BAB 2
LANDASAN TEORI

2.1 Aplikasi

Aplikasi adalah sebuah program atau perangkat lunak yang dirancang untuk memenuhi tujuan tertentu, menurut *English Oxford Living Dictionary*. Sedangkan menurut Jogiyanto [10] dalam penelitian Lingga Agitya [7] aplikasi merupakan penerapan, menyimpan, sesuatu hal, data permasalahan. Pekerjaan ke dalam suatu sarana atau media yang dapat digunakan untuk menerapkan atau mengimplementasikan hal atau permasalahan yang ada sehingga berubah menjadi suatu bentuk yang baru tanpa menghilangkan nilai-nilai dasar dari hal data, permasalahan, pekerjaan itu sendiri.

Dari pengertian diatas peneliti dapat menyimpulkan bahwa aplikasi adalah suatu program atau perangkat lunak yang dibuat untuk tujuan tertentu untuk dapat menyelesaikan suatu masalah dalam lingkup tertentu yang dihadapi baik itu dibuat dan dikembangkan dengan tujuan untuk melakukan tugas yang bersifat umum atau juga dapat dikembangkan untuk melakukan tugas khusus. Hal ini didasarkan dari aplikasi dengan paket-paket tertentu yang unik sehingga tidak ada paket-paket program yang sesuai yang dapat digunakan. Oleh karena itu pengembangan aplikasi itu sendiri perlu dilakukan.

2.2 Monitoring

Monitoring adalah proses menjaga atau pengawasan terhadap keberadaan dan besarnya perubahan keadaan dan arus data dalam sebuah sistem. *Monitoring* bertujuan untuk mengidentifikasi kesalahan dan kemajuan dalam menentukan keputusan selanjutnya. Teknik yang digunakan dalam *monitoring* informasi sistem memotong bidang pengolahan real-time, statistik, dan analisis data. Satu set Komponen perangkat lunak yang digunakan untuk pengumpulan data, pengolahan, dan presentasi disebut sistem *monitoring* [11].

2.3 Konten Negatif

Menurut Kamus Besar Bahasa Indonesia secara arti kata konten berarti informasi yang tersedia melalui media atau produk elektronik, sedangkan negatif berarti tidak pasti atau kurang pasti, tidak baik atau kurang baik [12]. Jadi, dapat diambil kesimpulan bahwa konten negatif adalah suatu informasi yang terdapat di pada media elektronik baik itu TV, Radio, *Smartphone*, atau dalam sebuah halaman web yang mengandung unsur-unsur yang menyimpang atau kurang baik untuk disajikan kepada masyarakat umum yang biasanya bermuatan kekerasan dan pornografi.

Menurut Kementerian Komunikasi dan Informatika RI (KOMINFO) sejak bulan April 2015, Menteri Komunikasi dan Informatika melalui SK Menkominfo Nomor 290 Tahun 2015 membentuk Forum Penanganan Situs Internet Bermuatan Negatif yang terdiri atas 4 (empat) Panel, yaitu:

1. Panel Pornografi, Kekerasan Pada Anak, dan Keamanan Internet.
2. Panel Terorisme, SARA, dan Kebencian.
3. Panel Investasi Ilegal, Penipuan, Perjudian, Obat & Makanan, dan Narkoba.
4. Panel Hak Kekayaan Intelektual.

Dalam penelitian ini peneliti lebih memfokuskan konten negatif pada perangkat *smartphone* Android berupa gambar yang mengandung unsur *pornografi*.

2.4 Perangkat Pintar

Perangkat pintar adalah sebuah alat elektronik interaktif yang mengerti perintah sederhana yang dikirim oleh pengguna dan membantu dalam aktivitas sehari-hari. Beberapa perangkat pintar yang paling umum digunakan adalah smartphone, tablet, phablets, smartwatch, kacamata pintar dan peralatan elektronik pribadi lainnya. Sementara banyak perangkat pintar kecil, elektronik pribadi portabel, perangkat tersebut sebenarnya ditentukan oleh kemampuan mereka untuk terhubung ke jaringan untuk berbagi dan berinteraksi dari jarak jauh. Banyak TV dan kulkas juga dianggap perangkat cerdas [13].

Dalam penelitian ini peneliti menggunakan perangkat pintar berupa ponsel pintar atau *smartphone* dengan sistem operasi Android yang digunakan sebagai media dalam penelitian.

2.5 Smartphone

Menurut Williams dan Sawyer [14] *smartphone* adalah telepon seluler dengan mikroprosesor, memori, layar tampilan, dan modem *built-in*. *Smartphone* menggabungkan fungsionalitas PDA atau (*Pocket* PC) dengan telepon dalam satu alat yang sama sehingga menghasilkan *gadget* yang memiliki banyak fitur dan fungsionalitas, dimana terdapat pesan teks, kamera, pemutar musik, video, *game*, akses email, tv digital, *search engine*, pengelola informasi pribadi, fitur GPS, jasa telepon internet, dan bahkan terdapat telepon yang juga berfungsi sebagai kartu kredit.

Dalam penelitian ini, peneliti menggunakan *smartphone* dengan sistem operasi Android sebagai alat untuk dapat melakukan *monitoring* dengan mengakses perangkat keras melalui *Application Programming Interface (API)* guna mendapatkan data dan hasil yang diinginkan.

2.6 Android

Menurut Nazaruddin Safaat H [15] 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 Google Inc. membeli Android Inc. yang merupakan pendatang baru yang membuat peranti lunak untuk ponsel/*smartphone.* Kemudian untuk mengembangkan android dibentuklah *Open Handset Aliance,* 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 *open source* pada perangkat *mobile*. Di lain pihak, Google merilis kode-kode Android dibawah lisensi Apache, sebuah lisensi perangkat lunak dan *open platform* perangkat seluler.

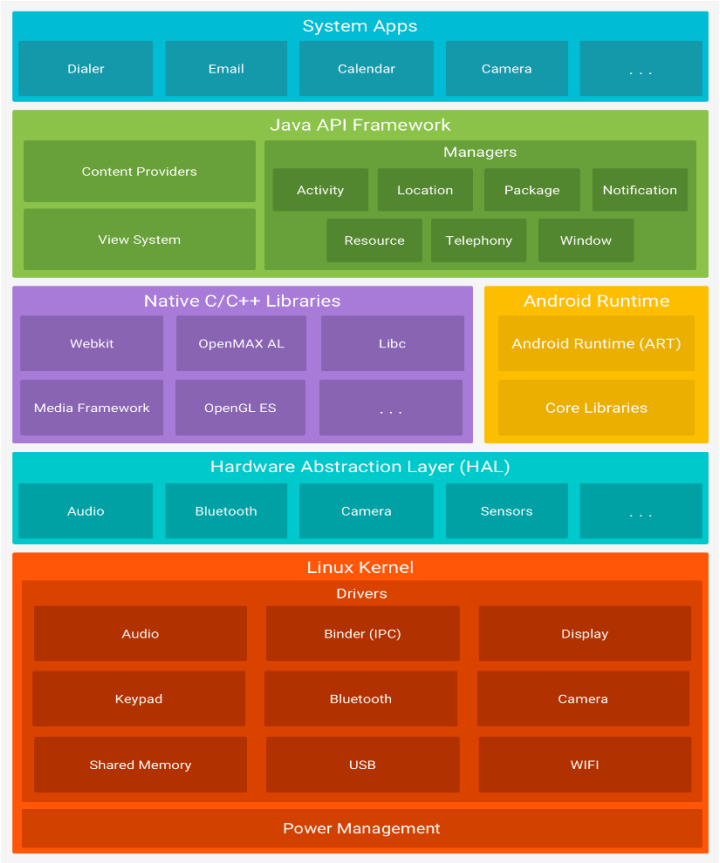
Hingga awal menjelang penghujung tahun 2017 terdapat beberapa versi Android yang tersedia untuk digunakan, diantaranya adalah sebagai berikut:

Tabel 2.1 Rilis Versi Android

Nama Kode	Versi	API level
Oreo	8.0	Api level 26
Nougat	7.1	API level 25
Nougat	7.0	API level 24
Marshmallow	6.0	API level 23
Lollipop	5.1	API level 22
Lollipop	5.0	API level 21
KitKat	4.4 - 4.4.4	API level 19
Jelly Bean	4.3.x	API level 18
Jelly Bean	4.2.x	API level 17
Jelly Bean	4.1.x	API level 16
Ice Cream Sandwich	4.0.3 - 4.0.4	API level 15, NDK 8
Ice Cream Sandwich	4.0.1 - 4.0.2	API level 14, NDK 7
Honeycomb	3.2.x	API level 13
Honeycomb	3.1	API level 12, NDK 6
Honeycomb	3.0	API level 11
Gingerbread	2.3.3 - 2.3.7	API level 10
Gingerbread	2.3 - 2.3.2	API level 9, NDK 5
Froyo	2.2.x	API level 8, NDK 4
Eclair	2.1	API level 7, NDK 3
Eclair	2.0.1	API level 6
Eclair	2.0	API level 5
Donut	1.6	API level 4, NDK 2
Cupcake	1.5	API level 3, NDK 1
(Tidak ada nama kode)	1.1	API level 2
(Tidak ada nama kode)	1.0	API level 1

2.7 Arsitektur Android

Secara garis besar arsitektur android terdiri dari komponen utama yang tersusun seperti pada Gambar 2.



Sumber Gambar : https://developer.android.com/guide/platform/index.html?hl=id

Gambar 2.1 Arsitektur Android

1. *Linux Kernel*

Fondasi platform Android adalah kernel Linux. Sebagai contoh, Android *Runtime* (ART) bergantung pada kernel Linux untuk fungsionalitas dasar seperti threading dan manajemen memori tingkat rendah.

Menggunakan kernel Linux memungkinkan Android untuk memanfaatkan fitur keamanan inti dan memungkinkan produsen perangkat untuk mengembangkan *driver* perangkat keras untuk kernel yang cukup dikenal [16].

2. *Hardware Abstraction Layer (HAL)*

Hardware Abstraction Layer (HAL) menyediakan antarmuka standar yang mengekspos kemampuan perangkat keras di perangkat ke kerangka kerja Java API yang lebih tinggi. HAL terdiri atas beberapa modul pustaka, masing-masing mengimplementasikan antarmuka untuk komponen perangkat keras tertentu, seperti modul kamera atau bluetooth. Bila API kerangka kerja melakukan panggilan untuk mengakses perangkat keras, sistem Android memuat modul pustaka untuk komponen perangkat keras tersebut [16].

3. Android Runtime

Untuk perangkat yang menjalankan Android versi 5.0 (API level 21) atau yang lebih tinggi, setiap aplikasi menjalankan proses masing-masing dengan tahap Android *Runtime* (ART). ART ditulis guna menjalankan beberapa mesin virtual pada perangkat bermemori rendah dengan mengeksekusi file DEX, format *bytecode* yang didesain khusus untuk Android yang dioptimalkan untuk *footprint* memori minimal. Buat rantai aplikasi, misalnya Jack, mengumpulkan sumber Java ke bytecode DEX, yang dapat berjalan pada platform Android.
Beberapa fitur utama ART mencakup:

1. Kompilasi mendahului waktu (AOT) dan tepat waktu (JIT).
2. Pengumpulan sampah (GC) yang dioptimalkan.
3. Dukungan debug yang lebih baik, mencakup profiler sampling terpisah, pengecualian diagnostik mendetail dan laporan kerusakan dan kemampuan untuk mengatur titik pantau guna memantau bidang tertentu.

Sebelum ke Android versi 5.0 (API level 21), Dalvik adalah waktu proses. Android. Jika aplikasi Anda berjalan baik pada ART, semestinya berfungsi baik juga pada Dalvik, tetapi mungkin tidak sebaliknya. Android juga menyertakan serangkaian pustaka waktu proses inti yang menyediakan sebagian besar fungsionalitas bahasa pemrograman Java, termasuk beberapa fitur bahasa Java 8, yang digunakan kerangka kerja Java API [16].

4. *Native C/C++ Library*

Banyak komponen dan layanan sistem Android inti seperti ART dan HAL dibuat dari kode asli yang memerlukan pustaka asli yang tertulis dalam C dan C++. Platform Android memungkinkan kerangka kerja Java API mengekspos fungsionalitas beberapa pustaka asli pada aplikasi. Misalnya, Anda bisa mengakses OpenGL ES melalui kerangka kerja Java OpenGL API Android guna menambahkan dukungan untuk menggambar dan memanipulasi grafik 2D dan 3D pada aplikasi Anda.

Jika Anda mengembangkan aplikasi yang memerlukan kode C atau C++, Anda bisa menggunakan Android NDK untuk mengakses beberapa pustaka *platform* asli langsung dari kode asli [16].

5. *Java API Framework*

Keseluruhan rangkaian fitur pada Android OS tersedia untuk Anda melalui API yang ditulis dalam bahasa Java. API ini membentuk elemen dasar yang Anda perlukan untuk membuat aplikasi Android dengan menyederhanakan penggunaan kembali inti, komponen dan layanan sistem modular, yang menyertakan berikut ini:

1. Tampilan Sistem yang kaya dan luas bisa Anda gunakan untuk membuat UI aplikasi, termasuk daftar, kisi, kotak teks, tombol, dan bahkan *browser* web yang dapat disematkan.
2. Pengelola Sumber Daya, memberikan akses ke sumber daya bukan kode seperti string yang dilokalkan, grafik, dan file *layout*.
3. Pengelola Notifikasi yang mengaktifkan semua aplikasi guna menampilkan lansiran khusus pada bilah status.
4. Pengelola Aktivitas yang mengelola daur hidup aplikasi dan memberikan back-stack navigasi yang umum.
5. Penyedia Materi yang memungkinkan aplikasi mengakses data dari aplikasi lainnya, seperti aplikasi Kontak, atau untuk berbagi data milik sendiri.

Developer memiliki akses penuh ke API kerangka kerja yang sama dengan yang digunakan oleh aplikasi sistem Android [16].

6. Systen Apps

Android dilengkapi dengan serangkaian aplikasi inti untuk email, perpesanan SMS, kalender, menjelajahi internet, kontak, dll. Aplikasi yang disertakan bersama platform tidak memiliki status khusus pada aplikasi yang ingin dipasang pengguna. Jadi, aplikasi pihak ketiga dapat menjadi browser web utama, pengolah pesan SMS atau bahkan keyboard utama (beberapa pengecualian berlaku, seperti aplikasi *Settings* sistem).

Aplikasi sistem berfungsi sebagai aplikasi untuk pengguna dan memberikan kemampuan kunci yang dapat diakses oleh *developer* dari aplikasi mereka sendiri. Misalnya, jika aplikasi Anda ingin mengirimkan pesan SMS, Anda tidak perlu membangun fungsionalitas tersebut sendiri-sebagai gantinya Anda bisa menjalankan aplikasi SMS mana saja yang telah dipasang guna mengirimkan pesan kepada penerima yang Anda tetapkan [16].

2.8 Komponen Aplikasi Android

Aplikasi android ditulis dalam bahasa pemrograman java. kode java dikompilasi bersama dengan data file yang dibutuhkan oleh aplikasi dimana prosesnya dikemas oleh *tools* yang dinamakan "apt *tools*" ke dalam paket android sehingga menghasilkan file dengan ekstensi apk. file apk itulah yang kita sebut dengan aplikasi dan nantinya dapat dipasang pada perangkat mobile [15].
terdapat empat jenis komponen pada aplikasi android yaitu:

1. Activity

Activity akan menyajikan *user interface* (ui) kepada pengguna sehingga pengguna dapat melakukan interaksi. Sebuah aplikasi android bisa jadi hanya memiliki satu *activity*, tetapi umumnya aplikasi memiliki banyak *activity* yang bergantung pada tujuan aplikasi dan desain aplikasi itu sendiri. Untuk berpindah dari satu *activity* ke *activity* lain dapat dilakukan menggunakan sebuah *trigger* seperti klik tombol pada tampilan aplikasi.

2. Service

Service tidak memiliki *graphic user interface* (gui), tetapi *service* berjalan secara *background*. Sebagai contoh dalam memutar musik, *service* mungkin

memutar musik atau mengambil data dari jaringan, tetapi setiap *service* harus berada dalam kelas induknya. Apabila sebuah pemutar musik sedang memutar lagu dari list yang ada, aplikasi akan memiliki dua atau lebih *activity* yang memungkinkan pengguna untuk memutar sambil memilih lagu baru. Untuk menjaga musik tetap berjalan sebuah *activity* dapat menjalankan *service*.

3. Broadcast Receiver

Broadcast receiver berfungsi menerima dan bereaksi untuk menyampaikan notifikasi. contoh *broadcast* seperti notifikasi zona waktu telah berubah, baterai lemah, gambar berhasil diambil oleh kamera, dan lain-lain. Aplikasi juga dapat meng-inisialisasi *broadcast* misalnya memberikan informasi pada aplikasi lain bahwa ada data yang telah diunduh ke perangkat dan siap untuk digunakan.

4. Content Provider

Content provider membuat kumpulan aplikasi data secara spesifik sehingga bisa digunakan oleh aplikasi lain. Data disimpan dalam file sistem seperti database sqlite. content provider menyediakan cara untuk mengakses data yang dibutuhkan oleh suatu *activity*.

2.9 Android SDK(Software Development Kit)

Android SDK (*Software Development Kit*) adalah *tools* API (*Application Programming Interface*) yang diperlukan untuk mulai mengembangkan aplikasi pada *platform* Android. Android SDK disediakan sebagai alat bantu untuk mulai mengembangkan aplikasi pada *platform* Android menggunakan bahasa pemrograman Java. Sebagai *platform* aplikasi-netral, Android memberi kesempatan untuk membuat aplikasi yang dibutuhkan [15].

Android SDK versi resmi dapat ditemukan dan diunduh di situs resmi Google. Google akan merilis SDK versi baru yang disesuaikan ketika Android versi terbaru juga. Untuk dapat mengembangkan aplikasi dengan fitur terbaru maka pengembang harus mengunduh Android SDK dengan versi terbaru pula yang disesuaikan dengan target *device* masing-masing.

2.10 Android Studio

Android Studio adalah Lingkungan Pengembangan Terpadu *Integrated Development Environment* (IDE) untuk pengembangan aplikasi Android, berdasarkan IntelliJ IDEA . Selain merupakan *editor* kode IntelliJ dan alat pengembang yang berdaya guna, Android Studio menawarkan fitur lebih banyak untuk meningkatkan produktivitas Anda saat membuat aplikasi Android, misalnya:

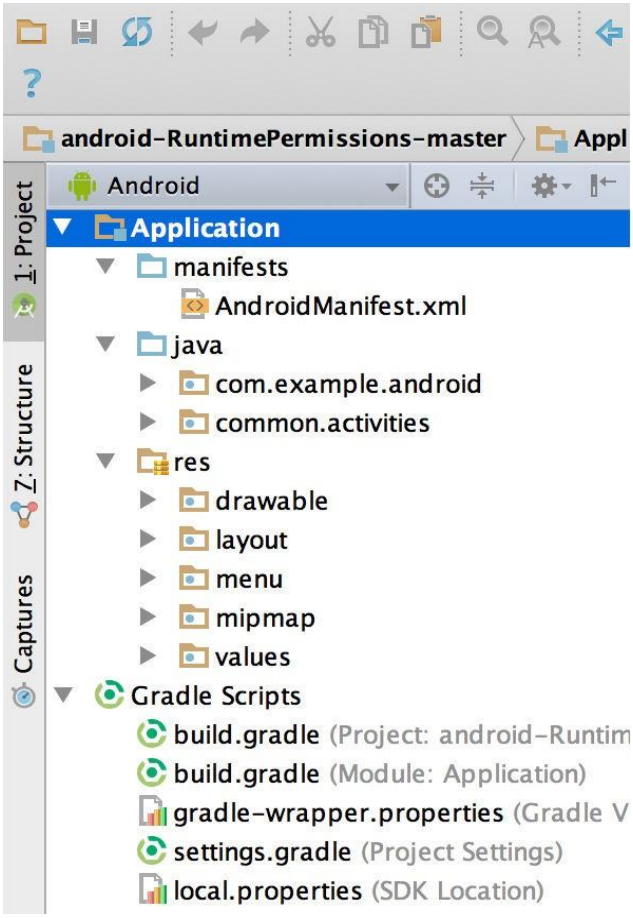
1. Sistem versi berbasis Gradle yang fleksibel.
2. *Emulator* yang cepat dan kaya fitur.
3. Lingkungan yang menyatu untuk pengembangan bagi semua perangkat Android.
4. *Instant Run* untuk mendorong perubahan ke aplikasi yang berjalan tanpa membuat APK baru.
5. Template kode dan integrasi GitHub untuk membuat fitur aplikasi yang sama dan mengimpor kode contoh.
6. Alat pengujian dan kerangka kerja yang ekstensif.
7. Alat Lint untuk meningkatkan kinerja, kegunaan, kompatibilitas versi, dan masalah-masalah lain.
8. Dukungan C++ dan NDK.
9. Dukungan bawaan untuk Google *Cloud Platform*, mempermudah pengintegrasian Google *Cloud Messaging* dan *App Engine*.

Setiap proyek di Android Studio berisi satu atau beberapa modul dengan file kode sumber dan file sumber daya. Jenis-jenis modul mencakup:

1. Modul aplikasi Android.
2. Modul Pustaka.
3. Modul Google *App Engine*.

Secara *default*, Android Studio akan menampilkan file proyek Anda dalam tampilan proyek Android, seperti yang ditampilkan dalam gambar 3. Tampilan

disusun berdasarkan modul untuk memberikan akses cepat ke *file* sumber utama proyek Anda.



Sumber Gambar : https://developer.android.com/studio/intro/index.html?hl=id

Gambar 2.2 File Proyek di Tampilan Android

Semua file versi terlihat di bagian atas di bawah Gradle *Scripts* dan masing-masing modul aplikasi berisi folder berikut:

1. *Manifests:* berisi *file* AndroidManifest.xml.
2. *java*: Berisi file kode sumber Java, termasuk kode pengujian JUnit.
3. *res*: Berisi semua sumber daya bukan kode, seperti tata letak XML, string UI, dan gambar bitmap.

Struktur proyek Android pada *disk* berbeda dari representasi rata ini. Untuk melihat struktur file sebenarnya dari proyek ini, pilih Project dari menu tarik turun Project (dalam *Gambar 3*, struktur ditampilkan sebagai Android) [17].

2.11 Bahasa Pemrograman Java

Java merupakan sebuah Bahasa pemrograman berorientasi objek yang dapat berjalan pada *platform* yang berbeda, baik di Windows, Linux, serta system operasi lainnya. Dengan menggunakan Java kita dapat mengembangkan banyak aplikasi yang dapat digunakan pada lingkungan yang berbeda, seprti pada *Desktop, Mobile, Internet* dan lain-lain.

Untuk menginstalasi dan menggunakan Java, Sun Micro System selaku pengembang Java menyediakan paket instalasi sesuai dengan kebutuhan kita dalam membangun suatu aplikasi. Berikut ini uraian singkat mengenai paket aplikasi Java yang tersedia.

1. J2ME (Java 2 *Micro Edition*)

   Paket instalasi ini dapat digunakan untuk membangun *software* yang berjalan pada perangkat yang memiliki memori dan sumber daya yang kecil, seperti pada *handphone,* PDA, dan *Smartcard.*

2. J2SE (Java 2 *Standard Edition*)

   Paket instalasi ini digunakan untuk mengembangkan aplikasi yang berjalan pada lingkungan *workstation,* seperti aplikasi desktop.

3. J2EE (Java 2 *Enterprise Edition*)

   Paket instalasi ini dapat digunakan untuk mengembangkan aplikasi pada lingkungan internet maupun aplikasi skala *enterprise.*

Java juga merupakan bahasa pemrograman resmi yang digunakan untuk pembangunan aplikasi android yang didukung penuh oleh Google. Namun meskipun demikian saat ini java bukanlah satu-satunya Bahasa yang dapat digunakan untuk membangun aplikasi Android seperti Xamarin dengan menggunakan Bahasa pemrograman C#, Cordova dengan menggunakan bahasa pemrograman web seperti HTML, CSS, dan Javascript dan lain-lain.

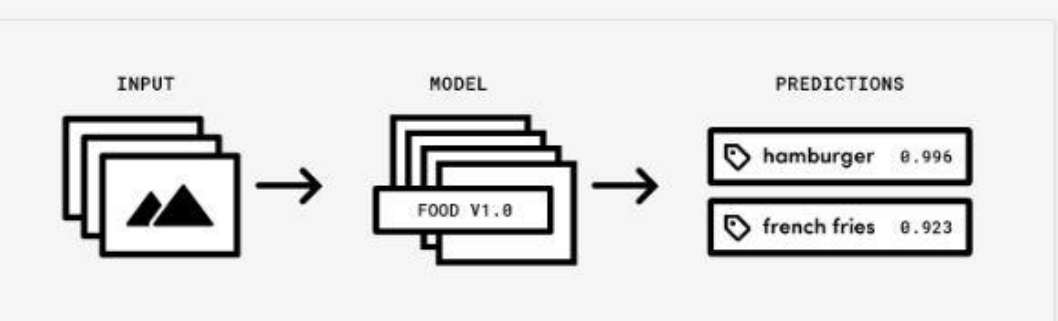
2.12 Clarifai

Clarifai adalah perusahaan kecerdasan buatan yang unggul dalam pengenalan visual(*Visual Recognition*), Clarifai didirikan pada tahun 2013 oleh Matthew Zeiler, seorang ahli terkemuka dalam *Machine Learning*, Clarifai telah menjadi pemimpin pasar sejak memenangkan lima tempat teratas dalam klasifikasi citra pada kompetisi ImageNet 2013 [8].

Clarifai adalah alat yang dapat mengidentifikasi atau mengenali gambar atau video yang dimasukkan sebagai inputan yang dapat memberikan hasil berupa prediksi tentang apa yang ada di dalam gambar atau video berupa besaran probabilitas kemungkinan nya [18].

API Clarifai dibangun dengan konsep ide dasar yang sederhana dimana anda memesukkan inputan berupa gambar atau video dan kemudian anda akan mendapatkan hasil dari prediksi yang diminta.

Setiap tipe dari prediksi adalah berdasar dari model apa yang dipilih untuk menjalankan prediksi dari inputan gambar atau video. Sebagai contoh apabila anda memilih model makanan untuk gambar masukkan anda maka Clarifai akan memprediksi gambar berdasarkan model makanan yang telah diketahui dan begitu seterusnya.



Sumber Gambar : https://www.clarifai.com/developer/guide/#getting-started

Gambar 2.3 Konsep kerja Clarifai

2.12.1 Model Clarifai

Gambar atau video yang menjadi *data* masukkan yang dijalankan ditandai oleh model dimana sebuah model adalah *classifier* yang terlatih dan dapat

mengenali apa yang terdapat di dalam gambar atau video sesuai dengan apa yang telah diketahui. Model satu dan yang lainnya digunakan untuk mengetahui hal-hal yang berbeda pula. Menjalankan model yang berbeda untuk mengenali suatu gambar atau video dapat memberikan hasil yang berbeda secara drastis.

Berikut adalah beberapa contoh dari model pada Clarifai :

1. *General Model*

   General model adalah model dimana berisi tag yang lebih umum dan berisi berbagai topik yang berbeda dan akan menghasilkan prediksi yang umum yang terdapat di gambar.

2. *Face Model*

   Face model memberikan hasil prediksi berupa besaran probabilitas yang terdeteksi berdasar model wajah yang diketahui dari gambar yang Anda masukkan

3. *Not Safe For Work* (NSFW) *model*

   NSFW model analisis berupa gambar atau video berdasar pada besaran skor probabilitas hasilpengembalian kemunginan bahwa gambar bermuatan pornografi.

   Selain yang telah dipaparkrn di atas masih ada model-model yang lain seperti *color, weding, travel, celebrity, demographic, logo, food, apparel, focus, moderation* yang masing-masing memiliki fungsi dan fokus yang berbeda.

2.12.2 Authentication

Otentikasi ke API ditangani melalui API *key*. Anda dapat membatasi cakupan API *key*, yang memungkinkan kunci untuk melakukan operasi yang sangat spesifik pada aplikasi yang diberikan, agar aplikasi Anda tetap aman.

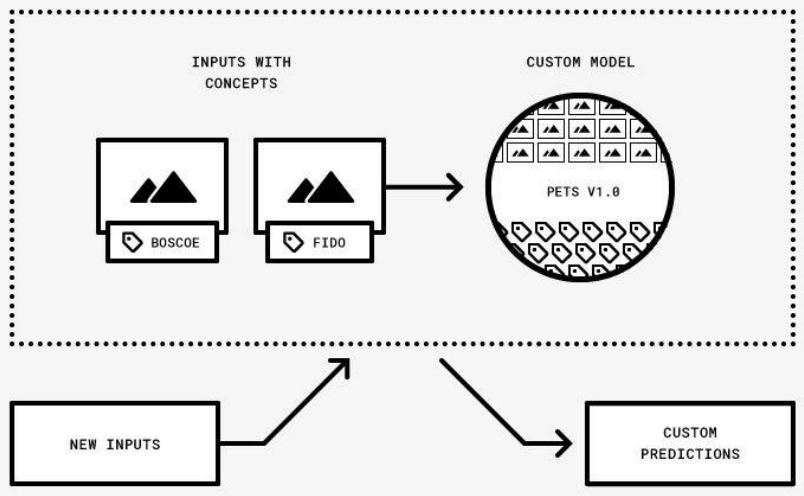
2.12.3 Predict

Predict, melakukan prediksi dan analisa gambar yang Anda masukkan dan memberi tahu Anda apa isi yang terkandung di gambar. API akan mengembalikan

daftar konsep dengan probabilitas yang sesuai dari seberapa besar kemungkinan konsep-konsep ini terkandung di dalam gambar. Bila Anda membuat prediksi melalui API, Anda memberi tahu model apa yang akan digunakan. Sebuah model berisi sekelompok konsep. Model hanya akan melihat konsep yang dikandungnya. Anda dapat menggunakan model yang berbeda untuk menganalisis gambar dengan berbagai cara.

2.12.4 Train

Train atau pelatihan memungkinkan Anda membuat model sendiri menggunakan konsep kustom Anda sendiri. Anda mulai dengan menambahkan masukan gambar yang sudah Anda ketahui berisi konsep yang Anda minati. Anda tidak memerlukan banyak gambar untuk memulai. Sebaiknya mulai dengan 10 dan tambahkan lebih banyak sesuai kebutuhan. Kemudian membuat model dan menentukan konsep apa yang dikandungnya. Setelah membuat model, Anda melatihnya berdasarkan gambar dan konsep yang Anda berikan. Operasi pelatihan ini bersifat asinkron. Mungkin perlu beberapa detik agar model Anda dilatih dan siap sepenuhnya. Setelah selesai latihan, Anda bisa menggunakan model itu untuk memprediksi konsep tersebut pada gambar baru.



Sumber Gambar : https://www.clarifai.com/developer/guide/train#train

Gambar 2.4 Train

Dalam penelitian ini peneliti menggunakan *Application Programming Interface* (API) Clarifai untuk melakukan identifikasi gambar dengan konten negatif yang telah dilakukan pelatihan sebelumnya.

2.13 Web Service

Web service adalah *standar* yang digunakan untuk pertukaran data antar aplikasi atau sistem. Standar diperlukan karena masing-masing aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda. Contoh implementasi dari *web service* antara lain adalah SOAP dan REST [19].

Peneliti menggunakan *web service* yang ditempatkan di sebuah server untuk dapat menjalankan sistem sesuai dengan yang diharapkan sehingga aplikasi dapat mengambil serta menyimpan data yang diperlukan seperti data pengguna, data *monitoring* dan lain sebagainya. Adapun metode yang digunakan untuk format pertukaran data adalah *Javascript Object Notation* (JSON).

2.14 Representational State Transfer (REST)

REST *(REpresentational State Transfer)* merupakan standar arsitektur komunikasi berbasis web yang sering diterapkan dalam pengembangan layanan berbasis web. Umumnya menggunakan HTTP *(Hypertext Transfer Protocol)* sebagai protocol untuk komunikasi *data*. REST pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000 [20].

Pada arsitektur REST, REST *server* menyediakan *resources* (sumber daya/data) dan REST *client* mengakses dan menampilkan *resource* tersebut untuk penggunaan selanjutnya. Setiap *resource* diidentifikasi oleh URIs (*Universal Resource Identifiers*) atau global ID. *Resource* tersebut direpresentasikan dalam bentuk format teks, JSON atau XML. Pada umumnya formatnya menggunakan JSON dan XML.

Keuntungan REST antara lain :

1. Bahasa dan *platform agnostic*.
2. Lebih sederhana/simpel untuk dikembangkan daripada SOAP.

3. Mudah dipelajari, tidak bergantung pada *tools*.
4. Ringkas, tidak membutuhkan *layer* pertukaran pesan (*messaging*) tambahan.
5. Secara desain dan filosofi lebih dekat dengan web.

Kelemahan REST antara lain :

1. Mengasumsi model *point-to-point* komunikasi tidak dapat digunakan untuk lingkungan komputasi terdistribusi di mana pesan akan melalui satu atau lebih perantara.
2. Kurangnya dukungan standar untuk keamanan, kebijakan, keandalan pesan, dll, sehingga layanan yang mempunyai persyaratan lebih canggih lebih sulit untuk dikembangkan ("dipecahkan sendiri").
3. Berkaitan dengan model *transport* HTTP*(Hypertext Transfer Protocol)*.

Berikut metode HTTP*(Hypertext Transfer Protocol)* yang umum digunakan dalam arsitektur berbasis REST :

1. *GET*, menyediakan hanya akses baca pada *resource*.
2. *PUT*, digunakan untuk menciptakan *resource* baru.
3. *DELETE*, digunakan untuk menghapus *resource*.
4. *POST*, digunakan untuk memperbarui *resource* yang ada atau membuat *resource* baru.
5. *OPTIONS*, digunakan untuk mendapatkan operasi yang disupport pada *resource*.

Web service yang berbasis arsitektur REST kemudian dikenal sebagai RESTful *web services*. Layanan web ini menggunakan metode HTTP*(Hypertext Transfer Protocol)* untuk menerapkan konsep arsitektur REST.

Cara Kerja RESTful web services yaitu :

Sebuah *client* mengirimkan sebuah *data* atau *request* melalui HTTP *Request* dan kemudian *server* merespon melalui HTTP *Response*. Komponen dari HTTP *request* :

1. *Verb*, HTTP *method* yang digunakan misalnya *GET, POST, DELETE, PUT* dll.
2. *Uniform Resource Identifier* (URI) untuk mengidentifikasikan lokasi *resource* pada *server*.
3. HTTP *Version*, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.
4. *Request Header*, berisi metadata untuk HTTP *Request*. Contoh, *type client/browser*, format yang didukung oleh *client*, format dari *body* pesan, seting *cache* dll.
5. *Request Body*, konten dari *data*.

Sedangkan komponen dari http *response* yaitu :

1. Status/*Response Code*, mengindikasikan status *server* terhadap *resource* yang *direquest*. misal : 404, artinya *resource* tidak ditemukan dan 200 *response OK*.
2. HTTP *Version*, menunjukkan versi dari HTTP yang digunakan, contoh HTTP v1.1.
3. *Response Header*, berisi *metadata* untuk HTTP Response. Contoh, *type server*, panjang *content*, tipe *content*, waktu *response*, dll
4. *Response Body*, konten dari data yang diberikan.

2.15 JavaScript Object Notation (JSON)

JavaScript 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, Standar ECMA-262 Edisi ke-3 - Desember 1999. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman 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 [21].

JSON terbuat dari dua struktur:

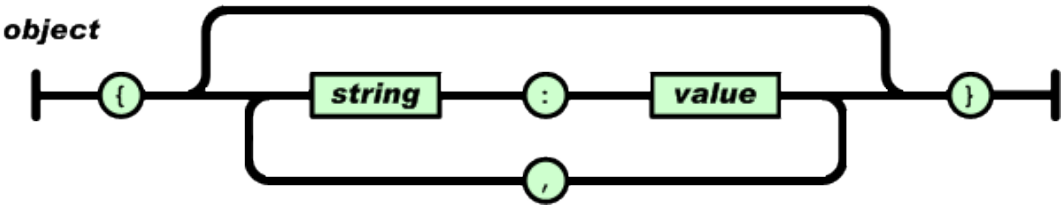
1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai terurutkan (*an ordered list of values*).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

JSON menggunakan bentuk sebagai berikut:

1. Objek

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).

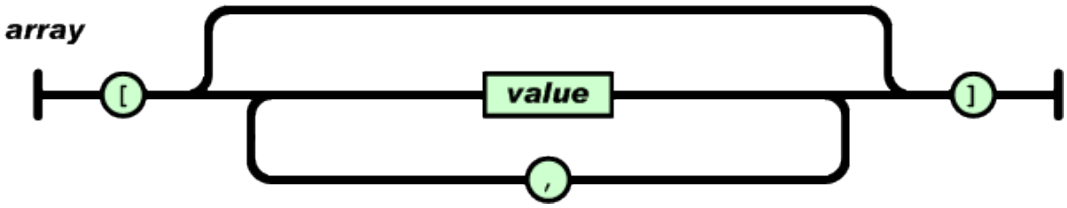


Sumber Gambar : http://www.json.org/json-id.html

Gambar 2.5 JSON Object

2. Larik

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

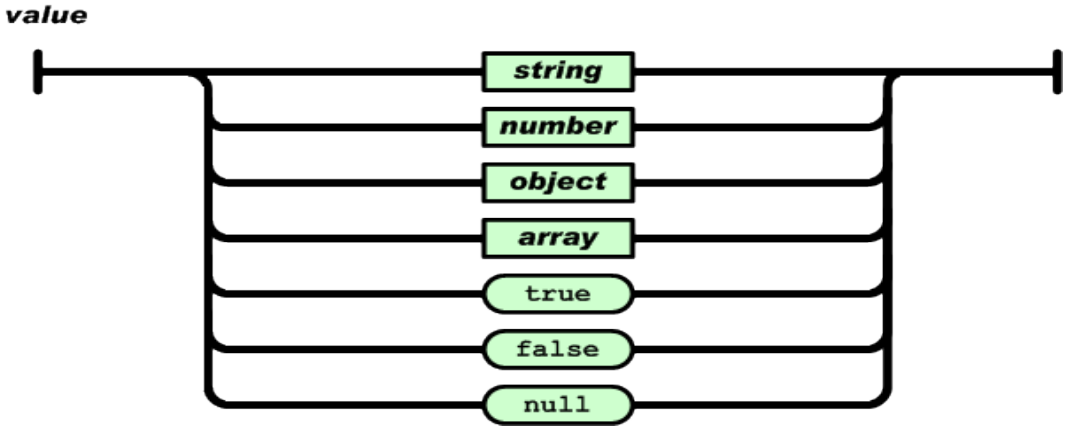


Sumber Gambar : http://www.json.org/json-id.html

Gambar 2.6 JSON Array

3. Nilai

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.

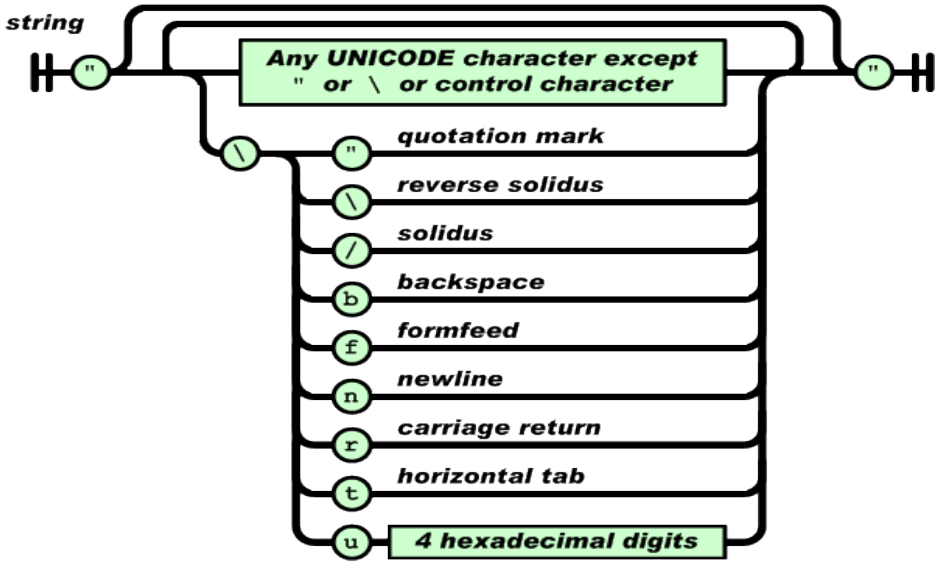


Sumber Gambar : http://www.json.org/json-id.html

Gambar 2.7 Value JSON

4. *String*

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.

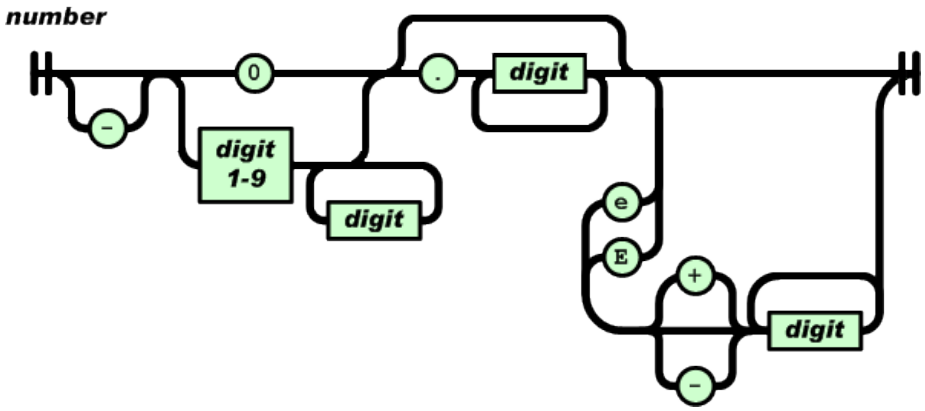


Sumber Gambar : http://www.json.org/json-id.html

Gambar 2.8 String JSON

5. Angka

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



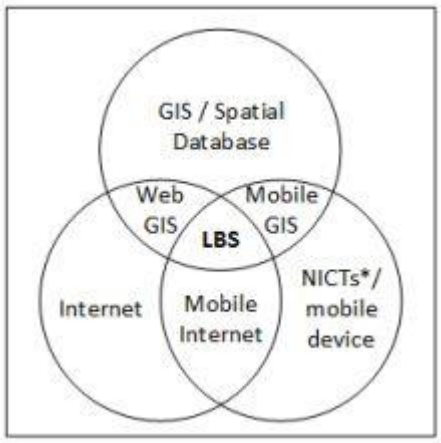
Sumber Gambar : http://www.json.org/json-id.html

Gambar 2.9 JSON Angka

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

2.16 Location Base Service (LBS)

Layanan Berbasis lokasi adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan *mobile network*, yang dilengkapi kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut. LBS memberikan kemungkinan komunikasi dan interaksi dua arah. Oleh karena itu pengguna memberitahu penyedia layanan untuk mendapatkan informasi yang dia butuhkan, dengan referensi posisi pengguna tersebut. Layanan berbasis lokasi dapat digambarkan sebagai suatu layanan yang berada pada pertemuan tiga teknologi yaitu : *Geographic Information System, Internet Service, dan Mobile Devices*, hal ini dapat dilihat pada gambar LBS adalah pertemuan dari tiga teknologi [22].

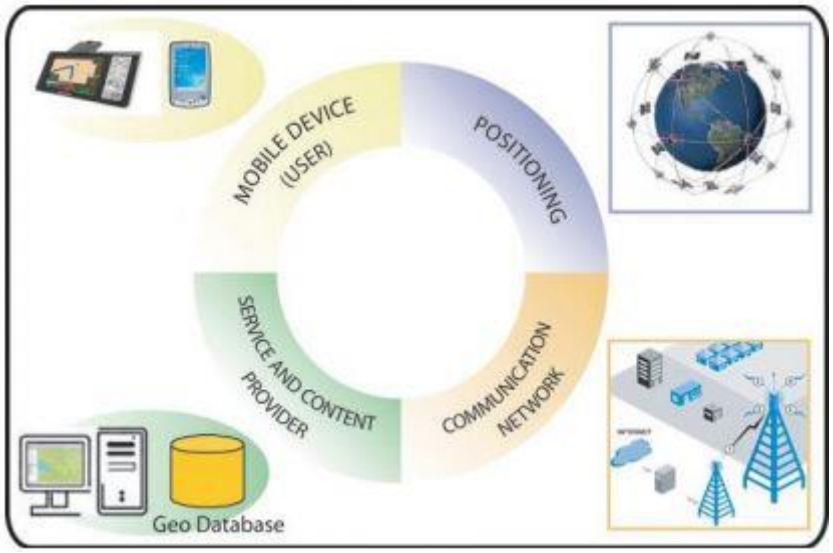


Sumber gambar: Jurnal Aplikasi Location Based Service untuk Sistem Informasi Publikasi Acara pada Platform Android [22]

Gambar 2.10 LBS Sebagai simpang tiga teknologi

Secara Garis besar jenis Layanan Berbasis Lokasi juga dapat dibagi menjadi dua, yaitu:

1. *Pull Service*: Layanan diberikan berdasarkan permintaan dari pelanggan akan kebutuhan suatu informasi. Jenis layanan ini dapat dianalogikan seperti menggakses suatu web pada jaringan internet.
2. *Push Service*: Layanan ini diberikan langsung oleh *sevice provider* tanpa menunggu permintaan dari pelanggan, tentu saja informasi yang diberikan tetap berkaitan dengan kebutuhan pelanggan.
3. *Positioning Component*: Untuk memproses suatu layanan maka posisi pengguna harus diketahui
4. *Service and Aplication Provider*: Penyedia layanan menawarkan berbagai macam layanan kepada pengguna dan bertanggung jawab untuk memproses informasi yang diminta oleh pengguna.
5. *Data and Content Provider*: Penyedia layanan tidak selalu menyimpan semua data yang dibutuhkan yang bisa diakses oleh pengguna. Untuk itu, data dapat diminta dari *data and content provider*.



Sumber gambar: Jurnal Aplikasi Location Based Service untuk Sistem Informasi Publikasi Acara pada Platform Android [22]

Gambar 2.11 Komponen dasar LBS

Selanjutnya mengirim informasi yang telah diolah melaui jaringan internet dan jaringan komunikasi. Pada akhirnya pengguna dapat menerima informasi yang diinginkan.

2.17 Mysql

MySQL merupakan *software RDBMS (Relational Database Management System)* atau server *database* yang dapat mengelola *database* dengan sangat cepat, dapat menampung data dalam jumlah sangat besar, dapat diakses oleh banyak user *multi-user),* dan dapat melakukan suatu proses secara sinkron atau berbarengan *(multi-threaded)* [23].

Saat ini, MySQL banyak digunakan di berbagai kalangan untuk melakukan penyimpanan dan pengolahan data, mulai dari kalangan akademis sampai ke industry, baik industri kecil, menengah, maupun besar.

Lisensi MySQL terbagi menjadi dua. Anda dapat menggunakan MySQL sebagai produk *opn source* di bawah GNU *(General Public License)* secara gratis atau dapat membeli lisensi dari versi komersialnya. MySQL versi komersial tentu memiliki nilai lebih atau kemampuan-kemampuan yang tidak disematkan pada

versi gratis. Pada kenyataannya, untuk keperluan industri menengah kebawah, versi gratis masih dapat digunakan dengan baik.

2.18 Structured Query Language (SQL)

SQL adalah pendekatan dari *Structural Query Language*, yang merupakan Bahasa atau kumpulan perintah standar yang digunakan untuk berkomunikasi dengan *database.* Perintah dalam SQL diklasifikasikan menjadi lima bagian besar [23], yaitu :

1. *Data Definition Language* (DDL)

Data Definition Language (DDL) adalah sekumpulan perintah SQL yang berkaitan dengan pembuatan, perubahan dan penghapusan *database* prosedur/fungsi, *trigger,* dan sebagainya.

Perintah SQL yang termasuk kategori DDL adalah :

1. CREATE DATABASE, Berfungsi untuk membuat *database* dan objek-objek di dalam *database.*
2. CREATE TABLE, Berfungsi untuk membuat tabel di dalam database.
3. CREATE INDEX , digunakan untuk membuat index.
4. CREATE VIEW, digunakan untuk membuat view.
5. DROP DATABASE,digunakan untuk menghapus basis data.
6. DROP TABLE, digunakan untuk menghapus tabel.
7. DROP INDEX, digunakan untuk menghapus index.
8. DROP VIEW, digunakan untuk menghapus view.
9. ALTER TABLE, digunakan untuk mengubah struktur suatu tabel.

2. *Data Manipulation Language* (DML)

Data Manipulation Language (DML) adalah kumpulan perintah SQL yang berkaitan dengan data atau isi dari suatu tabel. Dengan perintah-perintah di dalam DML, kita dapat memanipulasi (menambah, mengubah, dan menghapus) data yang terdapat pada suatu tabel secara mudah.

Perintah-perintah yang termasuk ke dalam DML adalah :

1. INSERT, berfungsi untuk menambah atau memasukan data baru ke dalam tabel.
2. UPDATE, berfungsi untuk mengubah data dalam tabel dengan nilai baru.
3. DELETE, berfungsi untuk menghapus data dari suatu tabel.

3. *Data Control Language* (DCL)

Data Control Language (DCL) adalah salah satu komponen SQL yang berfungsi untuk mengontrol hak akses *user*. Perintah yang termasuk ke dalam DCL adalah GRANT dan REVOKE. Berikut penjelasannya.

1. GRANT, Perintah GRANT digunakan untuk memberikan hak akses (*privilege*) kepada *user* tertentu. Melalui perintah hak akses semacam ini, seseorang *user* memiliki keterbatasan dalam menggunakan *database* sehingga data akan aman dari pihak-pihak yang tidak berkepentingan.
   Untuk melihat daftar hak akses yang dimiliki oleh seorang *user* , gunakan perintah SHOW GRANTS. Untuk mengeksekusi perintah GRANT, anda perlu memiliki hak akses GRANT OPTION yang sebelumnya diberikan oleh administrator kepada anda.
2. REVOKE, Perintah REVOKE merupakan kebalikan dari perintah GRANT, yang berfungsi untuk mencabut salah satu atau beberapa hak akses dari *user* tertentu di dalam *database.* Sama seperti perintah GRANT, untuk menjalankan perintah REVOKE anda perlu memiliki hak akses GRANT OPTION.

4. *Transaction control Language* (TCL)

Selain pengolahan hak akses user, dalam SQL kita juga dapat mengontrol transaksi data yang telah dilakukan. Perintah-perintah yang berkaitan dengan pengontrolan transaksi data digolongkan ke dalam *Transactional Control Language* (TCL). Perintah yang termasuk ke dalam TCL adalah COMMIT dan ROLLBACK. Berikut penjelasannya.

1. COMMIT, perintah COMMIT berfungsi untuk menyimpan perubahan-perubahan yang dilakukan terhadap *database* (melalui perintah INSERT, UPDATE, atau DELETE) secara permanent.
2. ROLLBACK, perintah ROLLBACK merupakan kebalikan dari perintah COMMIT, yang berfungsi untuk membatalkan transaksi atau perubahan-perubahan yang telah dilakukan ke dalam *database* (melalui perintanh INSERT, UPDATE, atau DELETE). Dengan melakukan pembatalan transaksi, data di dalam *database* akan kembali ke keadaan awal (keadaan sebelum dilakukan perubahan). Sehingga dengan kata lain, perubahan yang dilakukan tidak akan pernah disimpan ke dalam *database.*

2.19 Personal Home Page (PHP)

PHP Pertama kali ditemukan pada 1995 oleh seorang *Software Developer* bernama Rasmus Lerdrof. Ide awal PHP adalah ketika itu Rasmus ingin mengetahui jumlah pengunjung yang membaca resume onlinenya. Script yang dikembangkan baru dapat melakukan dua pekerjaan, yakni merekam informasi *visitor*, dan menampilkan jumlah pengunjung dari suatu *website*. Dan sampai sekarang kedua tugas tersebut masih tetap populer digunakan oleh dunia web saat ini. Kemudian, dari situ banyak orang di milis mendiskusikan script buatan Rasmus Lerdrof, hingga akhirnya rasmus mulai membuat sebuah *tool/script,* bernama *Personal Home Page* (PHP) [24].

Kebutuhan PHP sebagai *tool* yang serba guna membuat Lerdorf melanjutkan untuk mengembangkan PHP hingga menjadi suatu bahasa tersendiri yang mungkin dapat mengkonversikan data yang di masukkan melalui *Form* HTML menjadi suatu variabel, yang dapat dimanfaatkan oleh sistem lainnya. Untuk merealisasikannya, akhirnya Lerdrof mencoba mengembangkan PHP menggunakan bahasa C ketimbang menggunakan Perl. Tahun 1997, PHP versi 2.0 di rilis, dengan *nama Personal Home Page Form Interpreter* (PHP-FI). PHP Semakin popular, dan semakin diminati oleh programmer web dunia.

Rasmus Lerdrof benar-benar menjadikan PHP sangat populer, dan banyak sekali *Team Developer* yang ikut bergabung dengan Lerdrof untuk

mengembangkan PHP hingga menjadi seperti sekarang, Hingga akhirnya dirilis versi ke 3-nya, pada Juni 1998, dan tercatat lebih dari 50.000 programmer menggunakan PHP dalam membuat website dinamis.

Pengembangan demi pengembangan terus berlanjut, ratusan fungsi ditambahkan sebagai fitur dari bahasa PHP, dan di awal tahun 1999, netcraft mencatat, ditemukan 1.000.000 situs di dunia telah menggunakan PHP. Ini membuktikan bahwa PHP merupakan bahasa yang paling populer digunakan oleh dunia *web development*. Hal ini mengagetkan para developernya termasuk Rasmus sendiri, dan tentunya sangat diluar dugaan sang pembuatnya. Kemudian Zeev Suraski dan Andi Gutsman selaku *core developer* (programmer inti) mencoba untuk menulis ulang PHP Parser, dan diintegrasikan dengan menggunakan Zend scripting engine, dan mengubah jalan alur operasi PHP. Dan semua fitur baru tersebut di rilis dalam PHP 4.

13 Juli 2004, evolusi PHP, PHP telah mengalami banyak sekali perbaikan di segala sisi, dan wajar jika netcraft mengumumkan PHP sebagai bahasa web populer di dunia, karena tercatat 19 juta domain telah menggunakan PHP sebagai *server side* scriptingnya. PHP saat ini telah Mendukung XML dan Web Services, Mendukung SQLite. Tercatat lebih dari 19 juta domain telah menggunakan PHP sebagai *server* scriptingnya. Benarbenar PHP sangat mengejutkan.

Yang menjadikan PHP berbeda dengan HTML adalah proses dari PHP itu sendiri. HTML merupakan bahasa statis yang apabila kita ingin merubah konten/isinya maka yang harus dilakukan pertama kali nya adalah, membuka file-nya terlebih dahulu, kemudian menambahkan isi kedalam file tersebut. Beda hal nya dengan PHP. Bagi anda yang pernah menggunakan CMS seperti wordpress atau joomla yang dibangun dengan PHP tentunya, ketika akan menambahkan konten kedalam website, anda tinggal masuk kedalam halaman admin, kemudian pilih *new artikel* untuk membuat halaman/*content* baru. Artinya hal ini, seorang user tidak berhubungan langsung dengan scriptnya. Sehingga seorang pemula sekalipun dapat menggunakan aplikasi seperti itu.

Keunggulan PHP antara lain :

1. Gratis

   Apa yang membuat PHP begitu berkembang sangat pesat hingga jutaan domain menggunakan PHP, begitu populernya PHP? Jawabannya adalah karena PHP itu gratis.

2. *Cross platform*

   Artinya dapat di gunakan di berbagai sistem operasi, mulai dari linux, windows, mac os dan os yang lain.

3. Mendukung banyak *database*

   PHP telah mendukung banyak *database*, ini mengapa banyak *developer web* menggunakan PHP Adabas D Adabas D, dBase dBase, Empress Empress, FilePro (*read-only*) FilePro (*read-only*) Hyperwave, IBM DB2, Informix, Ingres, InterBase, FrontBase mSQL, Direct MS-SQL, MySQL MySQL, ODBC, Oracle (OCI7 and OCI8), Ovrimos, PostgreSQL SQLite, Solid, Sybase, Velocis, Unix dbm.

4. *On The Fly*

   PHP sudah mendukung *on the fly*, artinya dengan php anda dapat membuat *document text*, Word, Excel, PDF, menciptakan *image* dan *flash*, juga menciptakan file-file seperti zip, XML, dan banyak lagi.

2.20 CodeIgniter (CI)

CodeIgniter adalah sebuah Application Development Framework (*toolkit*) bagi orang-orang yang ingin membangun website menggunakan PHP. Tujuannya adalah untuk memungkinkan Anda mengembangkan proyek-proyek lebih cepat daripada Anda menulis kode dari awal, tersedia banyak *library* untuk tugas-tugas yang biasa diperlukan, serta antarmuka dan struktur logis yang sederhana untuk mengakses *library* ini. CodeIgniter memungkinkan Anda fokus pada proyek Anda dengan meminimalkan jumlah kode yang dibutuhkan untuk tugas yang diberikan [25].

1. Sejarah singkat Codeigniter

CodeIgniter lahir pada tahun 2006. CodeIgniter adalah kerangka kerja PHP open source yang hebat dengan *foot print* yang sangat kecil, dibuat oleh Rick Ellis pada tahun 2006. CodeIgniter lahir dari Expression Engine, yang merupakan koleksi kelas *refactored* yang aslinya ditulis untuk CMS unggulan EllisLab. Diambil dari fungsi khusus aplikasi, CodeIgniter dibuat sebagai *toolkit* yang sederhana dan elegan, yang memungkinkan perkembangan pesat dari situs web dan aplikasi web, menarik ribuan pengembang PHP berbakat [26].

Pada tahun 2008 pemimpin industri Dalam lingkungan yang sekarang dan merasa jenuh dengan kerangka kerja PHP lain yang telah ada, CodeIgniter adalah satu-satunya pilihan padat bagi pengembang yang berkecimpung hidup di dunia akun *shared hosting* setiap harinya dan klien dengan *deadline*. CodeIgniter terbang dalam menghadapi kerangka kerja yang sangat besar dan tidak terdokumentasi dengan benar, yang menyebabkan pencipta PHP Rasmus Lerdorf, seorang kritikus kerangka kerja vokal, untuk memuji CodeIgniter.



Sumber Gambar: https://ellislab.com/codeigniter

Gambar 2.12 Logo CodeIgniter

Pada tahun 2009, EllisLab meluncurkan ExpressionEngine 2.0, dibangun kembali pada framework CodeIgniter. Basis kode fleksibel ini memungkinkan pengembang pihak ketiga untuk lebih mudah membangun alat dan *add-on* yang

hebat untuk ExpressionEngine, yang menghasilkan ledakan bakat berkualitas tinggi di komunitas ExpressionEngine.

2. Keunggulan Codeigniter

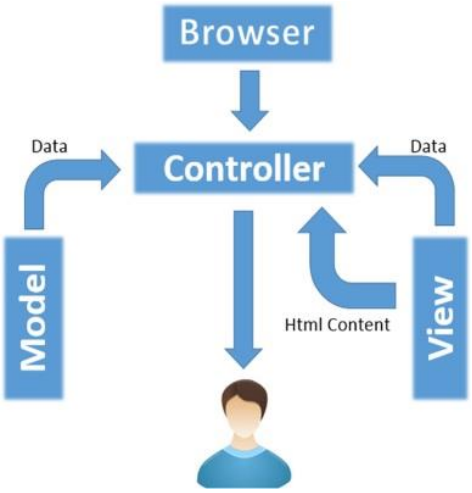
Fitur dalam dan dari diri mereka sendiri adalah cara yang sangat buruk untuk menilai sebuah aplikasi karena mereka belum memberitahu Anda apa-apa tentang pengalaman pengguna, atau bagaimana itu dirancang secara cerdas atau intuitif. Fitur tidak mengungkapkan apa-apa tentang kualitas kode, atau kinerja, atau perhatian terhadap detail, atau penerapan keamanan. Satu-satunya cara untuk benar-benar menilai sebuah aplikasi adalah dengan mencobanya dan mengenal kodenya. Instalasi CodeIgniter sangat mudah, oleh karena itu kami mendorong Anda untuk melakukan hal itu. Sementara itu, ini adalah daftar fitur utama CodeIgniter [25].

1. Sistem berbasis MVC
2. Sangat ringan
3. Database Class yang lengkap dengan dukungan untuk beberapa *platform*.
4. Dukungan query *builder* untuk *database*
5. *Form* dan validasi data
6. Keamanan dan penyaringan XSS
7. Manajemen *sessions*
8. Email *Sending Class*. Mendukung lampiran, HTML/Text email, beberapa protokol (*sendmail*, SMTP, dan *Mail*) dan banyak lagi.
9. *Image Manipulation Library* (cropping, mengubah ukuran, memutar, dll). Mendukung GD, *ImageMagick*, dan Netpbm
10. *File Uploading Class*
11. FTP *Class*
12. *Localization*
13. *Pagination*
14. Enkripsi data
15. *Benchmarking*
16. *Full Page Caching*

17. *Error Logging*
18. *Application Profiling*
19. *Calendaring Class*
20. *User Agent Class*
21. *Zip Encoding Class*
22. *Template Engine Class*
23. *Trackback Class*
24. XML-RPC *Library*
25. *Unit Testing Class*
26. *Search-engine Friendly* URLs
27. *Flexible* URI *Routing*
28. Dukungan untuk *Hooks* dan *Class Extensions*
29. Kumpulan *library* fungsi-fungsi "*helper*" yang besar

3. *Model-View-Controller*

CodeIgniter didasarkan pada pola pengembangan *Model-View-Controller*. MVC adalah pendekatan perangkat lunak yang memisahkan logika aplikasi dari presentasi. Dalam prakteknya, itu memungkinkan halaman web Anda memiliki *scripting* yang minimal karena presentasi terpisah dari *scripting* PHP.



Sumber Gambar : http://www.w3ii.com/id/codeigniter/codeigniter_mvc_framework.html

Gambar 2.13 Konsep Model View Controller(MVC)

1. *Model*, Model mewakili struktur data Anda. Biasanya *class* model Anda akan berisi fungsi yang membantu Anda mengambil, menyimpan, dan memperbarui informasi dalam *database* Anda.
2. *View* adalah informasi yang disajikan kepada pengguna. *View* yang biasanya akan menjadi halaman web, tetapi dalam CodeIgniter, view juga bisa menjadi bagian dari sebuah halaman seperti *header* atau *footer*. Hal ini juga dapat menjadi halaman RSS, atau jenis-jenis lain dari "halaman".
3. *Controller* berfungsi sebagai perantara antara *Model, View, dan resource* lain yang diperlukan untuk memproses HTTP *request* dan menghasilkan halaman web.

CodeIgniter memiliki pendekatan yang cukup longgar untuk MVC karena Model tidak selalu diperlukan. Jika Anda tidak perlu menambahkan pemisahan, atau menemukan bahwa mempertahankan sebuah model memerlukan kompleksitas lebih dari yang Anda inginkan, Anda bisa mengabaikan mereka dan membangun aplikasi Anda dengan minimal menggunakan *Controller* dan *View*. CodeIgniter juga memungkinkan Anda untuk memasukkan script Anda sendiri, atau bahkan mengembangkan *library* inti untuk sistem, memungkinkan Anda untuk bekerja dengan cara yang paling masuk akal bagi Anda.

4. Desain and Sasaran Arsitektur

Tujuan kami untuk CodeIgniter adalah kinerja yang maksimum, kemampuan, dan fleksibilitas dalam hal terkecil, *package* seringan mungkin. Untuk memenuhi tujuan ini kami berkomitmen untuk melakukan benchmarking, *re-factoring*, dan menyederhanakan setiap langkah dari proses pembangunan, menolak semua yang tidak menjadi tujuan berikutnya. Dari sudut pandang teknis dan arsitektur, CodeIgniter diciptakan dengan tujuan sebagai berikut:

1. *Dynamic Instantiation*. Dalam CodeIgniter, komponen dimuat dan rutinitas dieksekusi hanya jika diminta, bukan secara global. Tidak ada asumsi yang dibuat oleh sistem tentang apa yang mungkin diperlukan di luar resource utama minimal, sehingga sistem ini sangat ringan secara *default*. *Event*,

yang dipicu oleh HTTP *request*, dan *controller* dan *view* yang Anda rancang akan menentukan apa yang dipanggil.

2. *Loose Coupling*. *Coupling* adalah sejauh mana komponen-komponen dari sistem mengandalkan satu sama lain. Semakin sedikit komponen yang bergantung satu sama lain, maka komponen tersebut lebih dapat digunakan kembali dan sistem menjadi fleksibel. Tujuan kami adalah sistem yang sangat longgar *(very loosely coupled system).*
3. *Component Singularity*. *Singularity* adalah sejauh mana komponen memiliki tujuan yang difokuskan secara sempit. Dalam CodeIgniter, setiap *class* dan fungsinya sangat otonom supaya memungkinkan kegunaan yang maksimal.

CodeIgniter itu *dynamically instantiated*, sistem yang *loosely coupled* dengan singularitas komponen yang tinggi. Codeigniter berusaha untuk sederhana, fleksibel, dan kinerja tinggi dalam paket dengan *footprint* yang kecil.

Dalam Penelitian ini peneliti menggunakan *framework* CodeIgniter untuk membangun *Web Service* yang akan digunakan aplikasi dalam pertukaran data baik mengirim atau menerima data ke aplikasi dari *database.*

2.21 Unified Modeling language (UML)

UML (*Unifed Modeling Language*) adalah salah satu standar Bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis & desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. Ada beberapa diagram yang digunakan proses pembuatan perangkat lunak berorientasi objek diantaranya, *use case diagram, activity diagram, class diagram dan sequence diagram* [27].

2.21.1 Use Case Diagram

Use case atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat.

Ada dua hal utama pada use case yaitu pendefinisian apa yang disebut actor dan *use case.*

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibangun.
2. *Use Case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

2.21.2 Class Diagram

Diagram kelas atau *Class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

2.21.3 Sequence Diagram

Diagram Sekuen atau *Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Penomoran pesan berdasarkan urutan interaksi pesan. Penggambaran letak pesan harus berurutan, pesan yang lebih atas dari lainnya adalah pesan yang berjalan terlebih dahulu.

2.21.4 Activity Diagram

Diagram Aktivitas atau *Activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. 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 pengelompokan 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.