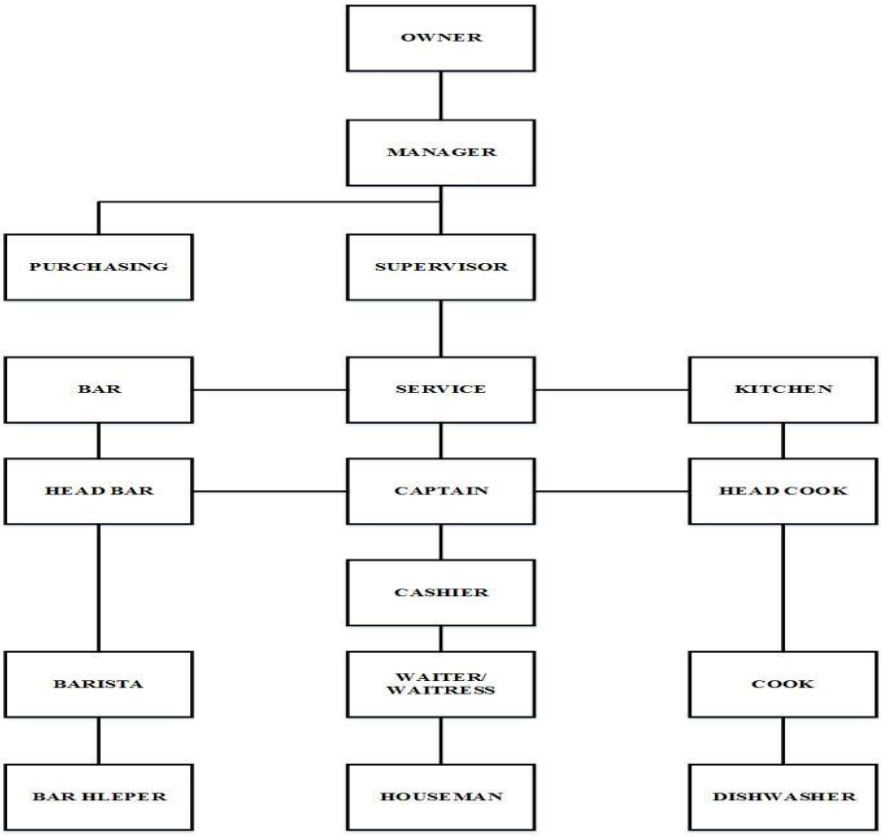
BAB 2
TINJAUAN PUSTAKA

2.1 Cafe Insomnia

Cafe Insomnia adalah usaha yang bergerak dalam bidang resto dan beralamat di Jalan Cikutra No 187 Bandung, Dengan mengusung konsep modern elegan minimalis, Cafe Insomnia menghadirkan suasana modern dalam interior dang penyajian berbagai macam jenis menu. Struktu organisasi Cafe Insomnia di jelaskan pada gambar berikut



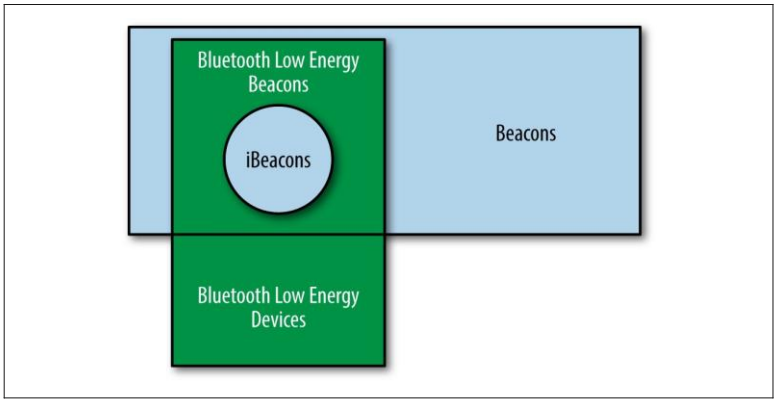
Gambar 2-1 Struktur organisasi Cafe Insomnia

2.2 Beacon

BLE sering disebut dengan *Bluetooth Smart* adalah bluetooth yang membutuhkan daya rendah dalam kerjanya. BLE merupakan peningkatan spesifikasi dari Bluetooth yang ada sebelumnya yang dirancang untuk penggunaan dengan konsumsi daya rendah. BLE beacon mentransmisikan informasi indentitias

yang dapat digunakan oleh suatu aplikasi untuk mengetahui ruangan atau benda apa yang telah terpasang beacon.

iBeacon merupakan merupakan bagian Spesifik dari BLE beacon. Seperte lihat pada



Gambar 2-2 Relasi antara iBeacon,BLE dan Beacon[2]

Pada gambar diatas terlihat bahwa semua iBeacon merupakan Beacon dan semua beacon merupakan perangkat BLE.

Dengan daya pancar rendah, Konsumsi daya pada beacon cukup rendah. Beacon dapat berfungsi lama dengan hanya memanfaatkan daya battery. Hanya dengan sumber daya dari baterai berbentuk koin (seperti baterai bios komputer) sebuah modul BLE dapat bertahan hingga 3 tahun.

Beacon tidak membutuhkan koneksi internet untuk dapat berfungsi. Tetapi perlu dicatat, walaupun Beacon tidak membutuhkan koneksi internet, aplikasi yang memanffatkan beaconlah yang terkadang membutuhkan koneksi internet untuk dapat berfungsi.

2.2.1 Protokol iBeacon

Cara kerja beacon cukup simple, untuk menggambarkan bagaimana teknologi ini bekerja dapat dapat dianalogikan seperti mercusuar. Mercusuar memancarkan cahaya yang siapapun dapat melihat, beacon memancarkan sinyal yang dapat dilihat oleh aplikasi atau device yang kompatibel.

2.2.1.1 Radio

Beacon biasanya menggunakan *bluetooth* dengan frekuensi *unlicensed* 2.4GHz ISM (*industrial, scientific and medical*). Ini berarti beacon menggunakan spectrum yang sama seperti wi-fi, pintu garasi otomatis, telepon nirkabel dan microwave. Namun beacon telah mengalokasikan channel untuk membantu beacon menghindari intervensi dari chanel *wi-fi* yang ada didekatnya[3]

2.2.1.2 *Broadcast* Signal

Beacon menggunakan GATT profile untuk mem*Broadcast* infromasi. Informasi yang di *Broadcast* bersifat statis, tidak menggandung informasi untuk pengguna seperti penawaran, iklan dan lain-lain. Adapun pesan yang di *Broadcast* adalah sebagai berikut

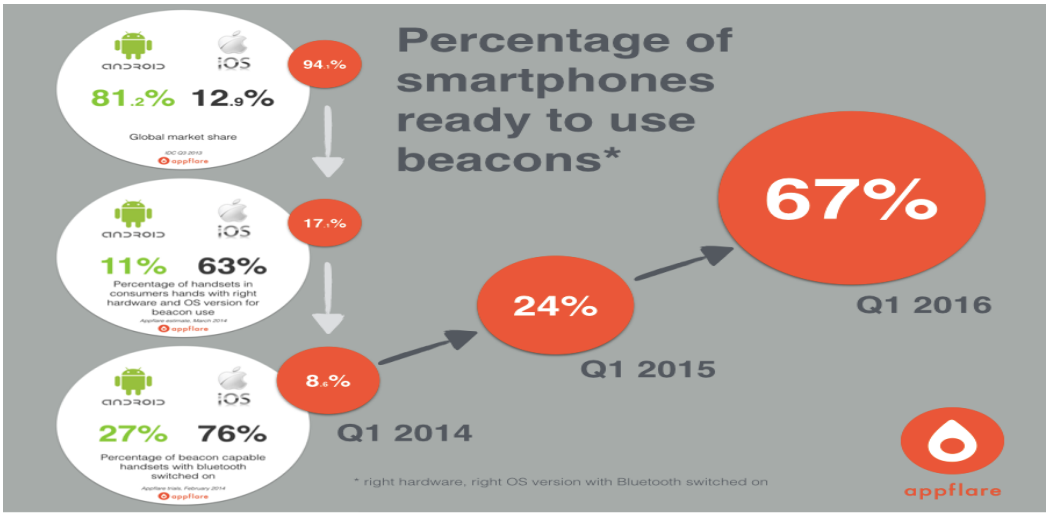
- Beacon identifier (2 bytes, 0x02-15)
  Beberapa sumber menyatakan besaran kpasitas yangdiganakan aoleh beacon identifier adalah 2 byte, yang mana byte pertama digunakan sebagai penanda identifikasi suatu protocol. Byte kedua sebagai indikasi jarak atau besaran (15 in hex is 21 decimal)
- Proximity UUID (16 bytes)
  digunakan untuk mengidentifikasi pemilik / deployer dari kelompok beacon (misalnya, merek dari rangkaian toko)
- Major (2 bytes)
  Digunakan untuk mengidentifikasi lokasi (sebagai contoh: lantai toko)
- Minor (2 byte)
  Digunakan untuk mengidentifikasi micro-location (sebagai contoh: barang yang ada pada lantai toko tersebut)
- Measured power (1 byte**)**
  Berupa informasi daya sinyal yang diterima, digunakan untuk menetukan jarak dari beacon.

2.2.2 Perangkat *mobile* yang *support* terhadap BLE

Contoh beberapa perangkat *mobile* yang support terhadap BLE terlihat pada t**Error! Reference source not found.**

Tabel 2-1 Contoh perangkat *mobile* yang suport BLE[2]

Manufacturer	Product Name
Apple	iPad (Air, Mini, 3rd & 4th gen)
Apple	iPhone (5s, 5c, 5 & 4s)
Google	Nexus 5, Nexus 4
Google	Nexus 7, Nexus 10
BlackBerry	Q10
BlackBerry	Z10
HTC	One, One Max
Microsoft	Surface
Motorola	Droid RAZR, Ultra, Maxx, Mini
Motorola	Moto G, X
Samsung	Galaxy Series
Sony	Xperia Series



Gambar 2-3 Persentase *smartphone* yang siap menggunakan beacon[3]

2.3 Pemesanan

Pemesanan adalah suatu aktifitas yang dilakukan oleh pelanggan sebelum membeli. Untuk mewujudkan kepuasan pelanggan maka perusahaan harus mempunyai sebuah sistem pemesanan yang baik. Menurur Kamus Besar Bahasa Indonesia yang dimaksud pemesanan adalah "proses, perbuatan, cara memesan (tempat, barang,dsb) kepada orang lain".

Menurut Edwin dan Chris (1999:1) Pemesanan dalam arti umum adalah perjanjian pemesanan tempat antara 2 (dua) pihak atau lebih, perjanjian pemesanan tempat tersebut dapat berupa perjanjian atas pemesanan suatu ruangan, kamar, tempat duduk dan lainnya, pada waktu tertentu dan disertai dengan produk jasanya. Produk jasa yang dimaksud adalah jasa yang ditawarkan pada perjanjian pemesanan tempat tersebut, seperti pada perusahaan penerbangan atau perusahaan pelayaran adalah perpindahan manusia atau benda dari satu titik (kota) ketitik (kota) lainnya.

Bauran pemasaran menurut Philip Kotler (2002:18), adalah sekumpulan alat pemasaran yang dapat digunakan oleh perusahaan untuk mencapai tujuan pemasarannya dalam pasar sasaran.Dalam aktifitas pemesanan harus ada 4P, penjelasan arti dari 4P sebagai berikut :

1. *Product* (produk).

   Adalah segala sesuatu yang dapat ditawarkan ke pasar untuk diperhatikan, diperoleh dan digunakan atau dikonsumsi untuk dapat memenuhi kebutuhan dan keinginan meliputi barang fisik, jasa, orang, tempat, organisasi, gagasan dan ide.

2. *Price* (harga).

   Yaitu jumlah uang yang harus dibayar oleh pelanggan untuk memperoleh produk atau jasa untuk memperoleh satu buah produk dan hendaknya harga akan dapat terjangkau oleh pelanggan.

3. *Place* (saluran distribusi/tempat).

   Termasuk aktivitas perusahaan untuk menyalurkan produk atau jasa tersedia bagi pelanggan. Kemudahan akses terhadap jasa bagi para pelanggan. Tempat dimana produk Serta merupakan keputusan distribusi menyangkut

kemudahan akses terhadap jasa bagi para pelanggan. Tempat dimana produk tersedia dalam sejumlah saluran distribusi dan outlet yang memungkinkan pelanggan dapat dengan mudah memperoleh suatu produk.

4. *Promotion* (promosi).

   Aktivitas yang meliputi berbagai metode, yaitu iklan, promosi penjualan, mengkomunikasikan produk dan membujuk pelanggan, Menggambarkan berbagai macam cara yang ditempuh perusahaan dalam rangka menjual produk ke pelanggan. Penjualan tatap muka dan hubungan masyarakat, mengkomunikasikan produk dan membujuk pelanggan, Menggambarkan berbagai macam cara yang ditempuh perusahaan dalam rangka menjual produk ke pelanggan.

2.4 Android

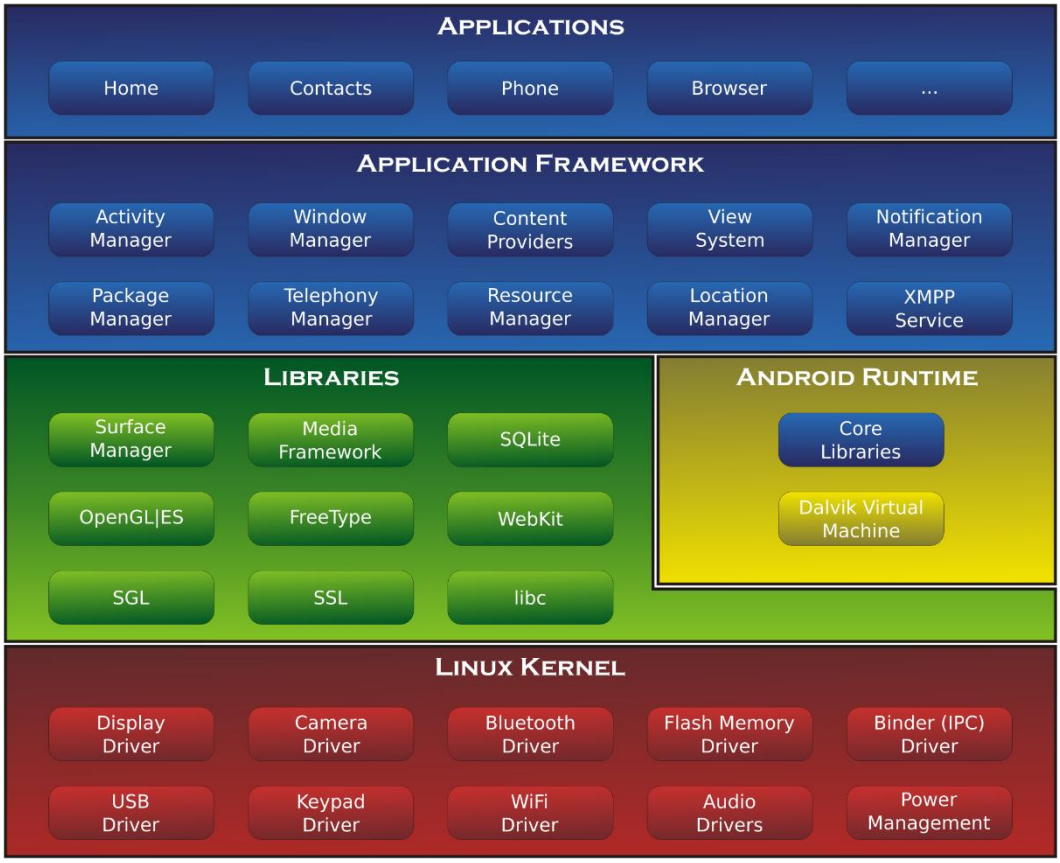
Android merupakan suatu *software stack* untuk *mobile device*. Di dalamnya terdapat sistem operasi, *middleware*, dan *key application*. Aplikasi pada *platform* ini dikembangkan dalam bahasa pemrograman java[9]. Android sendiri memiliki banyak fitur diantaranya adalah:

1. Merupakan sebuah *ApplicationFramework* sehingga programmer dapat menggunakan beberapa fungsi yang telah disediakan.
2. *Dalvik virtual machine*. Tiap aplikasi dalam Android memiliki instance *virtual machine* yang dapat bekerja secara efisien dalam lingkungan memoriyang terbatas.
3. *Integrated browser*. Web browser berbasis WebKit *engine* terdapat *pada browser default* Android atau pun dapat diintegrasikan dengan aplikasi lain.
4. *Optimized graphics*. *Library grafis* 2D yang kaya dan 3D berbasis OpenGL ES 1.0 yang mendukung akselerasi *hardware*.
5. SQLite. Basis data relasional yang ringan namun sangat powerful.
6. *Media Support*. Mendukung berbagai format audio, video, dan gambar (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF) 7
7. GSM Telephony. Mendukung fungsi komunikasi GSM.

8. Bluetooth ,EDGE,3G dan WiFi. Mendukung komunikasi pada jaringan (tergantung *hardware*).
9. Kamera, GPS, kompas dan accelerometer. Mendukung berbagai fitur yang disediakan oleh hardware.
10. Kakas pengembangan yang lengkap. Termasuk device emulator, tools untuk debugging, profiling memori dan performa, plugin untuk Eclipse IDE.

2.4.1 Arsitektur Android

Android terdiri dari beberapa stack *software* yang terdiri dari: Applications, *ApplicationFramework*, Libraries, Android Runtime dan *Kernel* Linux. Arsitektur lengkap *platform* ini dapat dilihat pada Gambar



Gambar 2-4 Arsitektur Android[11]

2.4.2 *Aplication*

*Application*merupakan program yang langsung berhubungan dengan *user*. Baik program yang merupakan bawaan dari Android sendiri maupun program yang dibuat oleh developer menggunakan bahasa pemrograman java. Contoh program bawaan dari *platform* Android sendiri adalah email client, program SMS, calendar, maps, web browser, contact dan sebagainya.

2.4.3 *ApplicationFramework*

Lapisan ini berisi sekumpulan API yang dapat digunakan oleh programmer maupun core *Application*dari Android. Lapisan ini dirancang untuk memudahkan penggunaan komponen dari Android sendiri. Aplikasi manapun dalam Android dapat berbagi fungsi sehingga aplikasi lain dapat memanfaatkannya

Aplikasi pada Android disusun atas beberapa komponen:

1. Sekumpulan *Views*. Digunakan untuk mengatur tampilan pada aplikasi. Contohnya adalah *lists, grids, text box, button*, bahkan *embeddable web browser*.
2. *Content providers*. Komponen yang mengatur agar aplikasi dapat mengakses *resources* dari aplikasi lain (seperti *Contacts*), atau berbagi data dengan aplikasi lain.
3. *Resource Manager*. Menyediakan akses ke pada *resource non-code* seperti *localized string, grafik* dan *file layout*.
4. *Notification Manager*. Memungkinkan agar suatu aplikasi dapat menampilkan peringatan yang dapat di kostumasi pada status bar.
5. *Activity Manager*. Mengatur siklus aplikasi dan navigasi antar aplikasi yang sedang berjalan.

2.4.4 *Libraries*

Android mendukung beberapa library C/C++ yang digunakan pada berbagai komponen Android. Kemampuan ini dapat diakses oleh developer melalui Android *ApplicationFramework*. Beberapa library diantaranya adalah:

1. *System C library*. Implementasi library C standar (libc).

2. *Media Libraries*. Mendukung berbagai format multimedia (termasuk MPEG4, H.264, MP3, AAC, AMR, JPG, PNG).
3. *Surface Manager*. Mengatur akses ke subsistem display.
4. LibWebCore. *Engine web browser modern*.
5. SGL. *Engine* grafis 2D.
6. 3D *Library*. Implementasi OpenGL ES 1.0 yang mendukung akselerasi *hardware*.
7. *FreeType. Rendering* untuk bitmap dan vector *font*.
8. SQLite. Basis data relasional yang kecil namun sangat ampuh

2.4.5 Android Runtime

Tiap aplikasi pada Android memiliki proses-nya masing-masing. Tiap aplikasi tersebut memiliki instans dari Dalvik *virtual machine* (VM). Dalvik *virtual machine* dirancang agar suatu device dapat menjalankan beberapa VM secara efisien. Dalvik VM mengeksekusi file dengan format Dalvik *Executable format* (.dex) yang dirancang untuk meminimalkan memory *footprin*t. Dalvik VM berbasis register, dan dapat menjalankan kelas-kelas yang dikompilasi dengan bahasa pemrograman java dan ditransformasikan menjadi format dex. Dalvik VM sendiri bergantung pada *Kernel* Linux untuk fungsi dasarnya, seperti *threading manajemen* memori secara *low-level*.

2.4.6 Linux *Kernel*

Android menggunakan *Kernel* Linux versi 2.6 sebagai sistem utama. Fungsi *kernel* yang digunakan antara lain untuk keamanan, manajemen memori, manajemen proses, manajemen jaringan dan driver model. *Kernel* juga berfungsi sebagai layer abstrak antara hardware dan lapisan lainnya pada *software* stack.

2.4.7 Anatomi Aplikasi Android

Terdapat empat building blocks pada aplikasi Android yaitu:

1. *Activity*
2. *Intent Receiver*
3. *Service*

4. *Content Provider*

Tidak semua aplikasi membutuhkan keempat blok ini, tetapi suatu aplikasi dibuat menggunakan kombinasi beberapa blok ini. Setelah memutuskan blok komponen yang akan digunakan, blok tersebut didaftarkan pada suatu file yang disebut dengan AndroidManifest.xml. File XML ini digunakan untuk menyatakan komponen apa saja yang dibutuhkan oleh sebuah aplikasi dan kemampuan serta kebutuhan aplikasi tersebut.

2.4.8 *Activity*

Activity paling umum diantara keempat building block. *Activity* biasanya suatu tampilan (*screen*) dalam suatu aplikasi. Tiap *activity* diimplementasikan sebagai satu kelas yang meng-extend *base class Activity*. Kelas yang dibuat akan menampilkan *user interface* yang terdiri dari beberapa views dan akan merespon suatu event.

Kebanyakan, aplikasi terdiri dari beberapa *screen*. Contohnya aplikasi pengiriman pesan akan memiliki *screen* untuk mendaftarkan pesan yang masuk, menulis pesan dan *screen* lainnya untuk melakukan pengaturan. Tiap *screen* pada contoh ini akan diimplementasi sebagai suatu *activity*. Aplikasi dapat berpindah ke *screen* lain dengan cara memulai suatu *activity* baru.

Pada suatu kasus tertentu, *activity* akan mengembalikan nilai kepada *activity* sebelumnya. Contohnya *activity* untuk memilih foto akan mengembalikan foto yang dipilih pada *activity* yang memanggilnya. Pada saat *screen* baru terbuka, *screen* sebelumnya akan dihentikan dan *state* nya akan disimpan pada *History stack*. *User* dapat melakukan navigasi ke *activity* sebelumnya melalui *History* tersebut. *Screen* akan di hapus dari *History stack* apabila sistem memutuskan bahwa *screen* tersebut tidak layak untuk berada pada memori. Android menyimpan *History stack* untuk setiap aplikasi yang dijalankan dari tampilan awal (*home screen*)

2.4.9 Intent dan Intent Filter

Android menggunakan kelas khusus bernama intent untuk berpindah dari *screen* ke *screen* lainnya. Sebuah intent mendeskripsikan apa yang ingin aplikasi penuhi. Bagian struktur data dari intent yang paling penting adalah action dan data dari aksi yang dilakukan. Nilai action yang biasanya adalah MAIN, VIEW, PICK, EDIT dan sebagainya.

Data diekspresikan sebagai URI. Sebagai contoh, untuk melihat informasi suatu *contact* seseorang, kita membuat intent dengan action VIEW dan data sebagai URI yang merepresentasikan orang tersebut. Terdapat suatu kelas yang dinamakan IntentFilter. Apabila intent adalah suatu permintaan untuk melakukan sesuatu, IntentFilter adalah deskripsi yang dapat dilakukan oleh intent *activity*. *Activity* yang dapat menampilkan informasi contact dari seseorang akan mempublikasikan IntentFilter yang menyatakan bahwa ia dapat menangani action VIEW apabila dipakai pada data yang merepresentasikan seseorang.

Suatu plikasi mempublikasikan IntentFilter pada file AndroidManifest.xml. Navigasi dari suatu *screen* ke *screen* lainnya dipenuhi dengan cara me-resolve intent. Untuk melakukan navigasi ke depan, *activity* memanggil *startActivity(myIntent)*. Sistem akan mencari IntentFilter untuk semua aplikasi dan memilih *activity* yang IntentFilter-nya yang paling mendekati dengan myIntent. *activity* baru akan mendapat informasi tentang intent yang menyebabkan *activity* tersebut dijalankan.

Proses pencarian intent terjadi sewaktu runtime pada saat *startActivity* dipanggil. Hal tersebut membawa keuntungan sebagai berikut:

1. *Activity* dapat melakukan *reuse functionality* dari komponen lain dengan cara melakukan permintaan dalam bentuk intent.
2. *Activity* dapat di *replace* kapan pun dengan *activity* yang ekuivalen.

2.4.10 *Intent Recivier*

Intent Receiver dapat digunakan pada saat kita menginginkan kode kita dijalankan akibat reaksi dari *external event* tertentu. Contohnya, pada saat telepon

berbunyi, atau pada saat ada data pada jaringan, atau pada suatu waktu tertentu. *Intent Receiver* tidak menampilkan *user interface*, meskipun ia dapat menggunakan *Notification Manager* untuk memperingatkan *user* apabila sesuatu terjadi.

Intent Receiver didaftarkan pada file AndroidManifest.xml, tetapi secara manual juga bisa ditambahkan dengan fungsi *Context.registerReceiver().* Suatu aplikasi tidak harus berjalan pada *Intent Receivernya* agar bisa dipanggil, tetapi sistem yang akan menjalankan aplikasi tersebut. Apabila diperlukan, pada saat *Intent Receiver* mendapat trigger, aplikasi juga dapat mengirimkan intent *Broadcast* kepada aplikasi lainnya dengan fungsi *Context.BroadcastIntent().*

2.4.11 *Service*

Service adalah sebuah kode yang hidup terus-menerus dan berjalan tanpa memiliki tampilan. Contoh yang sederhana adalah suatu aplikasi *media player* yang sedang memainkan lagu dari *play list*. Pada aplikasi *media player*, kemungkinan terdapat satu atau lebih *activity* yang memperbolehkan *user* memilih lagu dan memainkannya. Tetapi musik yang sedang diputar tidak boleh ditangani oleh *activity* karena *user* akan menginginkan musik tetap diputar meskipun ia berpindah ke *screen* baru. Pada kasus ini *activity media player* dapat menjalankan *service* menggunakan fungsi *Context.startService()* untuk menjalankan *service* di *background* yang memutar musik. Sistem akan tetap memutar musik sampai selesai. Selain itu *activity* dapat terhubung dengan *service* menggunakan fungsi *Context.bindService()*. Pada saat terhubung, aplikasi dapat berkomunikasi melalui interface yang diberikan oleh *service* yang bersangkutan. Pada contoh *media player*, *interface service* akan memberikan fungsi *pause, rewind* dan sebagainya.

2.4.12 *Content Provider*

Content provider digunakan untuk pertukaran data antara aplikasi. Data yang dipertukarkan bersifat *shared*. *Content provider* adalah sebuah kelas yang berisi method standar yang memungkinkan aplikasi lain dapat menyimpan dan mengambil suatu data yang ditangani oleh *content provider*

2.4.13 Siklus Hidup Aplikasi Android

Pada banyak kasus, tiap aplikasi pada Android masing-masing memiliki Linux proses. Proses ini diciptakan untuk aplikasi tersebut pada saat kode program tersebut akan dieksekusi, akan terus berjalan sampai tidak dibutuhkan lagi dan sistem memerlukan memori untuk aplikasi lain.

Konsep dasar program Android adalah suatu proses pada aplikasi tidak langsung di atur oleh aplikasi tersebut. Proses tersebut ditentukan oleh sistem melalui kombinasi : informasi aplikasi bagi sistem, seberapa pentingnya aplikasi tersebut bagi *user* dan ketersediaan memori. Contoh kasus *life-cycle bug* adalah IntentReceiver yang menciptakan *thread* pada saat menerima *intent* pada *method* onReceiveIntent() dan selesai dari fungsi tersebut. Setelah fungsi tersebut selesai, sistem menganggap bahwa *Intent Receiver* tersebut menjadi tidak aktif sehingga ia memiliki proses yang tidak dibutuhkan (kecuali terdapat komponen aplikasi lain di dalamnya).

Oleh karena itu, sistem dapat menghapus proses tersebut kapanpun apabila dibutuhkan. Hal tersebut akan menghapus semua *thread* yang berjalan pada proses tersebut. Solusi masalah ini adalah memakai *Service* dari *Intent Receiver*, sehingga sistem mengetahui bahwa ada suatu yang aktif dalam proses tersebut.

Sebagai programmer, kita harus mengetahui bagaimana cara yang tepat untuk merancang aplikasi. Kesalahan pada perancangan akan membuat suatu aplikasi akan dihentikan oleh sistem pada saat yang tidak diinginkan. Untuk menentukan proses mana yang akan dihapus pada situasi *low memory*, Android memiliki "*importancehierarchy*" berdasarkan komponen dan *state* yang sedang berjalan. Urutan berdasarkan kepentingan tersebut adalah:

1. *Foreground process*. Adalah suatu proses yang dibutuhkan untuk berinteraksi dengan *user* saat ini. Berbagai komponen suatu aplikasi dapat mengakibatkan proses di dalamnya menjadi *foreground* proses. Suatu proses dikatakan *foreground* apabila memenuhi kondisi sebagai berikut:
   1.1. Menjalankan *Activity* pada top *screen* yang sedang berinteraksi dengan *user* (setelah *method onResume()* dipanggil).

1.2. Mempunyai IntentReceiver yang sedang berjalan (method IntentReceiver.onReceiveIntent() sedang dieksekusi).

1.3. Memilik *service* yang fungsi *callback*-nya (*Service.onCreate(), Service.onStart()* atau *Service.onDestroy()*) sedang dieksekusi.

1.4. Hanya akan terdapat beberapa proses seperti itu dalam sistem, proses ini hanya akan dihapus apabila sistem tidak memiliki pilihan lain. Contohnya apabila memori tinggal sedikit sehingga proses *foreground* tidak bisa lagi berjalan. Hal ini diperlukan untuk menjaga *user interface* tetap interaktif dengan *user*.

2. *Visible process*. Adalah proses yang memegang *Activity* yang dapat dilihat *user* pada layar tetapi tidak *foreground* (pada saat *method onPause()* dipanggil). Hal ini dapat terjadi, sebagai contoh *activity foreground* muncul dengan dialog sehingga *activity* sebelumnya terlihat di belakangnya. Proses ini sangat penting dan tidak akan di hapus kecuali untuk menjaga aplikasi *foreground* tetap berjalan.
3. *Service process*. Adalah proses yang memegang *Service* yang dimulai dengan method Start*Service*().Meskipun proses ini secara langsung tidak dapat dilihat oleh *user*, biasanya proses ini menjalankan perintah dari *user* (seperti memutar lagu, men-download data dari jaringan dan lain-lain). Sistem akan tetap membiarkan proses ini berjalan selama proses *foreground* dan *visible* masih bisa berjalan.
4. *Background process*. Adalah proses yang memegang *Activity* yang saat ini tidak dapat dilihat oleh *user* (method onStop() sedang dipanggil). Proses ini tidak secara langsung mempengaruhi *user*. Apabila *activity life cycle* telah dipenuhi dengan benar, sistem dapat menghapus proses ini sewaktu-waktu untuk memberikan memori tambahan. Biasanya banyak ditemukan proses semacam ini, sehingga proses tersebut disimpan dalam LRU (Least RecentlyUsed) untuk menjaga agar proses yang baru saja dilihat oleh *user* akan dihapus paling akhir pada kondisi low memory.
5. *Empty Process*. Adalah proses yang tidak memegang suatu komponen yang sedang aktif. Alasan mengapa membiarkan proses semacam ini adalah untuk

melakukan *cache* sehingga proses startup menjadi lebih cepat apabila aplikasi tersebut akan dijalankan kembali. Sistem seringkali menghapus proses semacam ini untuk mencapai keseimbangan dari *system resource* antara *cached processes* dan *kernel caches*.

2.5 UML

Unified Modeling *Language* (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. Unified Modeling *Language* (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya.

UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut[11]. UML mulai diperkenalkan oleh *Object* Management Group, sebuah organisasi yang telah mengembangkan model, teknologi, dan standar OOP sejak tahun 1980-an.

Sekarang UML sudah mulai banyak digunakan oleh para praktisi OOP. UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM.UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi.

Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem.

2.5.1 Sejarah UML

Sampai era tahun 1990 puluhan metodologi pemodelan berorientasi objek telah bermunculan di dunia. Diantaranya adalah: metodologi booch, metodologi coad, metodologi OOSE, metodologi OMT, metodologi shlaer-mellor, metodologi wirfs-brock, dsb. Masa itu terkenal dengan masa perang metodologi (method war) dalam pendesainan berorientasi objek. Masing-masing metodologi membawa

notasi sendiri-sendiri, yang mengakibatkan timbul masalah baru apabila kita bekerjasama dengan kelompok/perusahaan lain yang menggunakan metodologi yang berlainan. Dimulai pada bulan Oktober 1994 Booch, Rumbaugh dan Jacobson, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan mempelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 *direlease* draft pertama dari UML (versi 0.8). Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh *Object* Management Group

2.5.2 Diagram UML

UML menyediakan 10 macam diagram untuk memodelkan aplikasi berorientasi objek , yaitu:

1. *Use case* Diagram untuk memodelkan proses bisnis.
2. *Conceptual Diagram* untuk memodelkan konsep-konsep yang ada di dalam aplikasi.
3. *Sequence diagram* untuk memodelkan pengiriman pesan (*message*) antar *object*s.
4. *Collaboration Diagram* untuk memodelkan interaksi antar *object*s.
5. *State Diagram* untuk memodelkan perilaku *object*s di dalam sistem.
6. *Activity diagram* untuk memodelkan perilaku *Use cases* dan *object*s di dalam system.
7. *Class diagram* untuk memodelkan struktur kelas.
8. *Object Diagram* untuk memodelkan struktur *object*.
9. *Component Diagram* untuk memodelkan komponen *object*.
10. *Deployment Diagram* untuk memodelkan distribusi aplikasi.

2.5.2.1 *Use case Diagram*

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *Use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-create sebuah daftar belanja, dan sebagainya.

Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. Sebuah *Use case* dapat meng-*include* fungsionalitas *Use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *Use case* yang di-*include* akan dipanggil setiap kali *Use case* yang meng-*include* dieksekusi secara normal. Sebuah *Use case* dapat di*include* oleh lebih dari satu *Use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang common.

Sebuah *Use case* juga dapat meng-extend *Use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *Use case* menunjukkan bahwa *Use case* yang satu merupakan spesialisasi dari yang lain.

2.5.2.2 *Activity diagram*

*Activity diagram*s menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state* diagram khusus, di mana sebagian besar *state* adalah action dan sebagian besar transisi di-trigger oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan *behaviour* internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *Use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *Use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas. Decision digunakan untuk menggambarkan *behaviour* pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal. *Activity diagram* dapat dibagi menjadi beberapa *Object* swimlane untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.

2.5.2.3 *Sequence diagram*

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *Use case*: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

2.5.2.4 *Class diagram*

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. *Class diagram* menunjukkan hubungan antar *class* dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan.

2.6 Bahasa Pemrograman

2.6.1 Java

Java adalah bahasa pemrograman serbaguna. Java dapat digunakan untuk membuat suatu program sebagaimana Anda membuatnya dengan bahasa seperti Pascal atau C++. Yang lebih menarik, Java juga mendukung sumber daya internet yang saat ini populer, yaitu World Wide Web atau yang sering disebut web. Java juga mendukung aplikasi klient/*server*, baik dalam jaringan local (LAN) maupun jaringan berskala luas (WAN)

Java dikembangkan oleh Sun Microsystems pada Agustus 1991, dengan nama semula Oak. Konon Oak adalah pohon semacam Jati yang terlihat dari jendela

tempat pembuatnya, James Gosling, bekerja. Ada yang mengatakan bahwa Oak adalah singkatan dari "*Object ApplicationKernel*", tetapi ada yang menyatakan hal itu muncul setelah nama Oak diberikan. Pada Januari 1995, karena nama Oak dianggap kurang komersial, maka diganti menjadi Java.

Dalam sejumlah literatur disebutkan bahwa Java merupakan hasil perpaduan sifat dari sejumlah bahasa pemrograman, yaitu C, C++, *Object*-C, SmallTalk, dan Common LISP. Selain itu Java juga dilengkapi dengan unsur keamanan. Yang tak kalah penting adalah bahwa Java menambahkan paradigma pemrograman yang sederhana. Jika telah mengenal C atau C++, yang mengandalkan pointer, Java justru meninggalkannya sehingga anda akan memperoleh kemudahan saat menggunakannya.

Program java bersifat tidak bergantung pada *platform*, artinya, Java dapat dijalankan pada sembarang komputer dan bahkan pada sembarang sistem operasi. Ketidakbergantungan pada *platform* sering dinyatakan dengan istilah portabilitas. Yang menarik, tingkat portabilitas java tidak hanya sebatas pada program sumber (*source code*), melainkan juga pada tingkat kode biner yang disebut *bytecode*. Dengan demikian bila telah mengkompilasi program java pada komputer berbasis operasi Windows, dapat juga menjalankan hasil kompilasi pada Macintosh secara langsung, tanpa perlu mengkompilasi ulang.

.

2.6.2 PHP

PHP (*Personal Home Page*) merupakan *Script* untuk pemrograman *Web server-side*, *Script* yang membuat dokumen HTML secara *on the fly*, dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan teks *editor* atau *editor* HTML. Dengan menggunakan PHP, maintanance suatu *situs web* menjadi lebih mudah. Proses *update* data dapat dilakukan dengan menggunakan aplikasi yang dibuat dengan menggunakan *Script* PHP. PHP/FI merupakan nama awal dari PHP, dibuat pertama kali oleh Rasmus Lerdoff. PHP awalnya merupakan program CGI yang dikhususkan untuk menerima *input* melalui *form* yang ditampilkan dalam *browser web*. *Software* ini disebarkan

dan dilisensikan sebagai perangkat lunak *Open source*. Adapun keunggulan yang dimiliki oleh PHP adalah:

1. *Life Cycle* yang sangat singkat, sehingga PHP selalu *up to date* mengikuti perkembangan teknologi internet.
2. *Cross Platform*, yakni PHP dapat dipakai di hampir semua *webserver* yang ada di pasaran (terutama Apache dan Microsoft IIS) dan dijalankan pada berbagai sistem operasi (Linux, Windows, FreeBSD).
3. PHP mendukung koneksi ke banyak *Database* baik yang gratis maupun komersil, seperti *MySQL*, *mSQL*, *Oracle*, *Microsoft SQL Server*, *Interbase*, dan banyak lagi.
4. PHP bersifat *open source* dan gratis.

2.6.3 SQL (*Structured Query Language*)

Menurut Alan Beaulieu, pada bukunya yang berjudul *Learning SQL* (2009), SQL adalah sebuah bahasa untuk mengambil, menambahkan, dan memanipulasi data dari sebuah basis data relasional. Bahasa SQL sendiri dibagi ke beberapa kategori pernyataan atau *statements*. Kategori pertama adalah pernyataan atau *statements* untuk membuat objek basis data, seperti tabel, *index*, *constraints*, dan sebagainya yang biasa disebut dengan *schema statements*. Kategori kedua merupakan pernyataan atau *statements* yang digunakan untuk membuat, memanipulasi, dan mengambil data yang tersimpan dalam basis data. Kategori kedua ini biasa disebut dengan SQL *data statements*. Biasanya, seseorang yang bekerja sebagai *programmer* atau penulis laporan hanya perlu menggunakan (atau hanya diberi akses) *data statements* saja, karena yang perlu dilakukan hanyalah berhubungan dengan data yang tersimpan pada sebuah basis data, bukan struktur basis data itu sendiri [2].

Contoh bahasa SQL yang banyak digunakan bisa dilihat pada tabel berikut.

Tabel 2-2 Contoh Contoh Structured Query Language

USE namadatabase
SELECT * FROM namatabel
DELETE FROM namatabel WHERE namakolom = parameter
UPDATE namatabel SET namakolom = parameter_baru

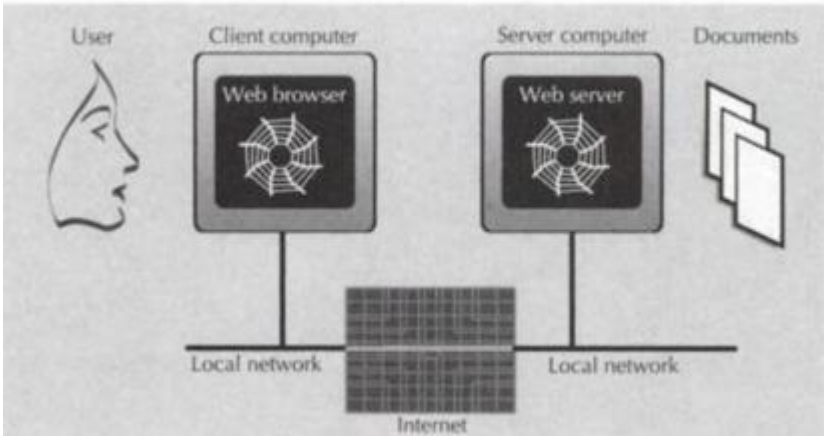
Bahasa SQL banyak diadopsi oleh beberapa aplikasi basis data relational, seperti MySQL, Microsoft SQL *Server*, PostgreSQL, dan SQLite.

2.7 Perangkat Lunak

Perangkat lunak adalah istilah yang digunakan untuk menggambarkan instruksi-instruksi yang memberitahu perangkat keras untuk melakukan suatu tugas sesuai dengan perintah [3]. Perangkat lunak yang digunakan dalam pembangunan sistem pada penelitian ini adalah *Web server* Apache, *Web Browser* Google Chrome, dan MySQL.

2.8 *Web server* Apache

Sebuah *Web server* adalah perangkat lunak yang menawarkan pelayanan yang bisa didapatkan oleh seluruh jaringan melalui protokol HTTP. Dengan adanya *Web server* memungkinkan sebuah komputer untuk menyediakan dokumen atau halaman web yang bisa diakses melalui *web browser*. Skema permintaan dokumen dari *web browser* ke *Web server* bisa dilihat pada gambar berikut.



Gambar 2-5 Skema *Web server*

Pada saat sekarang ini ada banyak jenis-jenis *Web server* yang tersedia, seperti nginx, Litespeed, Apache dan IIS. Salah satu yang banyak digunakan adalah Apache.

Apache merupakan salah satu jenis *Web server* yang didistribusikan di bawah lisensi *open source*. Apache mampu berjalan di berbagai macam sistem operasi yang berbasiskan UNIX dan Windows, seperti Linux, Windows XP, Windows 7, Windows 8.1, Windows 10. Berikut adalah keunggulan dari *Web server* Apache :

1. Apache mampu menangani berbagai macam kebutuhan web, mulai dari web pribadi yang kecil sampai web kelas *enterprise*.
2. Apache gratis dan *open source*, untuk bisa menggunakannya, pengguna hanya perlu mengunjungi halaman http://www.apache.org dan mengunduh tipe Apache yang sesuai dengan sistem operasi yang dipakai pengguna.

Komunitas *open source* Apache sangat banyak, sehingga mempercepat waktu perbaikan apabila terjadi *bugs* atau *error* pada aplikasi Apache tersebut. Penelitian ini menggunakan Apache versi 2.4.4.

2.8.1 MySQL

MySQL merupakan salah satu dari sekian banyak *Database server*. *Database server* merupakan sebuah perangkat lunak bertugas untuk menyediakan tempat penyimpanan basis data, serta melayani permintaan informasi dari basis data

tersebut. Beberapa jenis *Database server* lain diantaranya, PostgreSQL, Microsoft SQL *Server*, Oracle, dan banyak lagi.

MySQL adalah *Relational Database Management System* (RDBMS) atau sistem manajemen basis data relasional yang didistribusikan secara gratis di bawah lisensi GPL (General Public License). Lisensi GPL ini memungkinkan semua orang untuk bebas menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat komersial [14].

Sebagai salah satu *Database server*, MySQL harus mempunyai beberapa kemampuan, salah satunya harus menyediakan suatu sistem manajemen basis data yang dapat mengatur bagaimana menyimpan, menambah, mengakses dan melakukan transaksi transaksi pada sebuah basis data.

Perintah-perintah yang dijalankan dalam MySQL menggunakan bentuk SQL (*Structured Query Language*). SQL adalah sebuah bahasa komputer yang digunakan untuk menyimpan, memanipulasi, dan mengambil data yang tersimpan dalam basis data relational [15]. SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah.

Beberapa keunggulan atau keistimewaaan MySQL diantaranya :

1. Portabilitas

   MySQL dapat berjalan stabil pada berbagai sistem operasi, seperti Windows, Linux, FreeBSD, Mac Os *Server*, dan masih banyak lagi.
2. *Open source*

   MySQL didistribusikan secara *open source*, di bawah lisensi GPL sehingga dapat digunakan secara cuma-cuma.
3. *Multiuser*

   MySQL dapat digunakan oleh beberapa pengguna dalam waktu bersamaan tanpa mengalami masalah konflik
4. *Performance Tuning*

   MySQL mampu memproses lebih banyak *Query* SQL per satuan waktu.
5. Jenis Kolom

MySQL memiliki tipe kolom yang sangat kompleks, seperti integer, float, double, char, text, date, datetime, dan lain lain.

MySQL yang digunakan dalam pembangunan *prototype* pada penelitian ini adalah MySQL versi 5.5.32 (*Community Server*).

2.8.2 Android Studio

Android Studio adalah sebuah IDE (*Integrated Development Environment*) resmi untuk mengembangkan perangkat lunak di *platfrom Android*. [4] Aplikasi ini dibuat pada sebuah *teks editor* menggunakan Android Studio

2.8.3 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generated*) oleh komputer[12]. JSON terbuat dari dua struktur yaitu:

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 (*list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (list), atau urutan (*Sequence*).

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).

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).

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.

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.Angka, Format oktal dan heksadesimal tidak digunakan

2.8.4 XAMPP

XAMPP Kepanjangan dari XAMPP yaiu APACHE, PHP, MySQL dan PHPMyAdmin. XAMPP merupakan tool yang menyediakan paket perangkat lunak ke dalam satu buah paket. Dengan menginstall XAMPP maka tidak perlu lagi melakukan instalasi dan konfigurasi *Web server* APACHE, PHP dan MySQL secara manual. XAMPP akan menginstalasi dan mengkonfigurasikannya secara otomatis untuk anda atau auto konfigurasi

2.8.5 APACHE

APACHE merupakan salah satu jenis program yang bertujuan untuk mengirimkan kembali sintaks yang telah diberikan oleh *user* dan menampilkan kembali dalam bentuk dalam format hasilnya saja. Program APACHE ini hanya terdapat pada sebuah *server* yang terdapat pada ISP (Internet *Service* Provider) atau di Indonesia disebut Jasa Penyelenggara Internet. Saat ini program yang paling

banyak terdapat pada *server-server* di dunia adalah APACHE. Hal ini disebabkan karena beberapa keunggulan yang diberikan. Keunggulan-keunggulan tersebut antara lain :

1. Memiliki kemapuan pembacaan yang tinggi dan mudah disesuaikan
2. Memiliki tingkat keamanan yang tinggi. Hal ini dibutuhkan bagi para penyimpan data yang memiliki tingkat kerahasian tertentu.
3. Dapat dijalankan untuk berbagai macam operasi sistem.

APACHE merupakan suatu nama yang diambil dari istilah "A PAtCHy" *server* yang dapat diartikan dengan mengambil suatu patch atau bagian dari program yang diberikan. APACHE sudah berkembang sejak versi pertamanya. Sampai saat ditulisnya artikel ini versi terakhirnya yang ada yaitu APACHE ver 2.4.23 APACHE bersifat *open source*, artinya setiap orang boleh menggunakannya, mengambil dan bahkan mengubah kode programnya.

Tugas utama APACHE adalah menghasilkan halaman web yang benar kepada peminta, berdasarkan kode PHP yang dituliskan oleh pembuat halaman web. Jika diperlukan juga berdasarkan kode PHP yang dituliskan, maka dapat saja suatu *Database* diakses terlebih dahulu (misalnya dalam MySQL) untuk mendukung halaman web yang dihasilkan.

2.8.6 CSS

CSS merupakan bentuk kependekan dari *Cascading Style Sheets*, istilah ini apabila diartikan memiliki arti "hamparan pasir melata". Persepsi baru bahwa dokumen ini merupakan bentuk *Script* yang sangat membantu untuk membuat desain lebih menarik. CSS bukan merupakan bahasa pemograman, akan tetapi dapat disebut dengan spesial yang berguna untuk melakukan pengaturan pada setiap komponen (bagian-bagian seperti tabel, form, dan lain – lain) web. Inti dari dokumen CSS adalah Tag HTML, di dalam CSS tag HTML tidak lagi disebut sebagai Tag akan tetapi akan disebut sebagai Selector.

Fungsi terpenting dari dokumen CSS adalah melakukan pengaturan halaman web dengan sistem *Style*, karena dokumen CSS merupakan bentuk turunan

dari elemen HTML (perintah html) maka dokumen ini merupakan bentuk kode yang embeded dengan Tag HTML, dalam penggunaannya kedua objek ini akan sangat berhubungan dan seakan tidak pernah terpisahkan.

2.8.7 Ionic

Ionic *Framework* adalah kerangka pembangunan aplikasi *mobile* HTML5 yang ditargetkan untuk membangun aplikasi *mobile* hybrid. Aplikasi hybrid pada dasarnya adalah *website* yang berjalan dalam browser sebuah aplikasi yang memiliki akses ke lapisan *platform native*. Apps Hybrid memiliki banyak manfaat lebih dari aplikasi asli murni, khususnya dalam hal mendukung *platform*, kecepatan dalam pembuatannya, dan akses ke kode pihak ke-3. Ionic adalah kerangka front-end *user interface* yang menangani semua tampilan dan nuansa serta interaksi *user interface* agar aplikasi menarik. Seperti jenis "*Bootstrap for native*," tetapi dengan dukungan untuk berbagai komponen umum *nativemobile*, animasi, dan desain yang bagus. Tidak seperti kerangka responsif, Ionic dilengkapi dengan elemen *user interface* ponsel yang sangat bergaya *native*dan layout yang dapat diperoleh dengan SDK *native*pada iOS atau Android, namun, *user interface* ditampilkan dalam bentuk sebuah web. Ionic menggunakan Cordova untuk dapat bekerja sebagai *native*app. Ionic *Framework* dibangun untuk menghasilkan performa terbaik pada perangkat *mobile*. Ionic *Framework* menggunakan AngularJS dalam rangka menciptakan *Framework* yang paling cocok untuk pengembangan aplikasi *mobile[12]*.

2.8.8 Laravel *Framework*

Laravel merupakan salah satu PHP *Framework* yang digunakan untuk memudahkan web developer dalam membuat dan mengembangkan suatu *website*. Laravel membutuhkan composer yang merupakan dependency management dalam bahasa pemrograman PHP. Versi terbaru dari laravel adalah versi 5.3. Laravel menggunakan arsitektur MVC (*Model View Controller*) yaitu salah satu arsitektur perangkat lunak yang memisahkan aplikasi menjadi tiga bagian yaitu model yang memiliki peran untuk berinteraksi dengan *Database* yang digunakan, view untuk

menampilkan interface pengguna, dan *controller* berperan untuk menampilkan view atau melakukan update pada model sesuai dengan permintaan pengguna[13]