BAB 2
TINJAUAN PUSTAKA

2.1. Talent Solution API

Cloud Talent Solution adalah layanan yang menghadirkan pembelajaran untuk pengalaman pencarian kerja, mengembalikan hasil yang berkualitas tinggi kepada pencari kerja yang jauh melampaui batasan metode berbasis kata kunci yang khas. Setelah terintegrasi dengan konten pekerjaan, *Cloud Talent Solution* secara otomatis mendeteksi dan memasukkan berbagai jenis data, seperti judul terkait, senioritas, dan industri [3].Selain teknologi pencarian yang mendasarinya, *Cloud Talent Solution* menyertakan beberapa fitur bawaan, seperti:

1. Pekerjaan yang diindeks menggunakan bidang terstruktur untuk lebih mewakili konten pekerjaan.
2. Mode yang disukai dari penelusuran transit untuk pekerjaan berdasarkan waktu perjalanan
3. Jumlah histogram pekerjaan di berbagai aspek berbeda.
4. Memamerkan pekerjaan unggulan.
5. Optimasi hasil pencari kerja pasif untuk pemberitahuan email.
6. Mencari pekerjaan dalam berbagai bahasa.
7. Judul autocompletion yang disarankan untuk perusahaan dan pekerjaan yang secara khusus terbuka dalam korpus pekerjaan
8. Koreksi ejaan, pengenalan jargon perusahaan, pengayaan pekerjaan, dan banyak lagi.

Cara menentukan id proyek dengan menggunakan window, yaitu:

1. Command prompt

```
set GOOGLE_CLOUD_PROJECT="your-project-id"
```

2. Selanjutnya, buka quickstart untuk membuat perusahaan dan pekerjaan, untuk memulai mengintegrasikan Cloud Talent Solution:

```
<dependency>
      <groupId>com.google.apis</groupId>
      <artifactId>google-api-services-jobs</artifactId>
      <version>LATEST</version>
</dependency>
```

3. Lalu menginstal client libery

```
private static final JsonFactory JSON_FACTORY = new
JacksonFactory();
private static final NetHttpTransport NET_HTTP_TRANSPORT =
new NetHttpTransport();
private static final String SCOPES =
"https://www.googleapis.com/auth/jobs";
private static final String DEFAULT_PROJECT_ID =
    "projects/" + System.getenv("GOOGLE_CLOUD_PROJECT");

private static CloudTalentSolution talentSolutionClient =
createTalentSolutionClient(
    generateCredential());

private static CloudTalentSolution
createTalentSolutionClient(GoogleCredential credential) {
  String url = "https://jobs.googleapis.com";
  return new CloudTalentSolution.Builder(
      NET_HTTP_TRANSPORT, JSON_FACTORY,
setHttpTimeout(credential))
      .setApplicationName("JobServiceClientSamples")
      .setRootUrl(url)
      .build();
}

private static GoogleCredential generateCredential() {
  try {
    // Credentials could be downloaded after creating
service account
    // set the `GOOGLE_APPLICATION_CREDENTIALS` environment
variable, for example:
    // export
GOOGLE_APPLICATION_CREDENTIALS=/path/to/your/key.json
```

```
    return GoogleCredential
        .getApplicationDefault(NET_HTTP_TRANSPORT,
JSON_FACTORY)
        .createScoped(Collections.singleton(SCOPES));
  } catch (Exception e) {
    System.out.print("Error in generating credential");
    throw new RuntimeException(e);
  }
}

private static HttpRequestInitializer setHttpTimeout(
    final HttpRequestInitializer requestInitializer) {
  return request -> {
    requestInitializer.initialize(request);
    request.setHeaders(new HttpHeaders().set("X-GFE-SSL",
"yes"));
    request.setConnectTimeout(1 * 60000); // 1 minute
connect timeout
    request.setReadTimeout(1 * 60000); // 1 minute read
timeout
  };
}
public static CloudTalentSolution getTalentSolutionClient()
{
  return talentSolutionClient;
}
public static void main(String... args) throws Exception {
  try {
    ListCompaniesResponse listCompaniesResponse =
talentSolutionClient.projects().companies()
        .list(DEFAULT_PROJECT_ID)
        .execute();
    System.out.println("Request Id is " +
listCompaniesResponse.getMetadata().getRequestId());
    if (listCompaniesResponse.getCompanies() != null) {
      for (Company company :
listCompaniesResponse.getCompanies()) {
        System.out.println(company.getName());
      }
```

```
    }
  } catch (IOException e) {
    System.out.println("Got exception while listing
companies");
    throw e;
  }
}
```

2.2. Landasan Teori

Landasan teori menjelaskan beberapa teori-teori dan penjelasan yang berkaitan dengan aplikasi atau media yang akan dibangun. Landasan teori yang digunakan dalam penyususnan aplikasi Sistem rekomendasi lowongan kerja memanfaatkan Talent Solution API berbasis android. Pengertian Aplikasi, Pengertian Mobile Android, Pengertian Java, *Talent Solution API*, GPS, UML.

2.2.1. Aplikasi

Aplikasi merupakan program yang berisikan perintah-perintah untuk melaksanakan pengolahan data, aplikasi secara umum adalah suatu proses dari cara manual yang ditransformasikan ke komputer dengan membuat sistem atau program agar data diolah lebih berdaya guna secara optimal. Aplikasi (*application*) juga adalah *software* yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya *Microsoft Word* dan *Microsoft Excel* [5].

Dari pengertian di atas dapat disimpulkan bahwa aplikasi merupakan *software* yang ditransformasikan ke komputer yang berisikan perintah-perintah yang berfungsi untuk melakukan berbagai bentuk pekerjaan atau tugas-tugas tertentu seperti penerapan, penggunaan dan penambahan data. Selain itu aplikasi dapat mengintegrasikan berbagai kemampuan komputer.

2.2.2. Android

Android Merupakan sebuah sistem operasi yang berbasis Linux untuk telepon seluler seperti telepon pintar dan komputer tablet. Android menyediakan platform terbuka bagi para pengembang untuk menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android

Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah *Open Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode–kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler. Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau *Google Mail Services* (GMS) dan kedua adalah yang benar–benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD) [6].

Android SDK (*Software Development Kit*) menyediakan *tools* dan API yang diperlukan untuk mengembangkan aplikasi pada *platform* android dengan menggunakan bahasa pemrograman Java.

Android memiliki paradigma pemrograman lain tidak seperti paradigma pemrograman biasa di mana aplikasi yang dijalankan pada fungsi main(), sistem android menjalankan kode dalam *method Activity* dengan menerapkan metode *callback* tertentu yang sesuai dengan tahap tertentu dari siklus hidup. Setiap aplikasi yang berjalan dalam sistem operasi android memiliki siklus hidup yang berbeda dengan aplikasi desktop atau *web*, Hal ini dikarenakan aplikasi *mobile* memiliki tingkat interupsi proses yang lumayan tinggi seperti ketika *handling* panggilan masuk aplikasi diharuskan menghentikan proses sementara, Penerapan siklus hidup juga berguna untuk memastikan aplikasi tidak menghabiskan sumber daya baterai pengguna.

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

1. Multi-proses dan *App Widgets*

   Sistem operasi android tidak melarang prosesor menjalankan lebih dari satu aplikasi dalam satu waktu. Sistem operasi android dapat mengatur aplikasi dan thread yang

berjalan secara *multitasking.* Keuntungan yang didapat adalah ketika aplikasi berjalan dan berinteraksi dengan pengguna di *layer* depan sistem operasi, proses dari aplikasi lain dapat berjalan untuk melakukan pembaruan informasi. Sebagai contoh misalnya ketika pengguna memainkan *game,* proses lain dapat berjalan di belakang aplikasi seperti memeriksa harga saham dan memunculkan peringatan.

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

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

2. *Touch Gestures* dan *Multi-touch*

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

3. *Hard* dan *Soft Keyboard*

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

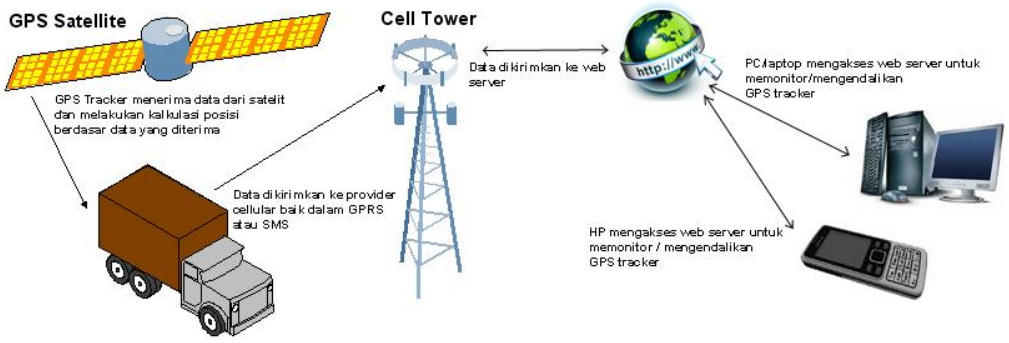
Gambar 2. 1 Logo Android

2.2.3. Global Positioning System (GPS)

Global Positioning System adalah sistem navigasi berbasis satelit, pertama kali diperkenalkan mulai tahun 1978. Layanan GPS dahulu hanya dipergunakan untuk keperluan militer namun mulai terbuka untuk publik. Satelit GPS berkekuatan energi sinar matahari, memiliki baterai cadangan untuk menjaga agar tetap berjalan pada saat gerhana matahari atau pada saat tidak ada energi matahari dan memiliki roket penguat kecil pada masing-masing satelit agar dapat mengorbit tepat pada tempatnya.

Satelit-satelit GPS harus selalu berada pada posisi orbit yang tepat untuk menjaga akurasi data yang dikirim ke GPS *receiver*, sehingga harus selalu dipelihara agar posisinya tepat. Posisi satelit-satelit tersebut selalu dipantau oleh stasiun pengendali. Stasiun-stasiun pengendali di bumi ada di Hawaii, Ascension Islan, Diego Garcia, Kwajalein dan Colorado Spring. Untuk dapat mengetahui posisi seseorang maka diperlukan alat yang diberi nama GPS *receiver* yang berfungsi untuk menerima sinyal yang dikirim dari satelit GPS. GPS *receiver* mengambil informasi tersebut dan melakukan perhitungan triangulation untuk menentukan lokasi pengguna dengan tepat. GPS *receiver* membandingkan waktu sinyal dikirim dengan waktu sinyal tersebut diterima untuk mengetahui jarak satelit. Dengan mengetahui jarak tersebut, GPS *receiver* dapat melakukan perhitungan dan menentukan posisi pengguna dan menampilkan dalam peta elektronik. Setelah menentukan posisi pengguna, selanjutnya GPS dapat menghitung informasi lain, seperti kecepatan, arah yang dituju, jalur, tujuan

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



Gambar 2. 2 Sistem Kerja GPS

2.2.4.1 Android SDK *(Software Development Kit)*

Android SDK adalah tools API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Android merupakan subset perangkat lunak

untuk ponsel yang meliputi sistem operasi, *middleware* dan aplikasi kunci yang di-*release* oleh Google. Saat ini disediakan Android SDK (*Software Development Kit*) sebagai alat bantu dan API untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi-netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan [7].

2.2.4.2. ADT *(Android Development Tools)*

Android Develpoment Tools adalah *plugin* yang di desain untuk IDE Eclipse yang memberikan kemudahan dalam mengembangkan aplikasi android dengan menggunakan IDE Eclipse. Dengan menggunakan ADT untuk *Eclipse* akan memudahkan dalam membuat aplikasi *project Android*, membuat GUI aplikasi, dan menambahkan komponen-komponen yang lainnya. Mengembangkan aplikasi Android dengan menggunakan ADT di *Eclipse* sangat dianjurkan dan sangat mudah untuk memulai mengembangkan aplikasi Android

Semakin tinggi platform Android yang digunakan, dianjurkan menggunakan ADT yang lebih terbaru, karena biasanya munculnya platform baru diikuti oleh munculnya versi ADT yang terbaru.

2.2.4.3. Arsitektur Android

Secara garis besar Arsitektur Android dapat di jelaskan dan di gambarkan sebagai berikut :

1. *Application and Widgets*

Application dan Widgets adalah layer dimana user berhubungan dengan aplikasi saja, dimana biasanya user men-download aplikasi, melakukan instalasi dan menjalankan aplikasi.

2. *Applications Frameworks*

Android adalah "*Open Development Platform*" yaitu Android menawarkan kepada pengembang atau member kemampuan untuk membangun aplikasi yang inovatif.

3. *Libraries*

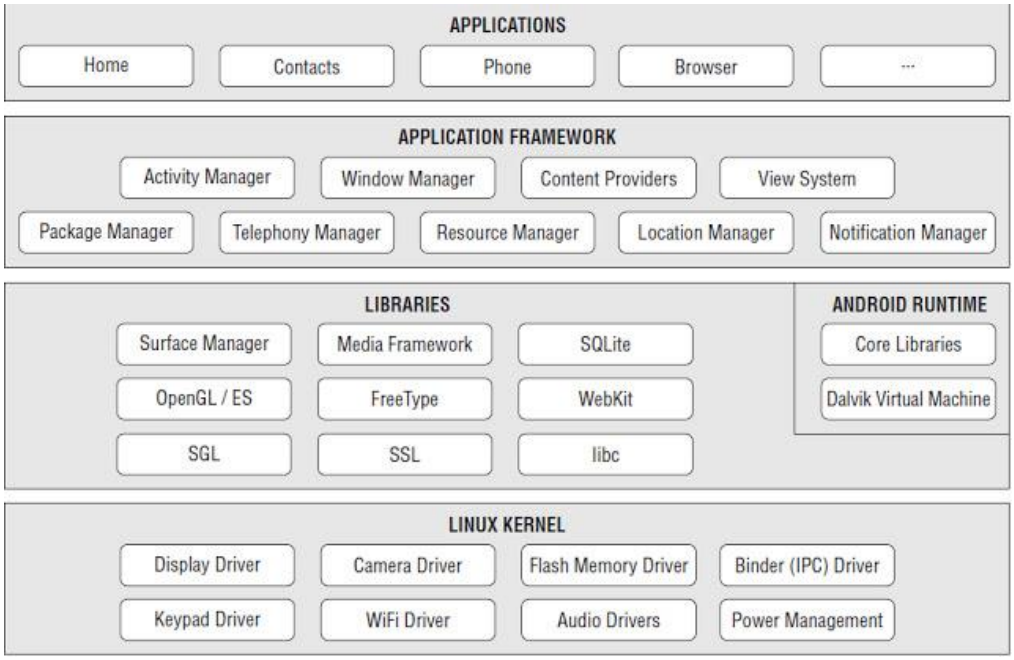
Libraries adalah *layer* dimana fitur-fitur Android berada, biasanya para pembuat aplikasi mengakses *libraries* untuk menjalankan aplikasinya.

4. *Android Run Time*

Layer yang membuat aplikasi Android dapat djalankan dimana dalam prosesnya menggunakan Implementasi Linux.

5. *Linux Kernel*

Linux Kernel adalah *layer* dimana inti dari *operating system* dari Android itu berada. Berisi *file* sistem yang mengatur sistem *processing*, *memory*, *resource*, *drivers*, dan sistem-sistem opersi Android lainnya.



Gambar 2. 3 Arsitektur Android

2.2.4.4. Versi Android

Android telah mengalami sejumlah pembaruan sejak pertama kali dirilis. Rata-rata, versi terbaru dari Android dirilis setiap 6 bulan. Tabel 2.1 menunjukkan beberapa jenis

Android dan nama kodenya. Penamaan kode menggunakan nama makanan dan huruf depannya urut sesuai abjad.

Tabel 2. 1 Daftar Android

Versi	Tanggal Rilis	Kode
1.1	9 Februari 2009	
1.5	30 April 2009	Cupcake
1.6	15 September 2009	Donut
2.0/2.1	26 Oktober 2009	Eclair
2.2	20 Mei 2010	Frozen Yoghurt (Froyo)
2.3	6 Desember 2010	Gingerbread
3.0	22 Februari 2011	Honeycomb
4.0	19 Oktober 2011	Ice Cream Sandwich
4.1	27 Juni 2012	Jelly Bean
4.2	29 Oktober 2012	Jelly Bean
4.3	24 Juli 2013	Jelly Bean
4.4	3 September 2013	Kitkat
5.0	Pada Tahun 2014	Lollipop
6.0	28 Mei 2015	Marshmallow
7.0	Pada Tahun 2016	Nougat

2.2.5. Android Studio

Android Studio adalah IDE resmi untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA. Di atas IntelliJ yang kuat *code editor* dan pengembang alat, Android Studio menawarkan lebih banyak fitur yang meningkatkan produktivitas Anda ketika membangun aplikasi Android, seperti :

1. Sebuah *fleksibel* berbasis *Gradle* membangun sistem
2. Membangun varian dan beberapa generasi file APK
3. Kode template untuk membantu Anda membangun fitur aplikasi umum
4. Sebuah *layout editor* kaya dengan dukungan untuk *drag* dan *drop tema editing*

5. Alat *Lint* untuk menangkap kinerja, kegunaan, kompatibilitas versi, dan masalah lainnya
6. Kode menyusut dengan ProGuard dan sumber daya menyusut dengan Gradle
7. Built-in dukungan untuk Google *Cloud Platform*, sehingga mudah untuk mengintegrasikan Google *Cloud Messaging* dan App *Engine*.

2.2.5.1. Struktur Proyek

Setiap proyek di Android Studio berisi satu atau lebih modul dengan file kode sumber dan file sumber daya. Berbagai jenis modul meliputi :

1. *Android app modules*
2. *Test modules*
3. *Library modules*
4. *App Engine modules*

Secara *default*, Android Studio menampilkan file proyek Anda dalam tampilan proyek Android. Pandangan ini diselenggarakan oleh modul untuk menyediakan akses cepat ke file kunci sumber proyek Anda. Semua file membangun terlihat di tingkat atas di bawah *Script Gradle* dan setiap modul aplikasi mengandung tiga unsur berikut :

1. Manifests : file Manifest.
2. Java : Sumber file kode.
3. Res : file Sumber Daya.

Struktur proyek Android pada disk berbeda dari representasi pipih ini. Untuk melihat struktur file yang sebenarnya dari proyek, pilih Proyek dari Proyek *drop-down*. Anda juga dapat menyesuaikan tampilan dari file proyek untuk fokus pada aspek-aspek tertentu dari pengembangan aplikasi Anda. Misalnya, memilih Masalah tampilan proyek Anda menampilkan link ke file sumber yang berisi setiap diakui coding dan sintaksis kesalahan, seperti hilang elemen XML tag penutup dalam *file layout*.

2.2.5.2. Versi Android Studio

Versi Android Studio sebagai berikut :

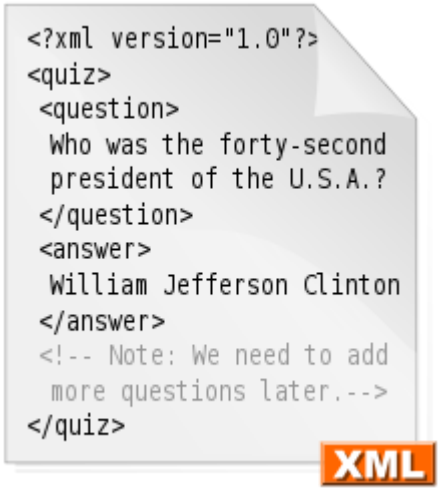
1. Android Studio v0.1.x (Mei 2013)
2. Android Studio v0.2.x (Juli 2013)

3. Android Studio v0.3.2 (Oktober 2013)
4. Android Studio v0.4.2 (Januari 2014)
5. Android Studio v0.4.6 (Maret 2014)
6. Android Studio v0.5.2 (Mei 2014)
7. Android Studio v0.8.0 (Juni 2014)
8. Android Studio v0.8.6 (Agustus 2014)
9. Android Studio v0.8.14 (Oktober 2014)
10. Android Studio v1.0 (Desember 2014)
11. Android Studio v1.0.1 (Desember 2014)
12. Android Studio v1.1.0 (Februari 2015)
13. Android Studio v1.2.0 (April 2015)
14. Android Studio v1.2.1 (Mei 2015)
15. Android Studio v1.2.2 (Juni 2015)
16. Android Studio v1.3.0 (Juli 2015)
17. Android Studio v1.3.1 (Agustus 2015)
18. Android Studio v1.3.2 (Agustus 2015)
19. Android Studio v1.4.0 (September 2015)
20. Android Studio v1.4.1 (Oktober 2015)
21. Android Studio v1.5.0 (November 2015)
22. Android Studio v1.5.1 (Desember 2015)
23. Android Studio v2.0.0 (April 2016)
24. Android Studio v2.1.0 (April 2016)
25. Android Studio v2.1.1 (Mei 2016)

2.2.6. XML *(Extensible Markup Language)*

XML (*Extensible Markup Language*) adalah bahasa markup untuk keperluan umum yang disarankan oleh W3C untuk membuat dokumen markup keperluan pertukaran data antar sistem yang beraneka ragam. XML merupakan kelanjutan dari HTML

(*HyperText Markup Language*) yang merupakan bahasa standar untuk melacak Internet.



Gambar 2. 4 Contoh Dokumen XML

2.2.6.1. Pengenalan XML

XML didesain untuk mempu menyimpan data secara ringkas dan mudah diatur. Kata kunci utama XML adalah data (jamak dari datum) yang jika diolah bisa memberikan informasi. XML menyediakan suatu cara terstandarisasi namun bisa dimodifikasi untuk menggambarkan isi dari dokumen. Dengan sendirinya, XML dapat digunakan untuk menggambarkan sembarang *view database*, tetapi dengan suatu cara yang standar.

2.2.6.2. Tipe XML

XML memiliki tiga tipe file, yaitu :

1. XML, merupakan standar format dari struktur berkas (file).
2. XSL, merupakan standar untuk memodifikasi data yang diimpor atau diekspor.
3. XSD, merupakan standar yang mendefinisikan struktur database dalam XML.

2.2.7. PHP

Menurut Anhar, ST (2010), “PHP singkatan dari Personal Hypertext Preprocessor yaitu bahasa pemrograman web server-side yang bersifat open source. PHP merupakan script yang terintegrasi dengan HTML dan berada pada server. PHP adalah script yang digunakan untuk membuat halaman website yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Mekanisme ini menyebabkan informasi yang diterima client selalu yang terbaru. Semua script PHP dieksekusi pada server dimana script tersebut dijalankan.

PHP pertama kali di kembangkan pada tahun 1995 oleh Rasmus Lerdorf, namun sekarang di ambil oleh oleh The PHP Group. Pada awalnya PHP adalah singkatan dari Personal Home Page, namun dalam perkembangannya, di ubah menjadi PHP: Hypertext Preprocessor, sebuah kepanjangan rekursif [7].

PHP berjalan pada sisi server sehingga PHP disebut juga sebagai bahasa ServerSideScripting. Artinya bahwa dalam setiap/untuk menjalankan PHP, wajib adanya web server.PHP ini bersifat opensource sehingga dapat dipakai secara cuma-cuma dan mampu lintas platform, yaitu dapat berjalan pada sistem operasi Windows maupun Linux.PHP juga dibangun sebagai modul pada web apache dan sebagai binary yang dapat berjalan sebagai CGI. Pada pembangunan aplikasi pelaporan benacana kebakaran Kota Bandung khususnya halaman backend menggunakan script PHP untuk membuat halaman pelaporan. Berikut contoh skrip dasar PHP bisa dilihat pada gambar 2.5:

```
<html>
<head>
<title>Contoh Kode PHP </title>
</head>
<body>
<?php
echo ("Ini adalah Kode PHP yang diletakan di antara kode HTML");
?>
</body>
</html>
```

Gambar 2. 5 Contoh Skrip PHP

Menurut Vikram Vaswani (2004), PHP memiliki beberapa keunggulan, antara lain [8] :

1. Kinerja

   Script yang ditulis dalam PHP mengeksekusi lebih cepat dibandingkan yang ditulis dalam bahasa script lain.

2. Portabilitas

   PHP tersedia untuk UNIX, Microsoft Windows, Mac OS, dan OS/2.PHP merupakan program portable antar platform. Kemampuan untuk melakukan cross-platform merupakan salah satu keunggulan bagi lingkungan perusahaan yang multiplatform.

3. Kemudahan dalam penggunaan

   PHP adalah bahasa pemograman yang sangat canggih dan dilengkapi dengan 5000 fungsi. Hal ini merupakan salah satu alasan PHP disukai sebagai alat prototyping untuk aplikasi berbasis web.

4. Open Source

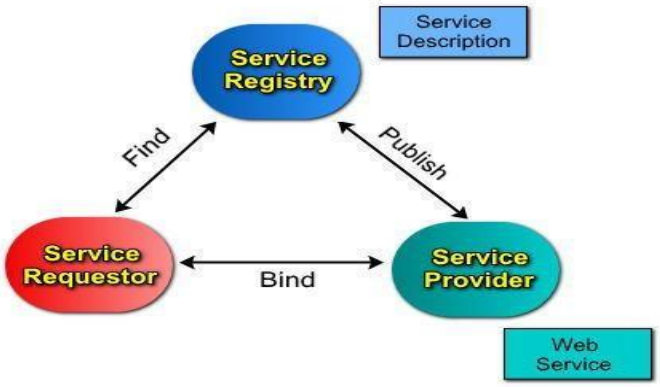
   PHP merupakan bahasa opensource dan tersedia secara bebas di web serta dapat digunakan tanpa biaya lisensi.

2.2.8. Web Service

Web service adalah salah satu bentuk sistem perangkat lunak yang didesain untuk mendukung interaksi mesin-ke-mesin melalui jaringan. *Web service* memiliki *interface* yang dideskripsikan dalam format yang dapat dibaca oleh mesin. Sistem-sistem lainnya berinteraksi dengan web service menggunakan pesan SOAP yang umumnya dikirim melalui HTTP dalam bentuk XML.

Definisi diatas diberikan oleh *World Wide Web Consortium*(W3C) yang merupakan badan yang menciptakan dan mengembangkan standar *web service*. Tetapi secara umum, *web service* tidak terbatas hanya pada standar SOAP saja. Salah satu pustaka yang mengulas lengkap tentang *web service* menyebutkan definisi yang lebih umum: *web service* adalah aplikasi yang diakses melalui internet menggunakan protokol standar internet dan menggunakan XML sebagai format pesannya [9].

2.2.8.1. Arsitektur Web Service



Gambar 2. 6 Arsitektur Web Service

Pada gambar diatas, ada tiga komponen yang membuat *web service* berjalan. Ketiga komponen itu adalah :

1. *Service provider*, merupakan pemilik *web service* yang berfungsi menyediakan kumpulan operasi dari *web service*.
2. *Service requestor*, merupakan aplikasi yang bertindak sebagai klien dari *web service* yang mencari dan memulai interaksi terhadap layanan yang disediakan.
3. *Service registry*, merupakan tempat dimana *service provider* mempublikasikan layanannya. Pada arsitektur *web service*, *service registry* bersifat optional. Teknologi *web service* memungkinkan kita dapat menghubungkan berbagai jenis software yang memiliki platform dan sistem operasi yang berbeda.

2.2.9. JSON *(JavaScript Object Notation)*

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer. 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. Kelebihan-kelebihan dari JSON adalah :

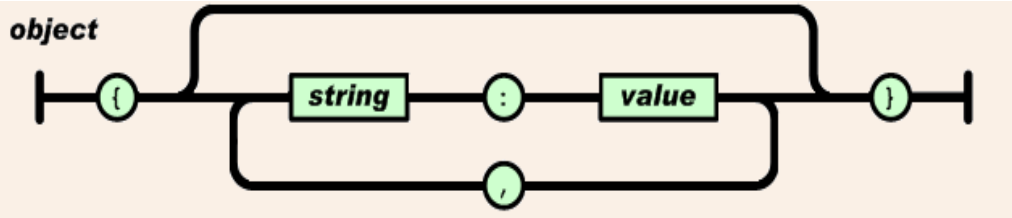
1. Format Penulisan : Untuk merepresentasikan sebuah struktur data yang rumit dan berbentuk hirarkis penulisan JSON relatif lebih terstruktur dan mudah.
2. Ukuran karakter yang dibutuhkan JSON lebih kecil dibandingkan XML untuk data yang sama.
3. Proses parsing merupakan proses pengenalan token atau bagian-bagian kecil dalam rangkaian dokumen XML/JSON.

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*).

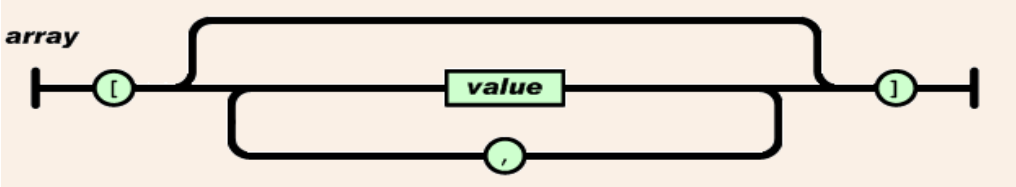
Struktur-struktur data ini disebut sebagai struktur data universal. Pada dasarnya, semua bahasa pemprograman moderen mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini. JSON menggunakan bentuk sebagai berikut [6]:

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).



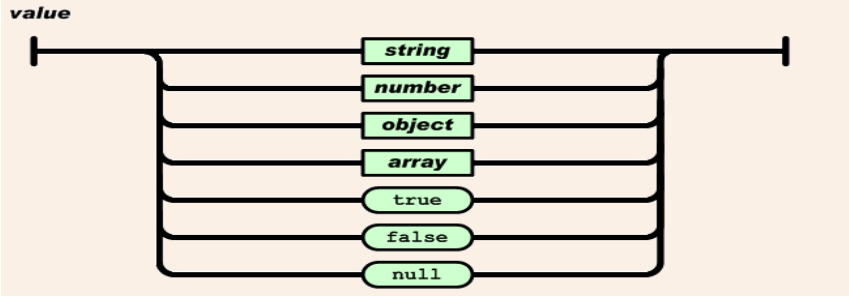
Gambar 2. 7 Objek JSON

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



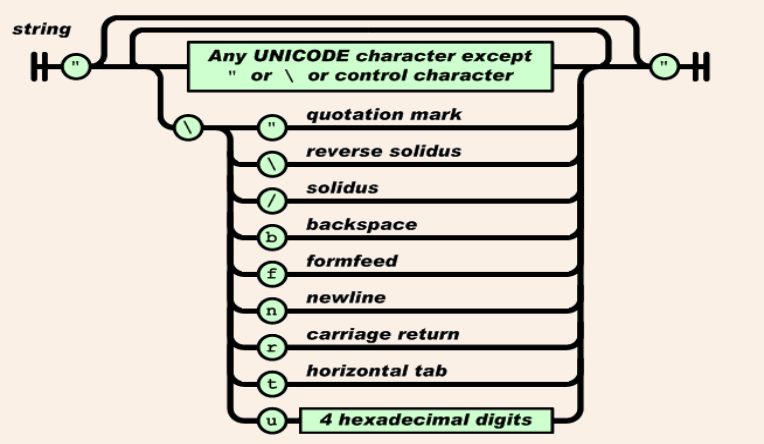
Gambar 2. 8 Array JSON

Nilai (value) dapat berupa sebuah string dalam tanda kutip ganda, atau angka, atau true atau false atau null, atau sebuah objek atau sebuah larik. Struktur- struktur tersebut dapat disusun bertingkat.



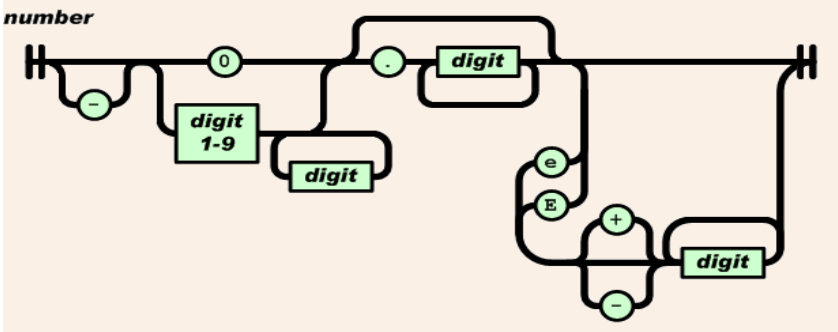
Gambar 2. 9 Value JSON

String adalah kumpulan dari nol atau lebih karakter Unicode, yang dibungkus dengan tanda kutip ganda. Di dalam string dapat digunakan backslash escapes "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada string. String sangat mirip dengan string C atau Java.



Gambar 2. 10 String JSON

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



Gambar 2. 11 Number JSON

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

2.2.10. Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun

dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "Tulis sekali, jalankan di mana pun". Java merupakan bahasa pemrograman yang paling populer digunakan, dan secara luas dimanfaatkan dalam membangun berbagai jenis perangkat lunak aplikasi ataupun aplikasi berbasis *web*. Java digunakan untuk membuat aplikasi sistem rekomendasi ukm yang sedang dibangun [10].

Gambar 2. 12 Logo Java

2.2.11. 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 system 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 *operating system*. Keunggulan dari API ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya untuk saling berinteraksi.

Keuntungan dengan menggunakan API adalah sebagai berikut:

1. Portabilitas.

   Developer yang menggunakan API dapat menjalankan programnya dalam sistem operasi mana saja asalkan sudah terinstal API tersebut.

2. Lebih Mudah Dimengerti

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

Cara menggunakan API :

1. Dilakukan dengan mengimpor *package*/kelas.
2. Ada beberapa kelas bernama sama dipackage yang berbeda, yaitu:
   a. Import salah satu dan gunakan nama lengkap untuk yang lain.
   b. Gunakan nama lengkap semua kelas

Kebanyakan Sistem Operasi seperti Windows, menyediakan fasilitas API sehingga *programmer* dapat melakukan aktivitas programming dengan lebih konsisten. Meskipun API didesain untuk programer, namun API juga baik untuk user karena setidaknya dapat menjamin bahwa program tersebut memiliki interface yang sama, sehingga lebih mudah untuk dipelajari

2.2.12. Firebase

Adalah Backend as a Service (BaaS) yang saat ini dimiliki oleh Google. Firebase merupakaan solusi yang ditawarkan oleh Google untuk mempermudah pengembangan aplikasi mobile. Dua fitur menarik dari Firebase adalah Firebase Remote Config dan Firebase Real Time Database. Selain itu juga terdapat fitur pendukung untuk aplikasi yang memerlukan push notification yaitu Firebase Notification Console. Firebase Database merupakan penyimpanan basis data nonSQL yang memungkinan untuk menyimpan beberapa tipe data. Tipe data itu antara lain String, Long, dan Boolean. Data pada Firebase Database disimpan sebagai objek JSON tree. Tidak seperti basis

data SQL, tidak ada tabel dan baris pada basis data non-SQL. Ketika ada penambahan data, data tersebut akan menjadi node pada struktur JSON. Node merupakaan simpul yang berisi data dan bisa memiliki cabang-cabang berupa node lainnya yang berisi data pula. Proses pengisian suatu data ke Firebase Database dikenal dengan istilah push. Selain Firebase Database, Firebase menyediakan beberapa layanan lainnya yang juga dimanfaatkan dalam pengembangan aplikasi ini. Layanan tersebut antara lain Firebase Authentication, Storage, dan *Cloud* Messaging. Pada pengembangan aplikasi, layanan lainnya yang digunakan pada pengembangan aplikasi adalah Firebase Storage. Layaknya sebuah penyimpanan awan, Firebase Storage memungkin pengembang untuk mengunggah atau mengunduh sebuah berkas. Pada pengembangan aplikasi [11].

2.2.13. UML *(Unified Modelling Language)*

Unified Modelling Language (UML) adalah bahasa grafis untuk mendokumentasi, menspesifikasikan, dan membangun sistem perangkat lunak. UML berorientasi objek, menerapkan banyak level abstraksi, tidak bergantung proses pengembangan, tidak bergantung bahasa dan teknologi, pemaduan beberapa notasi di beragam metodologi, usaha bersama dari banyak pihak, didukung oleh kakas-kakas yang diintegrasikan lewat XML (XMI). Standar UML dikelola oleh OMG (*Object Management Group*).

UML adalah bahasa pemodelan untuk menspesifikasikan, memvisualisasikan, membangun dan mendokumentasikan artifak-artifak dari sistem. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya yaitu : *Grady Booch* OOD (*Object Oriented Design*), James Rumbaugh OMT (*Object Modeling Technique*) dan Ivar Jacobson OOSE (*Object Oriented Software Engineering*) [12].

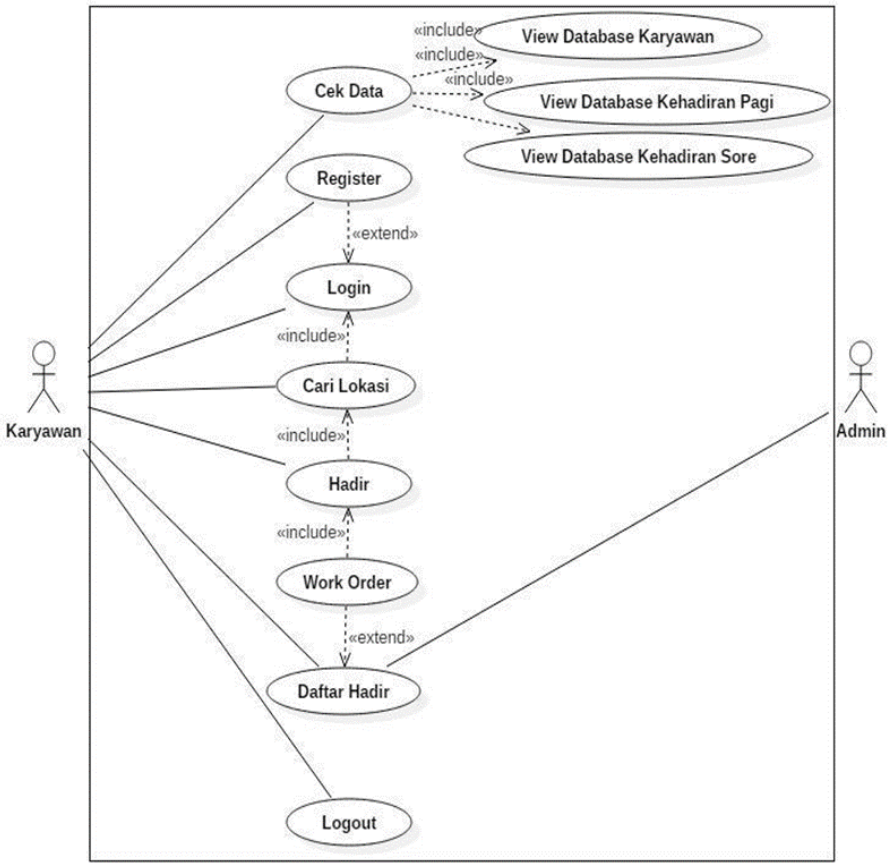
2.2.13.1. Use Case Diagram

Use case Diagram merupakan salah satu diagram untuk memodelkan aspek perilakau sistem. Masing-masing diagram *use case* menunjukkan sekumpulan *use case*, aktor, dan hubungannya. Diagram *use case* adalah penting untuk memvisualisasikan, menspesifikasikan, dan mendokumentasikan kebutuhan perilaku sistem. Diagram-diagram *use case* merupakan pusat pemodelan perilaku sistem,

subsistem, dan kelas. Diagram *use case* digunakan untuk mendeskripsikan apa yang seharusnya dilakukan oleh sistem. Ada empat hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan use case :

1. Sistem yaitu sesuatu yang hendak kita bangun.
2. Relasi adalah relasi antara aktor dengan *use case*.
3. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
4. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

   Berikut Contoh *Use Case* Diagram dapat dilihat pada Gambar 2.12.



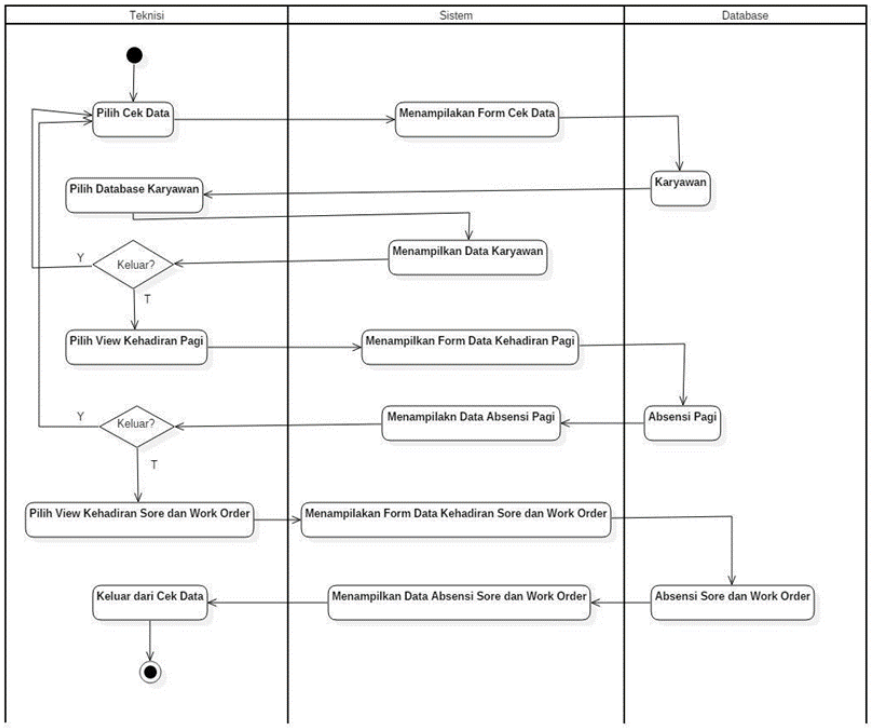
Gambar 2. 13 Contoh *Use Case* Diagram

2.2.13.2. Activity Diagram

Diagram aktivitas adalah diagram *flowchart* yang diperluas yang menunjukan aliran kendali satu aktivitas ke aktivitas lain di sistem. Diagram aktivitas ini digunakan untuk memodelkan aspek dinamis sistem. Diagram aktivitas mendeskripsikan aksi-aksi dan hasilnya. Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
3. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

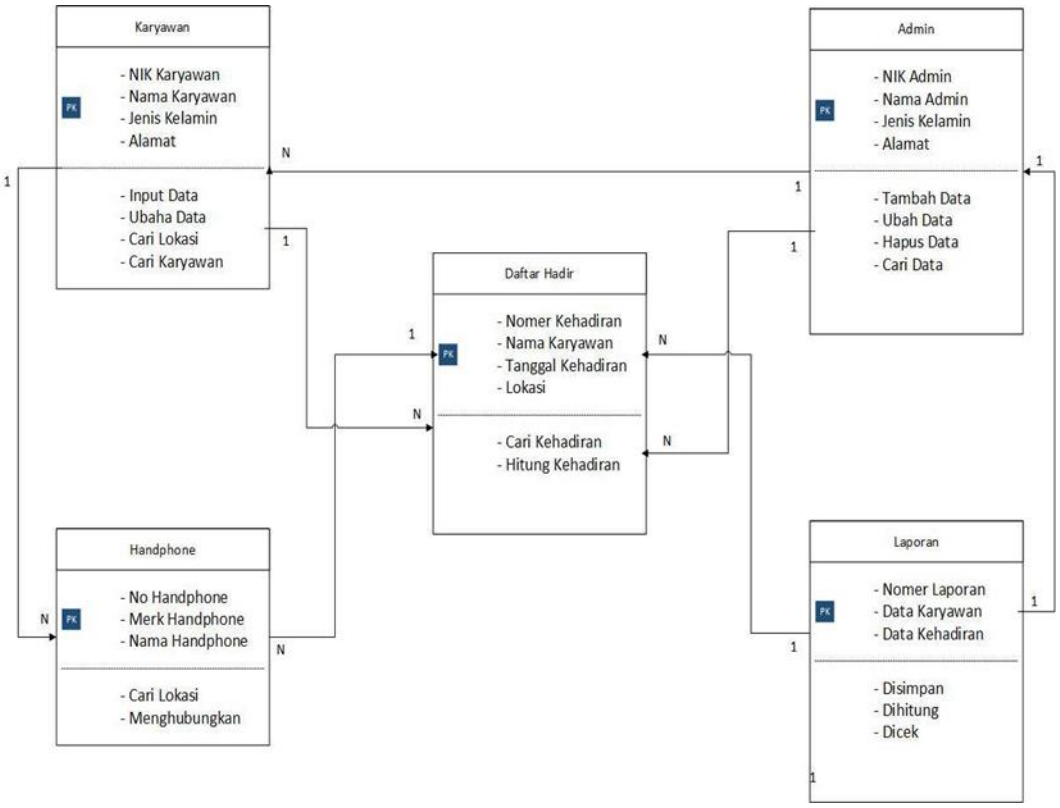
Berikut Contoh *Activity* Diagram dapat dilihat pada Gambar 2.13.



Gambar 2. 14 Contoh *Activity* Diagram

2.2.13.3. Class Diagram

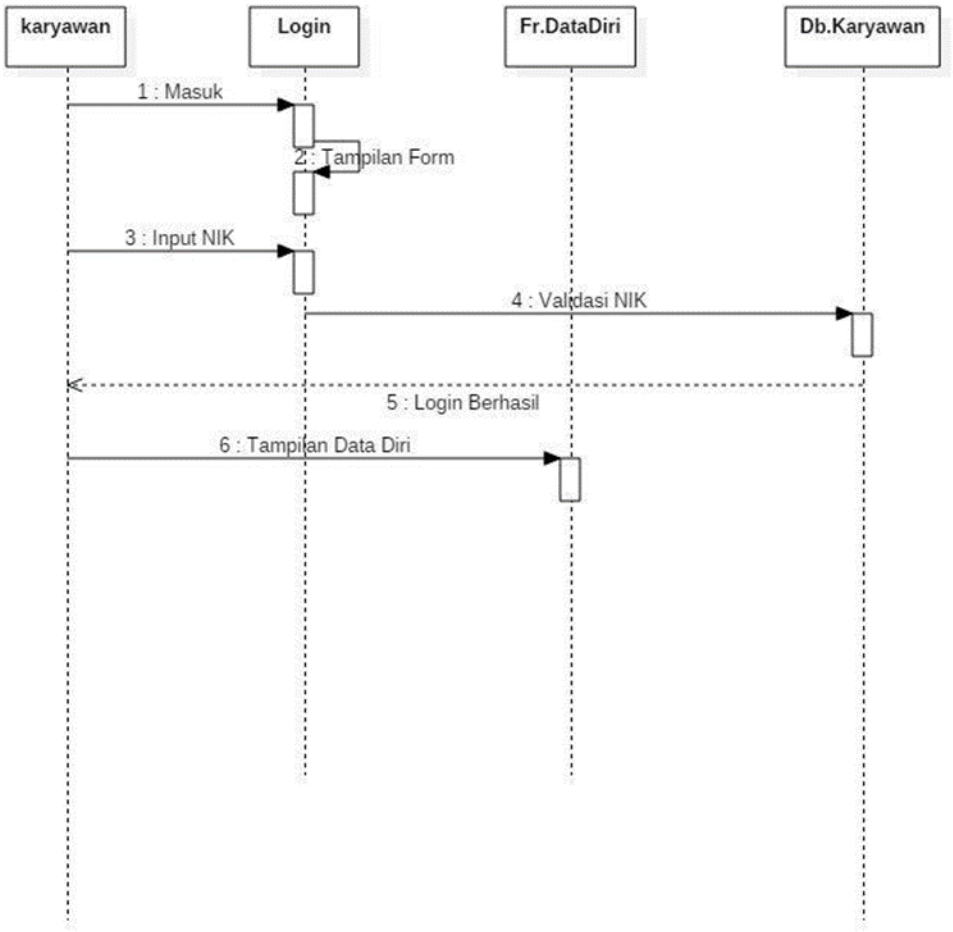
Class diagram merupakan diagram yang selalu ada di pemodelan system berorientasi objek. *Class* diagram menunjukkan hubungan antar class dalam system yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai satu tujuan. Kelas pada kelas diagram terdiri dari 3 bagian utama yaitu nama kelas, isi *property* dari kelas beserta metode yang ada pada kelas tersebut. Kelas juga memiliki jenis-jenis hubungan seperti asosiatif, dependensi, agregasi, komposisi, spesifikasi dan generalisasi. Hubungan ini digunakan untuk menggambarkan bagaimana hubungan dan interaksi yang terjadi antar kelas. Masing-masing komponen penyusun kelas memiliki hak akses seperti *public*, *private* dan *protected*. Berikut contoh *Class* Diagram dapat dilihat pada Gambar 2.14.



Gambar 2. 15 Contoh Class Diagram

2.2.13.4. Sequence Diagram

Sequence Diagram menunjukan interaksi yang terjadi antar objek. Diagram ini merupakan pandangan dinamis terhadap sistem. Diagram ini menekankan pada sisi basis keberurutan waktu dari pesan-pesan yang terjadi. Diagram sekuen menunjukan objek sebagai garis vertikal dan tiap kejadian sebagai panah horisontal dari objek pengirim ke objek penerima. Waktu berlalu dari atas ke bawah dengan lama waktu tidak relevan. Diagram ini hanya menunjukan barisan kejadian, bukan perwaktuan nyata. Kecuali untuk sistem waktu nyata yang mengharuskan konstrain barisan terjadi. Berikut Contoh *sequence* diagram dapat dilihat pada gambar 2.15.



Gambar 2. 16 Contoh Sequence Diagram

2.2.14. Google Maps

Seperti yang tercatat oleh Svennerberg (Beginning Google Maps API 3, p1), Google Maps API yang paling populer di internet. Pencatatan yang dilakukan pada bulan Mei 2010 ini menyatakan bahwa 43% mashup (aplikasi dan situs web yang menggabungkan dua atau lebih sumber data) menggunakan Google Maps API .Beberapa tujuan dari penggunaan Google Maps API adalah untuk melihat lokasi, mencari alamat, mendapatkan petunjuk mengemudi dan lain sebagainya. Hampir semua hal yang berhubungan dengan peta dapat memanfaatkan Google Maps.

Google Maps diperkenalkan pada Februari 2005 dan merupakan revolusi bagaimana peta di dalam web, yaitu dengan membiarkan user untuk menarik peta sehingga dapat menavigasinya. Solusi peta ini pada saat itu masih baru dan membutuhkan server khusus. Beberapa saat setelahnya, ada yang berhasil menhack Google Maps untuk digunakan di dalam webnya sendiri. Hal ini membuat Google Maps mengambil kesimpulan bahwa mereka membutuhkan API dan pada Juni 2005, Google Maps API dirilis secara publik [13].

2.2.15. Skala Likert

Skala likert adalah suatu skala psikometrik yang digunakan dalam kuesioner dan merupakan salah satu teknik yang dapat digunakan dalam evaluasi suatu program atau kebijakan perencanaan. Rensis Likert telah mengembangkan sebuah skala untuk mengukur sikap masyarakat di tahun 1932 yang sekarang terkenal dengan nama skala Likert. Skala Likert ini merupakan skala yang dapat dipergunakan untuk mengukur sikap, pendapat, dan persepsi seseorang atau sekelompok orang mengenai suatu gejala atau fenomena [14].

Dalam skala Likert terdapat dua bentuk pernyataan yaitu pernyataan positif yang berfungsi untuk mengukur sikap positif, dan pernyataan negative yang berfungsi untuk mengukur sikap negative objek. Skor pernyataan positif dimulai dari 1 untuk sangat tidak setuju (TS), 2 untuk tidak setuju (KS), 3 untuk raguragu (CS), 4 untuk setuju (S), dan 5 untuk sangat setuju (SS). Skor pernyataan negative dimulai dari 1 untuk sangat setuju (SS), 2 untuk setuju (S), 3 untuk raguragu (CS), 4 untuk tidak setuju (KS), dan 5 untuk sangat tidak setuju (TS). Beberapa menghilangkan option “Ragu-ragu” dalam instrument untuk memudahkan dalam melihat angket yang responden isikan. Skala Likert digunakan untuk mengukur kesetujuan dan ketidaksetujuan seseorang terhadap sesuatu rencana program, pelaksanaan program ataupun tingkat keberhasilan suatu program.

Kekurangan Skala Likert :

1. Karena ukuran yang digunakan adalah ukuran ordinal, skala Likert hanya dapat mengurutkan individu dalam skala, tetapi tidak dapat membandingkan berapa kali satu individu lebih baik dari individu yang lain.
2. Kadangkala total skor dari individu tidak memberikan arti yang jelas, karena banyak pola respons terhadap beberapa item akan memberikan skor yang sama. Adanya kelemahan di atas sebenarnya dapat dipikirkan sebagain error dari respons yang terjadi.

Kelebihan Skala Likert :

1. Mudah dibuat dan di terapkan.
2. Skala Likert lebih mudah membuatnya dibanding lain sepertti skala Thurstone.
3. Terdapat kebebasan dalam memasukan pertanyaan- pertanyaan, asalkan sesuai dengan konteks permasalahan yang diteliti.
4. Jawaban suatu item dapat berupa alternative, sehingga informasi mengenai item tersebut diperjelas.

Reliabilitas pengukuran bisa diperoleh dengan jumlah item tersebut diperjelas.