PHP

Gambar 2. 2 Arsitektur *Web Services*

1. *Service Provider* berfungsi untuk menyediakan layanan/*service* dan mengolah sebuah *registry* agar layanan-layanan tersebut dapat tersedia.
2. *Service Registry* berfungsi sebagai lokasi central yang mendeskripsikan semua layanan/*service* yang telah di-*register*.
3. *Service Requestor* meminta layanan yang mencari dan menemukan layanan yang dibutuhkan serta menggunakan layanan tersebut. [11]

2.8. PHP (*Hypertext Preprocessor*)

PHP dikenal secara umum dikenal sebagai bahasa pemrograman *script-script* yang membuat dokumen HTML secara *on the fly* yang dieksekusi di *server web*, dokumen yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan *editor* teks atau *editor* HTML, dikenal juga sebagai bahasa pemrograman *server side* [12]. PHP diciptakan oleh Rasmus Lerdorf pertama kali tahun 1994. Pada awalnya PHP adalah singkatan dari "*Personal Home Page Tools*".

Selanjutnya diganti menjadi FI ("*Forms Interpreter*"). Sejak versi 3.0, nama bahasa ini diubah menjadi "PHP: *Hypertext Prepocessor*" dengan singkatannya "PHP".

Beberapa kelebihan PHP sebagai bahasa pemrograman web antara lain sebagai berikut.

1. Bahasa pemrograman PHP adalah sebuah bahasa *script* yang tidak melakukan sebuah kompilasi dalam penggunaannya.
2. *Web Server* yang mendukung PHP dapat ditemukan di mana - mana dari mulai *apache*, IIS, *Lighttpd*, hingga *Xitami* dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah, karena banyaknya milis - milis dan *developer* yang siap membantu dalam pengembangan.
4. Dalam sisi pemahamanan, PHP adalah bahasa scripting yang paling mudah karena memiliki referensi yang banyak.
5. PHP adalah bahasa open source yang dapat digunakan di berbagai mesin (*Linux, Unix, Macintosh, Windows*) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat menjalankan perintah-perintah sistem.

2.9. *MySQL*

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL atau yang dikenal dengan DBMS (*Database Management System*), database ini *multithread, multi-user*. *MySQL* AB membuat *MySQL* tersedia sebagai perangkat lunak gratis dibawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual di bawah lisensi komersial untuk kasus-kasus yang bersifat khusus.

Kekuatan *MySQL* tidak ditopang oleh sebuah komunitas, seperti *Apache*, yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh pemilik masing-masing, tetapi *MySQL* didukung penuh oleh sebuah perusahaan profesional dan komersial, yakni *MySQL* AB dari Swedia.

MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis di bawah lisensi GPL (*General Public License*). Di mana setiap orang bebas untuk menggunakan *MySQL*, namun tidak boleh dijadikan produk

turunan yang bersifat *closed source* atau komersial. *MySQL* sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan mudah secara otomatis [13].

Dalam penggunaan database *MySQL*, setiap peritah yang diketikkan disebut *query*. Perintah MySQL terdapat 3 sub perintah, yaitu *DDL (Data Definition Language), DML (Data Manipulation Language)* dan *DCL (Data Control Language)*.

2.10.1. DDL (*Data Definition Language*)

Perintah dalam SQL yang pertama adalah perintah DDL. DDL sendiri mrupakan kependekan dari apa yang dikenal dengan nama *Data Definition Language*. DDL dapat berarti sebuah perintah yang berhubungan dengan pendefinisian dari suatu struktur database. Terdapat beberapa perintah DDL pada *MySQL* sebagai berikut :

1. *CREATE* berfungsi untuk membuat database baru, tabel baru, *view* baru dan kolom.
2. *ALTER* berfungsi untuk mengubah struktur tabel. Seperti mengganti nama tabel, menambah kolom, mengubah kolom, menghapus kolom maupun memberikan atribut pada kolom.
3. *DROP* berfungsi untuk menghapus database dan tabel.
4. *TRUNCATE* berfungsi untuk Menghapus semua catatan dari tabel.
5. *COMMENT* berfungsi untuk Menambahkan komentar pada data.
6. *RENAME* berfungsi untuk mengubah nama obyek.

2.10.2. DML (*Data Manipulation Language*)

Data Manipulation Language (DML) adalah perintah yang digunakan untuk mengelolah data dalam database. Terdapat beberapa perintah DML pada *MySQL* sebagi berikut :

1. *SELECT* untuk menganampilkan data dari *database*.
2. *INSERT* untuk menambah data pada *database*.
3. *UPDATE* untuk merubah data dalam tabel.

4. *DELETEI* untuk menghapus data dari tabel.
5. *CALL* untuk memanggil subprogram SQL atau *Java*
6. *EXPLAIN PLAN* untuk menjelaskan jalur akses ke data.
7. *LOCK* TABEL untuk mengunci tabel.

2.10.3. DCL (*Data Control Language*)

Data Control Language (DCL) ialah perintah yang digunakan untuk melakukan pengontrolan data dan server databasenya. Terdapat beberapa perintah DCL pada MySQL sebagai berikut :

1. *GRANT* berfungsi untuk memberikan hak akses pengguna ke database.
2. *REVOKE* berfungsi untuk menghilangkan hak akses yang telah diberikan dengan perintah *GRANT* [15].

Berikut adalah contoh sintak dasar yang perlu diketahui untuk mengelola basis data MySQL:

1. **Membuat Database**

   Contoh sintaks dasar sql dalam membuat database :

```
MariaDB [(none)]> create database mahasiswa;
Query OK, 1 row affected (0.00 sec)
```

Gambar 2. 3 Sintaks SQL Membuat Database

2. **Menggunakan Database**

   Contoh sintaks dasar sql untuk menggunakan database yang telah dibuat :

```
MariaDB [(none)]> use mahasiswa;
Database changed
```

Gambar 2. 4 Sintaks SQL Menggunakan Database

3. **Membuat Tabel**

Contoh sintaks dasar sql dalam membuat tabel kedalam database :

```
MariaDB [mahasiswa]> create table data_mahasiswa(
    -> ID varchar(8) not null,
    -> Nim varchar(8) not null,
    -> Nama varchar(25) not null,
    -> Alamat varchar(50) not null,
    -> Tanggal_lahir varchar(20) not null,
    -> Tahun_masuk varchar(8) not null,
    -> Kelas varchar(5) not null,
    -> primary key (ID)
    -> );
Query OK, 0 rows affected (0.07 sec)
```

Gambar 2. 5 Sintaks SQL Membuat Tabel

4. **Menambah Data**

Contoh sintaks dasar sql untuk menambah data kedalam tabel yang berada di database :

```
MariaDB [mahasiswa]> insert into data_mahasiswa (
    -> ID, Nim, Nama, Alamat, Tanggal_lahir, Tahun_masuk, Kelas )
    -> values (
    -> "001","10113067","Mazerianto Simanullang","Jl. Sekeloa Tengah 74 Bandung",
    -> "16 Maret 1993","2013","IF-3");
Query OK, 1 row affected (0.02 sec)
```

Gambar 2. 6 Sintaks SQL Menambah Data

5. **Mengambil Data**

Contoh sintaks dasar sql untuk mengambil data dari database :

```
MariaDB [mahasiswa]> SELECT * FROM data_mahasiswa;
+-----+----------+----------------------+-----------------------------+--------------+-------------+-------+
| ID  | Nim      | Nama                 | Alamat                      | Tanggal_lahir | Tahun_masuk | Kelas |
+-----+----------+----------------------+-----------------------------+--------------+-------------+-------+
| 001 | 10113067 | Mazerianto Simanullang | Jl. Sekeloa Tengah 74 Bandung | 16 Maret 1993 | 2013        | IF-3  |
+-----+----------+----------------------+-----------------------------+--------------+-------------+-------+
1 row in set (0.00 sec)
```

Gambar 2. 7 Sintaks SQL Mengambil Data

6. **Mengubah Data**

Contoh sintaks dasar sql untuk meng*update* atau mengubah data yang ada dalam database :

```
MariaDB [mahasiswa]> update data_mahasiswa
    -> SET Alamat ='Jl. Taman Kalijaga Permai Kota Cirebon'
    -> WHERE ID='001';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mahasiswa]> SELECT * FROM data_mahasiswa;
+-----+----------+-----------------------+-----------------------------------+--------------+-------------+-------+
| ID  | Nim      | Nama                  | Alamat                            | Tanggal_lahir | Tahun_masuk | Kelas |
+-----+----------+-----------------------+-----------------------------------+--------------+-------------+-------+
| 001 | 10113067 | Mazerianto Simanullang | Jl. Taman Kalijaga Permai Kota Cirebon | 16 Maret 1993 | 2013        | IF-3  |
+-----+----------+-----------------------+-----------------------------------+--------------+-------------+-------+
1 row in set (0.00 sec)
```

Gambar 2. 8 Sintaks SQL Mengubah Data

7. **Menghapus Data**

Contoh sintaks dasar sql untuk menghapus data yang ada didalam database :

```
MariaDB [mahasiswa]> delete from data_mahasiswa where Nim='10113067';
Query OK, 1 row affected (0.01 sec)

MariaDB [mahasiswa]> select * from data_mahasiswa;
Empty set (0.00 sec)
```

Gambar 2. 9 Sintaks SQL Menghapus Data

8. **Menghapus Tabel**

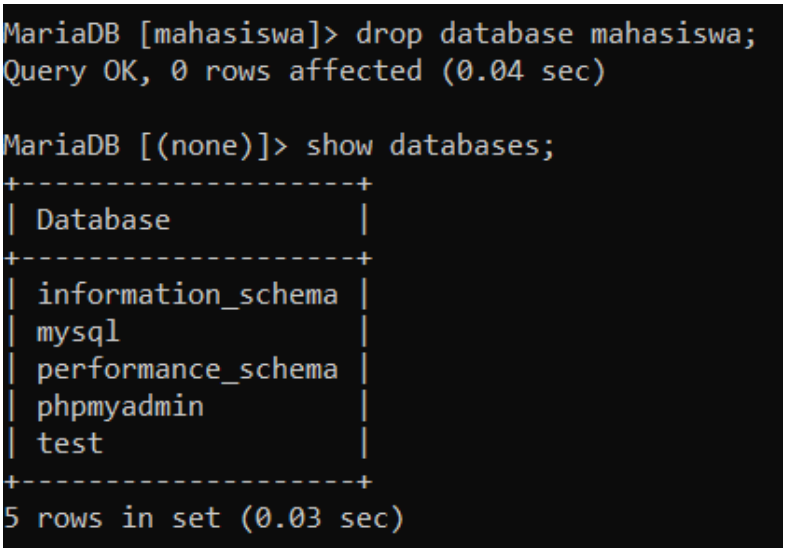
Contoh sintaks dasar sql menghapus tabel yang telah tidak digunakan :

```
MariaDB [mahasiswa]> drop table data_mahasiswa;
Query OK, 0 rows affected (0.06 sec)
```

Gambar 2. 10 Sintaks SQL Menghapus Tabel

9. **Menghapus Database**

Contoh sintaks dasar sql menghapus database yang otomatis akan menghapus seluruh data yang ada didalamnya :



Gambar 2. 11 Sintaks SQL Menampilkan Database

2.10. *Global Positioning System* (GPS)

Global Positioning System (GPS) adalah system *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. GPS terdiri dari 3 segmen yaitu segmen angkasa, *control*/ pengendali, dan pengguna. Segmen angkasa terdiri dari 24 satelit yang beroperasi dalam 6 orbit pada ketinggian 20.200 km dengan periode 12 jam (satelit akan kembali ke titik yang sama dalam 12 jam). *Segmen control*/ pengendali terdapat pusat pengendali utama yang terdapat di *Colorado Springs* dan 5 stasiun pemantau lainnya, 3 antena yang tersebar dibumi ini. Pada sisi pengguna dibutuhkan penerima GPS yang biasanya terdiri dari penerima, *rosesor* dan antena [15].

2.11. *Application Programming Interface* (API)

Application Programming Interface (API) adalah sekumpulan perintah, fungsi, dan protokol yang dapat digunakan saat membangun perangkat lunak untuk sistem

operasi tertentu. API memungkinkan *programmer* untuk menggunakan fungsi standar untuk berinteraksi dengan sistem operasi. API atau *Application Programming Interface* juga merupakan suatu dokumentasi yang terdiri dari antar muka, fungsi, kelas, struktur untuk membangun sebuah perangkat lunak.

Dengan adanya API, maka memudahkan seorang programmer untuk membongkar suatu *software* untuk kemudian dapat dikembangkan atau diintegrasikan dengan perangkat lunak yang lain. API dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya. Suatu rutin standar yang memungkinkan *developer* menggunakan *system function*. Proses ini dikelola melalui sistem operasi. Keunggulan dari API ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya untuk saling berinteraksi [16].

2.12. JSON (*JavaScript Object Notation*)

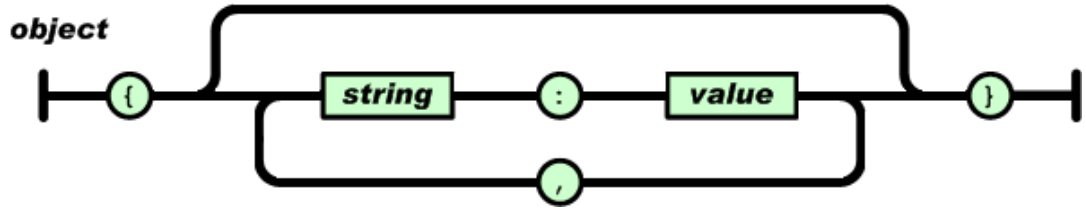
JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan. Sangat mudah dibaca dan menulisnya, serta mudah bagi komputer untuk mem-parsing dan menghasilkan. *JSON* format teks yang bebas Bahasa tetapi menggunakan konversi seperti C, C++, C#, *Java*, *Java*Script, *Perl*, *Python*, dan banyak lainnya. *JSON* kebanyakan juga sebagai Bahasa pertukaran data. *JSON* di bangun dengan dua struktur :

1. Kumpulan nama atau nilai. Dalam berbagai bahasa, ini derealisasikan sebagai *object*, *record*, *struct*, *dictionary*. *hash* tabel, *keyed list* atau *associative array*.
2. Daftar nilai yang terurut. Sebagian besar bahasa, ini direalisasikan sebagai *array*, *vector*, *list* atau *sequence*.

Struktur tersebut merupakan struktur data universal. Hampir semua Bahasa pemrograman yang modern sudah mendukungnya dalam satu bentuk atau lainnya. Masuk akal bahwa format data yang dapat dipertukarkan dengan bahasa pemrograman juga didasarkan pada struktur ini. *JSON* memiliki format – format tipe data seperti berikut [17] :

1. ***Object***

Berikut adalah format *JSON object* yang dapat dilihat pada Gambar 2.12:



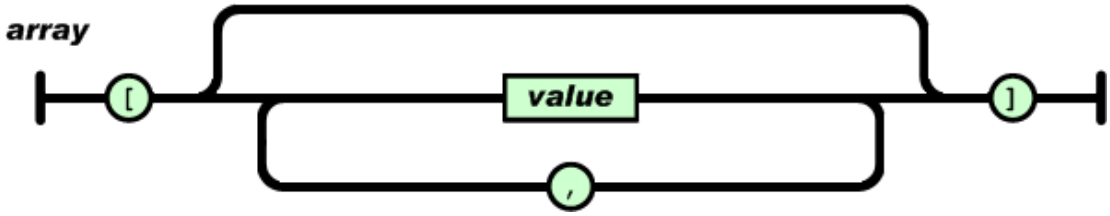
Sumber Gambar *: http://json.org/*

Gambar 2. 12 Format JSON *object*

Object adalah kumpulan dari pasangan nama atau nilai tak berurutan. *Object* dimulai dengan "**{**" (kurung kurawal kiri) dan diakhiri dengan "**}**" (kurung kurawal kanan). Setiap nama diikuti oleh "**:**" (titik dua) dan pasangan nama atau nilai dipasangkan oleh "**,**" koma.

2. ***Array***

Berikut adalah format *JSON array* yang dapat dilihat pada Gambar 2.13 :



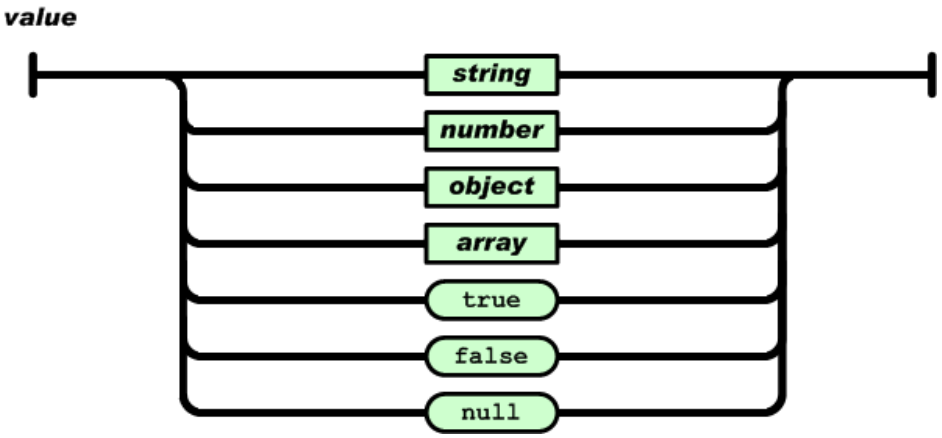
Sumber Gamber : *http://json.org/*

Gambar 2. 13 Format JSON *array*

Array adalah kumpulan nilai yang diurutkan. *Array* dimulai dengan "**[**" (kurung siku kiri) dan diakhiri dengan "**]**" (kurung siku kanan). Nilai dipisahkan oleh "**,**" (koma).

3. *Value*

Berikut adalah format *JSON value* yang dapat dilihat pada Gambar 2.14 :



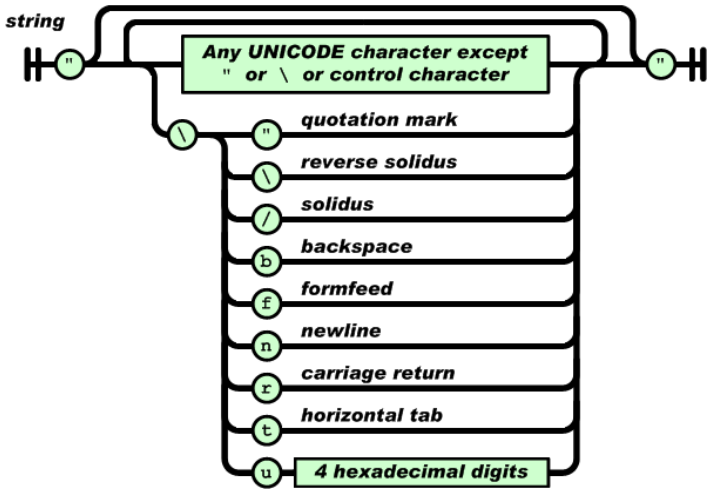
Sumber Gamber : *http://json.org/*

Gambar 2. 14 Format JSON *Value*

Value (nilai) dapat berupa *string* dalam tanda kutip ganda, *number*, *true* atau *false* atau *null, object* atau *array*. Struktur-struktur ini dapat diulang.

4. *String*

Berikut adalah format *JSON string* yang dapat dilihat pada Gambar 2.15 :



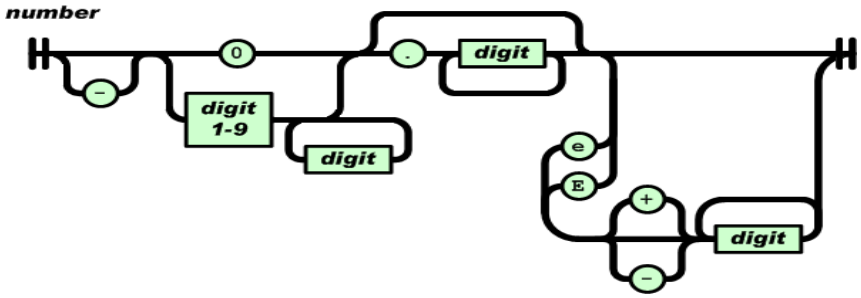
Sumber Gamber : *http://json.org/*

Gambar 2. 15 Format JSON *String*

String adalah urutan karakter kosong atau karakter *unicode* yang diapit oleh tanda kutip dua atau karekter khusus diawali dengan karakter "\" (garis miring terbalik). Karakter direpresentasikan sebagai *string* karakter tunggal. *String* sangat mirip dengan *string C* atau *Java*.

5. *Number*

Berikut adalah format *JSON number* yang dapat dilihat pada Gambar 2.16 :



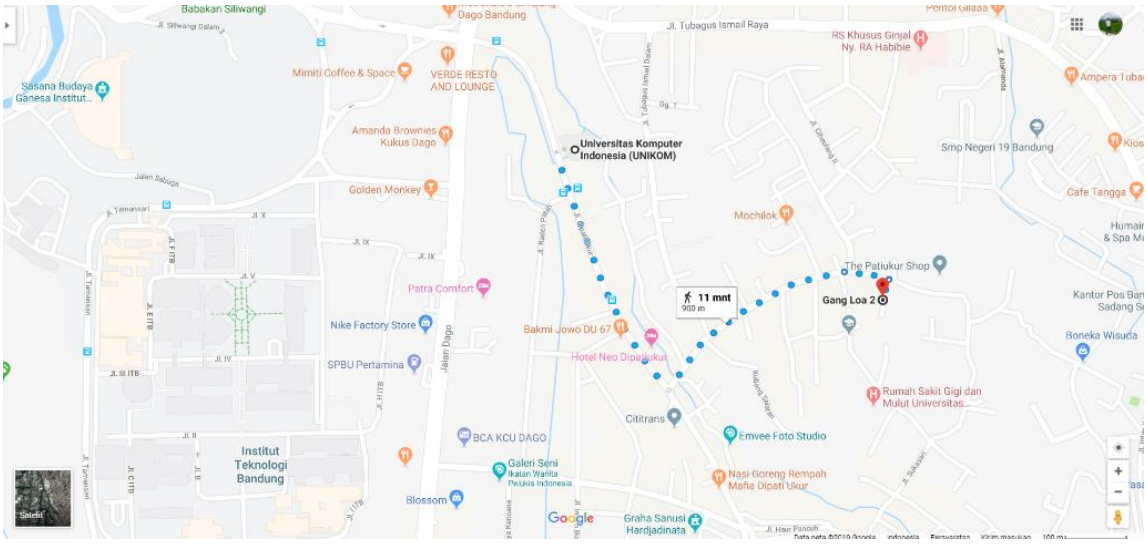
Sumber Gamber : *http://json.org/*

Gambar 2. 16 Format JSON *Number*

Number sangat mirip dengan angka *C* atau *Java*, kecuali format *octal* dan *heksadesimal* tidak digunakan.

2.13. *Google Maps*

Google maps merupakan aplikasi antarmuka yang dikeluarkan oleh *Google* yang dapat diakses lewat *javascript*. *Google Maps* menyediakan layanan berbasis peta yang sangat responsif dan mudah dalam penggunaannya. Dengan menggunakan g*oogle maps* ini, pengguna dapat dengan mudah mencari suatu lokasi serta dapat melakukan penelusuran *route* menuju lokasi yang diinginkan. (Sirenden dan Dachi, 2012). Ditingkat pemrograman, *Google maps* dapat dikembangkan dengan basis data, semua data yang terkait dengan titik lokasi disimpan dalam tabel dan dapat ditampilkan sesuai keinginan pengguna. Isi tabel yang berisi data posisi peta dapat ditampilkan Dengan menyajikan informasi lokasi yang menggunakan *google map*. Pengunjung tentunya akan mendapatkan informasi yang lebih detail terutama informasi lokasi perusahaan atau instansi [7].

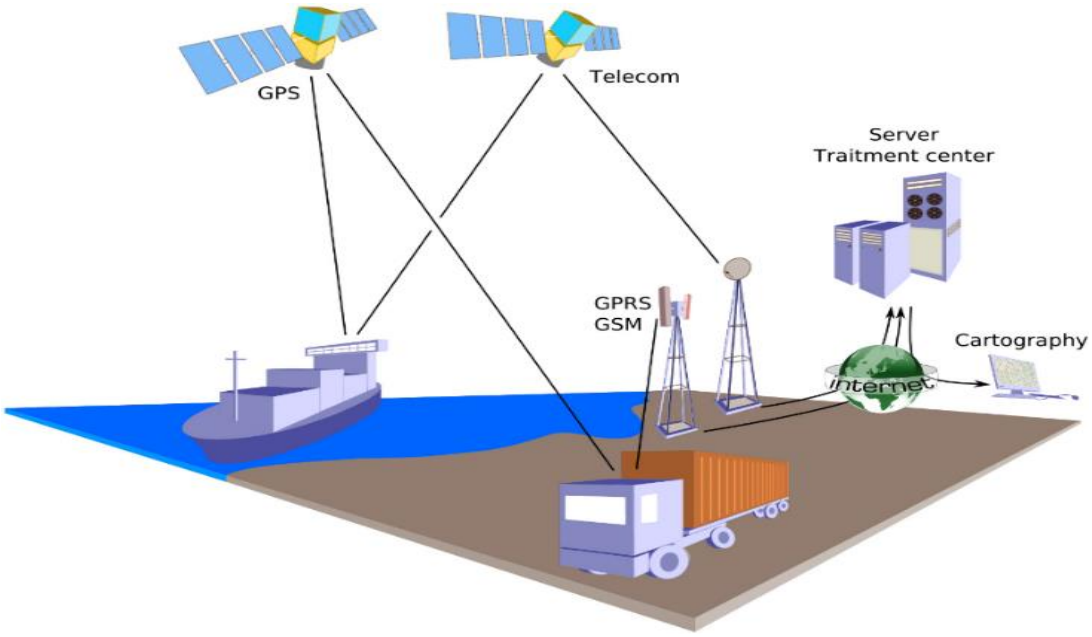


Sumber : *https://www.google.com/maps*

Gambar 2. 17 Tampilan *Google Maps*

2.14. *Geolocation* API

Geolocation adalah identifikasi lokasi geografis suatu objek pada dunia nyata dan dapat mendeteksi lokasi keberadaan kita juga pencarian rute sebagai informasi bagi pengguna dalam perjalan dengan menggunakan koneksi internet [20]. *Geolocation* memberikan informasi perangkat seperti garis lintang (*Latitude*) dan bujur (*Longitude*), sumber informasi umum didapatkan melalui *Global Positioning System* (GPS), sinyal jaringan seperti IP, RFID, WIFI, alamat *MAC Bluetooth* dan ID seluler GSM/CDMA [19].

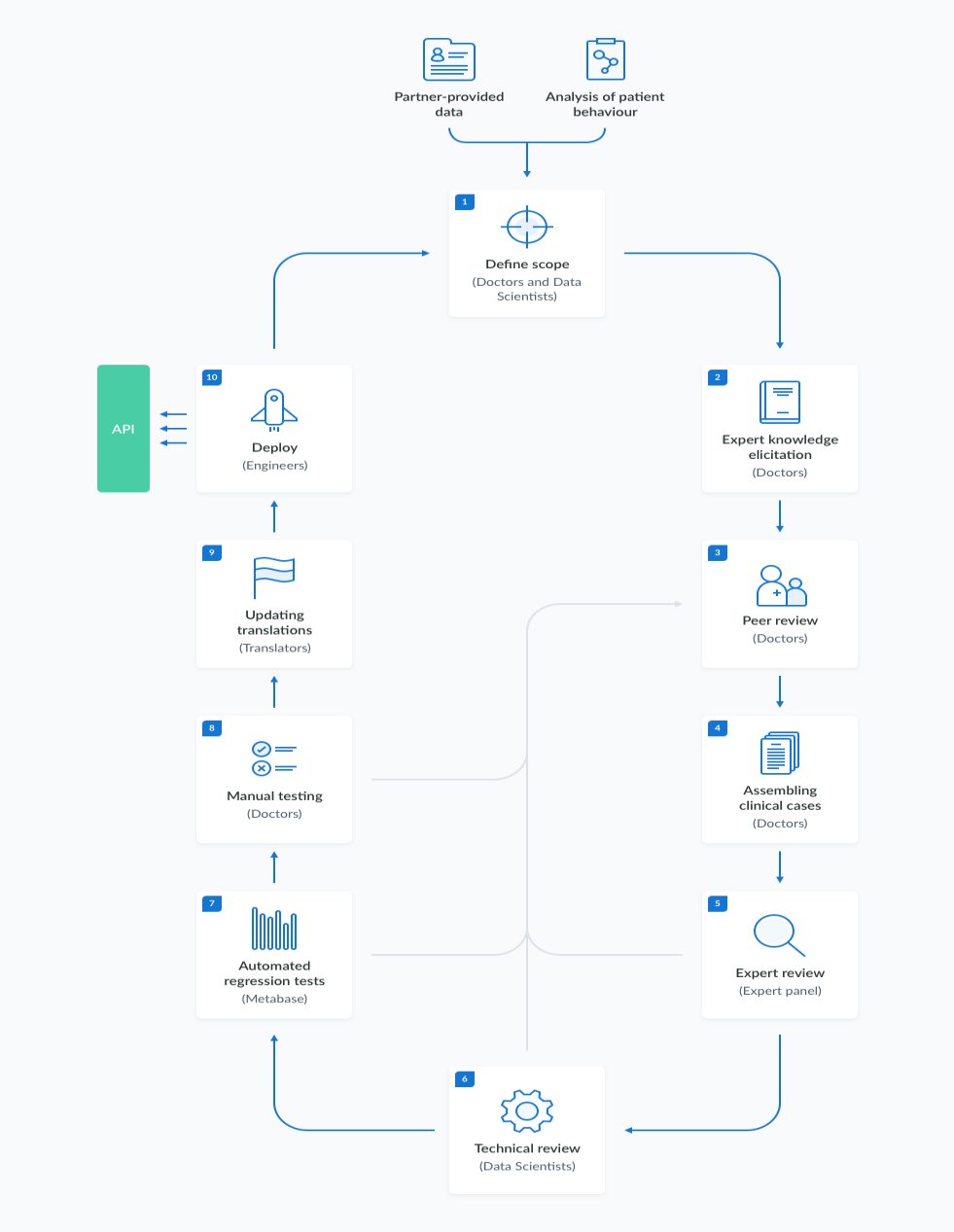


Sumber *: https://en.wikipedia.org/wiki/Geolocation#/media/File:Geolocation.png*

Gambar 2. 18 *Geolocation* API

2.15. *Infermedica* API

Infermedica adalah sebuah *Aplication Proggramming Interface* yang memungkinkan untuk di aplikasikan kedalam sebuah rancangan aplikasi untuk kesehatan, singkatnya infermedica menyediakan API untuk melakukan *triage* dan diagnosis medis awal yang dapat membantu menerapkan pemeriksaan gejala dengan mengirimkan data kondisi pasien seperti gejala, faktor resiko, hasil tes laboratorium atau demografi, mesin AI dalam *infermedica* akan mengalisis data yang dikirimkan dan akan memberi daftar kemungkinan kondisi dari hasil pengamatan yang relevan [20].



Sumber : *https://developer.infermedica.com/*

Gambar 2. 19 Struktur Kerja *Infermedica* API

2.16. *Firebase* API

Firebase adalah layanan infrastruktur (*Backend as a Service*) BaaS yang menawarkan kemudahan kepada para pengembang perangkat lunak dalam membangun aplikasi yang lebih baik serta mengembang bisnis yang sukses dengan memanfaatkan seluruh fitur kompleternya, beberapa fitur tersebut adalah :

1. *Analiytics*
2. *Cloud Messaging*
3. *Aunthentication*
4. *Realtime Database*
5. *Storage*
6. *Hosting*
7. *Test Lab*
8. *Crash Reporting*
9. *Notifications*
10. *Remote Config*
11. *App Indexing*
12. *Dynamic Links*
13. *Invites*
14. *AdWords*
15. *AMnob*

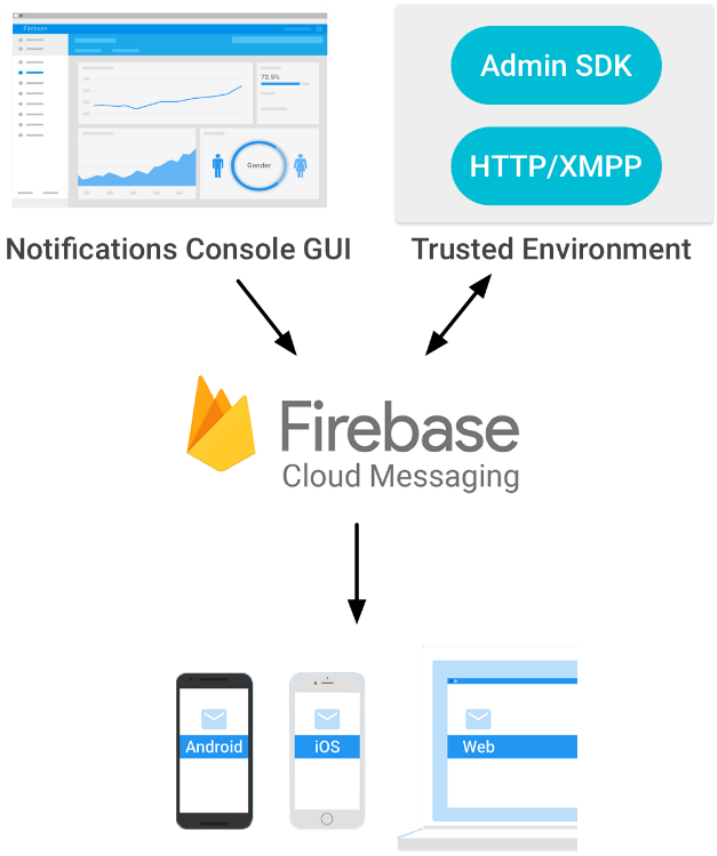
Seluruh fitur diatas dikemas pada sebah *SDK Firebase* sehingga para pengembang dapat dengan mudah menggunakan SDK ini untuk memecahkan masalah melalui aplikasi yang dikembangkannya dan tidak menghabiskan begitu banyak waktu untuk membangun infrastruktur yang begitu kompleks [21].

2.17.1. *Firebase Cloud Messaging* (FCM)

Firebase Cloud Messaging (FCM) adalah solusi pengiriman pesan lintas platform yang memungkinkan Anda mengirimkan pesan dengan tepercaya tanpa biaya. Dengan FCM, Anda dapat memberi tahu aplikasi klien bahwa email baru atau data lainnya tersedia untuk disinkronkan. Anda dapat mengirim pesan notifikasi untuk mendorong interaksi kembali dan retensi pengguna. Untuk kasus penggunaan seperti instant messaging, pesan dapat mentransfer payload hingga 4 KB ke aplikasi klien [21].

Implementasi FCM mencakup dua komponen utama untuk mengirim dan menerima pesan:

1. Lingkungan tepercaya seperti *Cloud Functions for Firebase* atau server aplikasi yang akan digunakan untuk membuat, menargetkan, dan mengirim pesan.
2. Aplikasi klien *iOS*, *Android*, atau *Web* (*JavaScript*) yang menerima pesan.



Sumber : https://firebase.google.com/docs/cloud-messaging

Gambar 2. 20 Cara Kerja *Firebase Cloud Messaging*

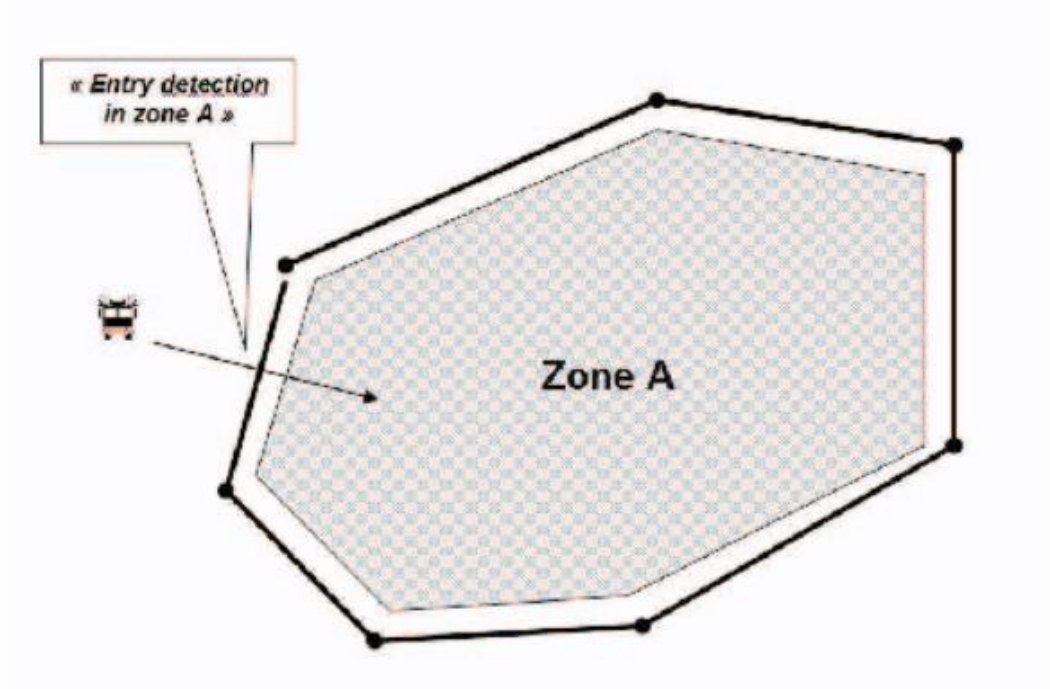
2.17. *Geofencing* API

Geofencing merupakan perangkat lunak yang digunakan bersamaan dengan *global positioning system* (GPS) dalam menentukan batas-batas geografis atau parameter virtual dari suatu peta. Program yang menggunakan *geofencing* dapat mengatur suatu *triggers* yang dapat memberikan informasi atau notifikasi apabila suatu target tertentu masuk atau keluar dari suatu batasan yang telah ditetapkan sebelumnya. Beberapa teknik dari geofencing adalah Geofence Area, *Proximity with Point of Interest, Route adherence*, dan *Route and schedule adherence* [22].

Salah satu cara untuk melakukan geofencing dengan memasang GPS *receiver* untuk dilacak ke objek dan menggunakan data GPS dari *receiver* untuk menentukan dimana objek tersebut berada yang dibandingkan dengan lokasi *geofence*. Fungsi utama *geofencing* yaitu untuk melakukan pemantauan jarak jauh (*monitoring*) suatu perangkat mobile dari peta *virtual* ketika perangkat virtual keluar atau memasuki daerah yang dibatasi *geofence* (pagar *virtual*) [22].

2.18.1. *Geofenced Area*

Teknik menyediakan monitoring otomatis dari objek *mobile* yang bergerak di sekitar ataupun berada dalam area *geofence*. Alarm akan berbunyi ketika perangkat mobile memasuki ataupun keluar dari batas (*boundery*) yang telah ditetapkan. Koordinat dari beberapa titik yang dibutuhkan untuk menentukan area *geofence*. Koordinat ini menjadi sumber perhitungan algoritma yang memungkinkan untuk pemberian peringatan, baik bersifat *inclusive* atau *exclusive* dari *geofence*.



Gambar 2. 21 *Geofence Area* Aktif di sekitar Wilayah A

2.18. *Unified Modeling Language* (UML)

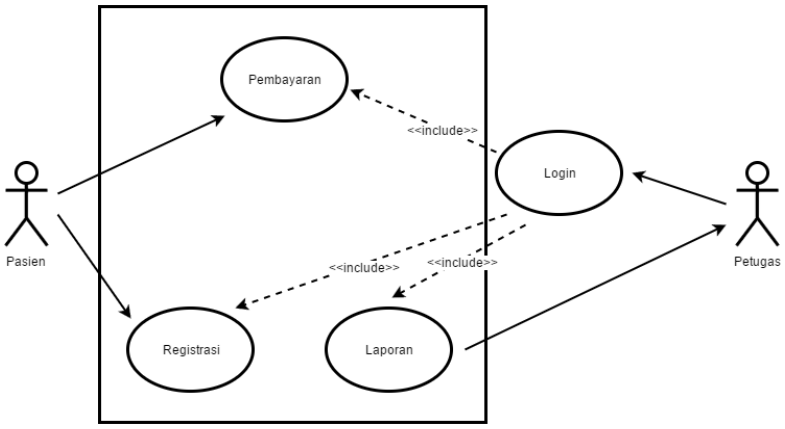
Unified Modeling Language (UML) adalah bahasa pemodelan visual yang digunakan untuk menspesifikasikan, menvisualisasikan, membangun, dan mendokumentasikan rancangan dari suatu sistem perangkat lunak (*Rumbaugh, Jacobson, & Booch*, 2005) [23].

Pemodelan memberikan gambaran yang jelas mengenai sistem yang akan dibangun baik dari sisi struktural ataupun fungsional. UML dapat diterapkan pada semua model pengembangan, tingkatan siklus sistem, dan berbagai macam domain aplikasi. Dalam UML terdapat konsep semantik, notasi, dan panduan masing-masing diagram. UML bertujuan menyatukan teknik-teknik pemodelan berorientasi objek-objek menjadi terstandarisasi [23].

Untuk mendapatkan banyak pandangan terhadap sistem yang akan dibangun, UML menyediakan beberapa diagram visual yang menunjukkan berbagai aspek dalam sistem. Ada beberapa diagram yang disediakan dalam UML antara lain :

2.19.1. *Use Case Diagram*

Use case diagram menyajikan interakasi antara use case dan aktor. Dimana, aktor dapat berupa orang, peralatan, atau sistem lain yang berinteraksi dengan sistem yang sedang dibangun. *Use case diagram* menggambarkan fungsionalitas sistem atau persyaratan-persyaratan yang harus dipenuhi sistem dari pandangan pemakai [23].



Gambar 2. 22 Contoh *Use Case Diagram*

2.19.2. Skenario *Use Case Diagram*

Skenario *Use case* merupakan hasil instansiasi dan penjelasan dari setiap *Use case*. Skenario *Use case* menceritakan detail yang terjadi. Format skenario *Use case* adalah berbentuk tabel.

Komponen-komponen Skenario *Use case* :

1. *Name*: Memberikan penjelasan singkat tentang nama dari *use case*.
2. *Actors:* Daftar aktor yang dapat mengakses *use case.*
3. *Goals*: Menjelaskan apa yang aktor coba dapatkan dari *use case*.
4. *Preconditions*: Kondisi sistem sebelum *use case* dijalankan.
5. *Summary*: Memberikan penjelasan singkat tentang deskripsi informal dari *use case*.
6. *Steps*: Menjelaskan setiap langkah yang dijalankan pada *use case*.
7. *Post conditions*: Kondisi sistem setelah *use case* dijalankan.

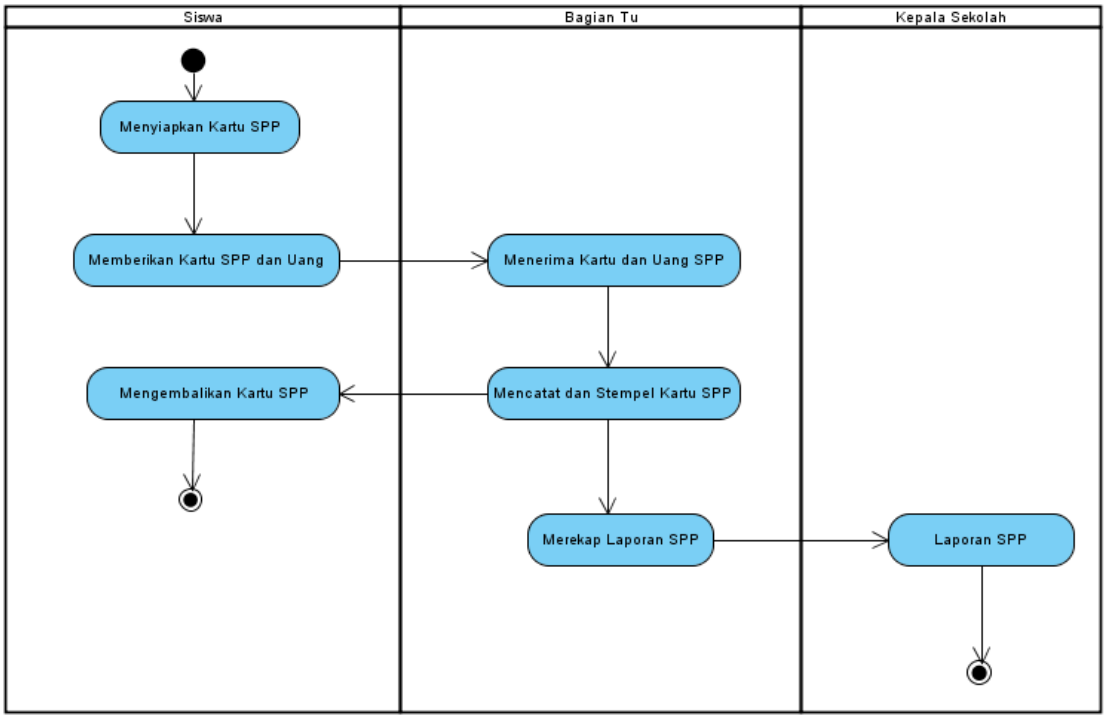
Tabel 2. 2 Contoh Skenario *Use Case Diagram*

Use case Name	Melakukan Login	
Related Requirements	SKPL – F – 02	
Goals	Menampilkan halaman utama aplikasi	
Preconditions	-	
Successful End Condition	Berhasil masuk kedalam system	
Failed End Condition	Gagal masuk kedalam system	
Primary Actors	Pengguna Umum, Pengguna Anggota	
Main Flow	*Steps*	*Actions*
	1	Aktor membuka halaman login
	2	Sistem menampilkan form login
	3	Aktor mengisi username dan password
	4	Sistem melakukan validasi username dan password
	5	Sistem menampilkan halaman utama aplikasi
Extension	*Steps*	*Branching Action*
	4.1	Sistem menampilkan pesan kesalahan

2.19.3. *Activity Diagram*

Activity diagram merupakan diagram yang menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. *Activity* diagram juga digunakan untuk mendefinisikan urutan atau pengelompokan tampilan dari sistem/*user*

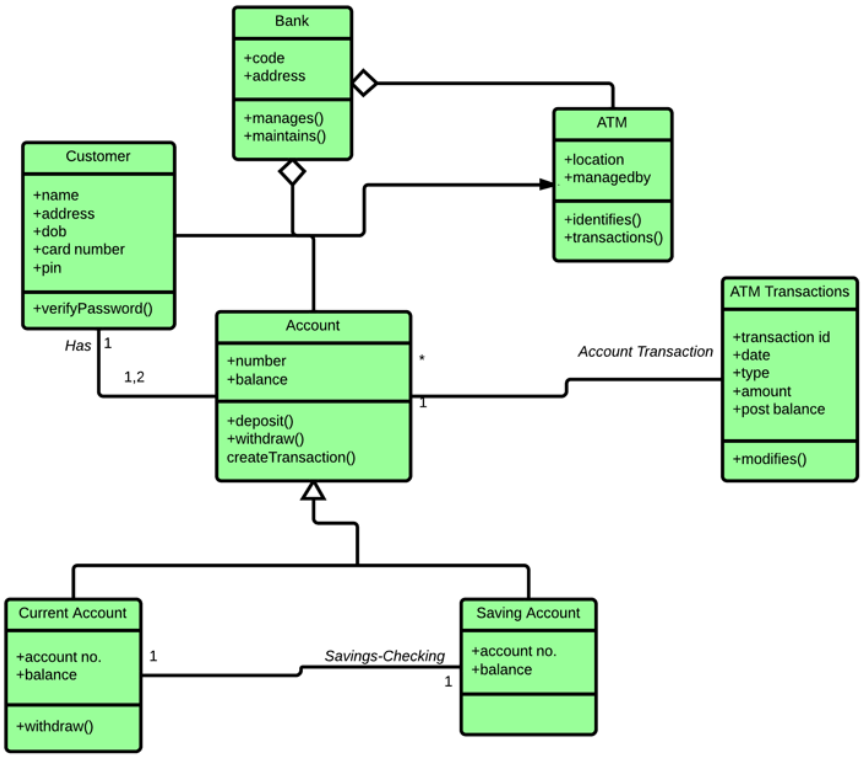
interface dimana setiap aktivitas dianggap memiliki sebuah rancangan antar muka tampilan serta rancang menu yang ditampilkan pada perangakat lunak [23].



Gambar 2. 23 Contoh *Activity Diagram*

2.19.4. *Class Diagram*

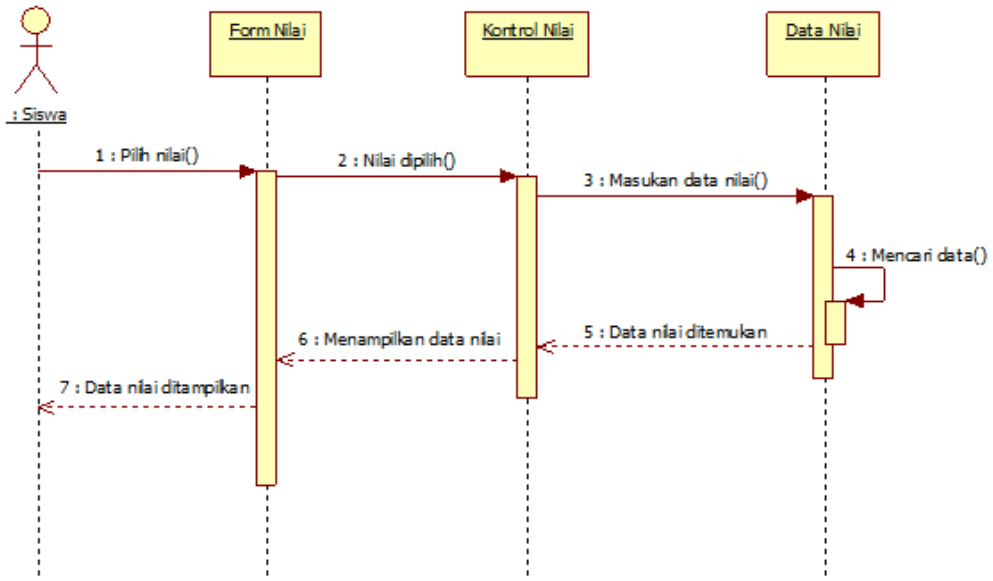
Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. *Class* diagram memiliki apa yang disebut atribut dan metode atau operasi. *Class* diagram mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat di antara mereka. *Class* Diagram juga menunjukkan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut [23].



Gambar 2. 24 Contoh *Class Diagram*

2.19.5. *Sequence Diagram*

Sequence diagram merupakan salah satu diagram *interaction* yang menjelaskan bagaimana suatu operasi itu dilakukan, *message* (pesan) apa yang dikirim dan kapan pelaksanaannya. Diagram ini diatur berdasarkan waktu dan objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut [23].



Gambar 2. 25 Contoh *Sequence Diagram*