BAB 2

TINJAUAN PUSTAKA

2.1. Android

2.1.1. *Platform* Android

Android adalah sistem operasi untuk telepon seluler yang berbasis Linux. Android menyediakan platform terbuka bagi para pengembang buat menciptakan aplikasi mereka sendiri untuk digunakan oleh bermacam peranti bergerak. Awalnya, Google Inc. membeli Android Inc., pendatang baru yang membuat peranti lunak untuk ponsel. Kemudian untuk mengembangkan Android, dibentuklah O*pen Handset Alliance*, konsorsium dari 34 perusahaan peranti keras, peranti lunak, dan telekomunikasi, termasuk Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, dan Nvidia. (Safaat, 2010).

Pada saat perilisan perdana Android, 5 November 2007, Android bersama *Open Handset Alliance* menyatakan mendukung pengembangan standar terbuka pada perangkat seluler. Di lain pihak, Google merilis kode-kode Android di bawah lisensi Apache, sebuah lisensi perangkat lunak dan standar terbuka perangkat seluler.

Di dunia ini terdapat dua jenis distributor sistem operasi Android. Pertama yang mendapat dukungan penuh dari Google atau Google *Mail Services* (GMS) dan kedua adalah yang benar-benar bebas distribusinya tanpa dukungan langsung Google atau dikenal sebagai *Open Handset Distribution* (OHD).

2.1.2. Versi Android

Android telah melewati cukup banyak pembaruan sejak pertama kali dirilis. Berikut merupakan versi-versi dari Android beserta *codename*-nya.

1. Android 1.0 (2008)

   Sistem operasi ini bisa dibilang sebagai Android bayi yang masih sangat sederhana. Pengguna diajak beradaptasi dengan peradaban baru dalam menjajal sebuah perangkat telekomunikasi mobile.

Di versi ini Google untuk pertama kalinya memperkenalkan mekanisme pengecekan notifikasi yang kerap diistilahkan *pull-down notification* tersebut. Selain notifikasi, dua komponen pada Android 1.0 yang masih digunakan hingga kini adalah *widget* aplikasi, serta toko aplikasi Google *Play Store* yang kala itu masih bertajuk "*Market*".

Sistem operasi ini juga menyatukan layanan Gmail. Dua aplikasi tersebut, Gmail dan *Market*, menjadi layanan bawaan paling purba yang dipatrikan Google pada Android 1.0.

2. Android 1.5 *Cupcake* (2009)

   Ini adalah debut versi Android yang menggunakan nama kudapan manis. Tradisi tersebut dipertahankan hingga sekarang.

   Pada *Cupcake*, Google juga memperkenalkan SDK *widget* untuk *developer* pihak ketiga. Gunanya agar aplikasi pihak ketiga bisa memiliki *widget* sendiri layaknya aplikasi bawaan Google.

   Dua pembaruan signifikan pada *Cupcake* juga meliputi kemampuan perekaman video dengan kamera ponsel, serta kemampuan *keyboard* layar sentuh.

3. Android 1.6 *Donut* (2009)

   Pada versi ini, Google mengumumkan bahwa Android bisa digunakan untuk perangkat *mobile* dengan ukuran layar berapa saja.

   Android *Donut* juga memunculkan kolom pencarian pada antarmuka ponsel. Pengguna bisa mencari informasi di internet, file lokal, kontak, dan apa saja secara lebih cepat dengan kolom tersebut.

4. Android 2.0 *Éclair* (2009)

   Eclair menjadi Android pertama yang menghadirkan layanan navigasi Google *Maps*. Sistem tersebut menjadi awal mula era GPS yang sekarang bukan cuma ada di ponsel, tapi juga di mobil-mobil *modern*. *Eclair* juga menjadi Android pertama yang mendukung HTML5 pada peramban sehingga bisa memutar video. Kemampuan membuka layar alias *unlock-screen* dengan mekanisme menyapu atau *swipe* juga diperkenalkan pada *Eclair*.

5. Android 2.2 *Froyo* (2010)

Dari segi tampilan, Android *Froyo* memungkinkan lima panel layar depan alias *home screen*. Sebelumnya, batas panel cuma sampai tiga saja. *Froyo* juga menambah pilihan keamanan penguncian bagi pengguna. Dari yang sebelumnya cuma penguncian pola (*pattern lock*), belakangan dilengkapi dengan opsi penguncian PIN atau PIN *lock*.

6. Android 2.3 *Gingerbread* (2010)

   Google membangun versi *Gingerbread* dengan kemampuan kamera depan untuk membidik foto mandiri. Pada versi ini, pengguna juga bisa melihat desain ulang antarmuka yang cukup signifikan.

   Selain itu, dari segi fungsi, *Gingerbread* memungkinkan pengguna memencet *keyboard* virtual secara berbarengan alias *multitouch*. Kemampuan ini dipertahankan hingga sekarang dengan berbagai peningkatan kinerja.

7. Android 3.0 *Honeycomb* (2011)

   Sistem operasi ini mendukung kemampuan tombol *virtual* untuk *home*, *back*, dan *menu*, untuk pertama kalinya. Sasarannya pun lebih ke perangkat tablet ketimbang *smartphone*.

8. Android 4.0 *Ice Cream Sandwich* (2011)

   Versi ini memboyong kemampuan pada *Honeycomb* tapi lebih menyasar *smartphone*. Beberapa pembaruan fitur lainnya mencakup kemampuan membuka layar menggunakan wajah (*face unlock*), analisa penggunaan data internet, serta paket aplikasi bawaan dari *vendor* yang mencakup kalendar, *mail*, kalkulator, dan lainnya.

9. Android 4.1 *Jelly Bean* (2012)

   Merupakan versi Android yang membawa pembaruan cukup signifikan setelah beberapa kali *update* yang dilakukan Google hanya membawa perbedaan minor.

   Salah satunya, *Jelly Bean* memungkinkan pengguna menggulir (*scroll*) cepat *home screen* ke bawah untuk melihat kumpulan informasi penting, seperti agenda, *email*, dan laporan cuaca. Sebelumnya, pengguliran ke bawah cuma memperlihatkan notifikasi aplikasi.

10. Android 4.4 *Kitkat* (2013)

Pada *KitKat*, Google menghadirkan perintah pencarian menggunakan suara atau disebut "*Ok*, *Google*". Di saat bersamaan, Google juga meluncurkan aplikasi pesan singkat *Hangouts* untuk pertama kalinya.

11. Android 5.0 *Lollipop* (2014)

    Pembaruan yang mencolok pada *Lollipop* tampak dari sisi desainnya yang diperhalus dan disesuaikan dengan zaman. Selain itu, fitur-fitur yang sudah hadir pada Android sebelumnya ditingkatkan.

12. Android 6.0 *Marshmallow* (2015)

    Menu aplikasi pada Android *Marshmallow* benar-benar dibuat baru. Desainnya membuat pengguna merasa naik kelas dari versi sebelumnya karena lebih dinamis. Selain itu, ada juga fitur *memory manager* yang memungkinkan pengguna mengecek penggunaan memori pada tiap aplikasi. Rentan waktu pengecekannya bisa disetel dari tiga jam yang lalu hingga 24 jam sebelumnya.

    Pembaruan kedua ditilik dari pengaturan *volume*. Pada *Marshmallow*, pengguna bisa mengontrol *volume* yang berbeda-beda pada panggilan, *media*, dan *alarm*.

    Keamanan juga mendapat peningkatan pada versi ini. Google memungkinkan *vendor* menyematkan sensor pemindai sidik jari karena sudah didukung *Marshmallow*.

13. Android 7.0 *Nougat* (2016)

    Nougat adalah versi Android termutakhir yang baru diperkenalkan pada ajang kumpul *developer* Google 2/O, pertengahan 2016 ini. Beberapa lama setelahnya, Google menghadirkan Nougat secara resmi untuk publik.

    Pembaruan paling mendasar pada versi *Nougat* adalah kehadiran *Google Assistant* yang menggantikan *Google Now*. Asisten digital tersebut lebih bisa diandalkan untuk menjalankan berbagai fungsi.

    Fitur-fitur baru lainnya mencakup layar *split-screen* saat dipakai *multitasking*, serta fitur *Doze* yang telah dikenalkan di versi Android *Marshmallow* namun telah ditingkatkan. Android *Nougat* juga memiliki dukungan terhadap platform *virtual reality* terbaru Google.

2.1.3. Fitur-Fitur Android

Pada Android terdapat beberapa fitur yang mendukung kerja dari Android sendiri, antara lain :

1. *Application framework*
2. *Dalvik virtual machine* (DVM), dioptimalkan untuk perangkat *mobile.*
3. *Browser* yang terintegrasi
4. Grafis yang dioptimasi, didukung oleh *library* grafis 2D yang *custom*; grafis 3D berdasarkan spesifikasi OpenGL ES 1.0.
5. SQLite, untuk penyimpanan data terstruktur.
6. Media pendukung, untuk *audio, video* dan gambar dalam beberapa format (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
7. GSM *Telephony* (bergantung pada *hardware* yang digunakan).
8. *Bluetooth, EDGE, 3G* dan *WiFi* (bergantung pada *hardware* yang digunakan).
9. *Camera, GPS, compass* dan *accelerometer* (bergantung pada *hardware* yang digunakan).
10. Lingkungan pengembang yang lengkap, termasuk perangkat emulator, alat untuk *debug,* memori dan kinerja profil, serta *plugin* untuk Eclipse IDE.

    (Sumber: http://developer.android.com/about/versions/index.html)

2.1.4. Arsitektur Android

Menurut Lee (2011), sistem operasi Android dibagi menjadi lima bagian di dalam empat lapisan utama, yaitu :

1. *Linux Kernel*

   Merupakan *kernel* yang menjadi dasar Android, di mana terdapat seluruh perangkat *driver* tingkat rendah untuk berbagai komponen perangkat keras pada perangkat Android.

2. *Libraries*

   Bagian ini berisi seluruh kode yang menyediakan fitur utama pada sistem operasi Android. Contohnya yaitu SQLite library menyediakan dukungan database sehingga sebuah aplikasi dapat menggunakannya sebagai penyimpanan data. WebKit library menyediakan fungsi untuk web browsing.

3. *Android Runtime*

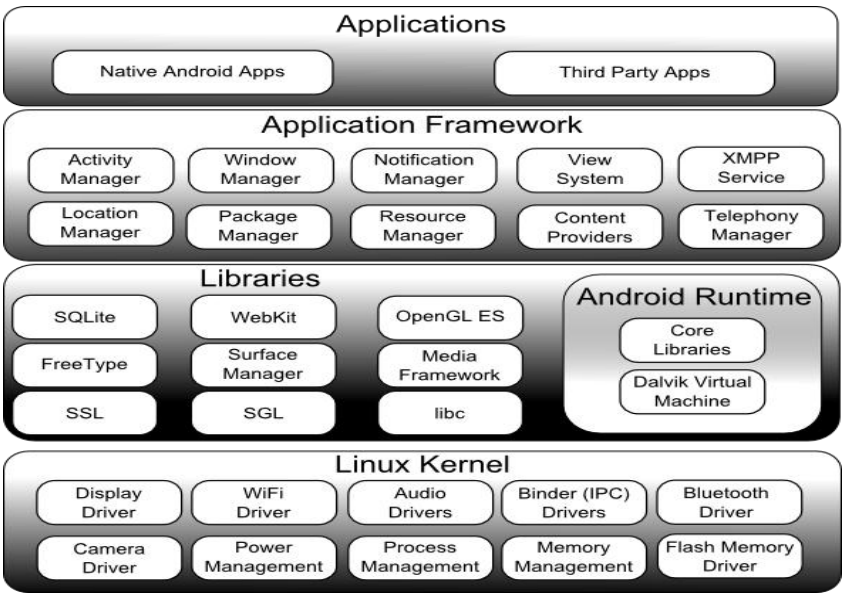
Android Runtime menyediakan sekumpulan core libraries yang memungkinkan developer untuk membuat aplikasi Android menggunakan bahasa pemrograman Java. Android Runtime juga mencakup Dalvik virtual machine yang memungkinkan setiap aplikasi Android untuk berjalan dalam prosesnya sendiri. Dalvik merupakan mesin virtual yang dirancang khusus untuk Android dan dioptimalkan untuk battery-powered perangkat mobile dengan memori yang terbatas dan CPU.

4. *Application Framework*

Bagian ini memperlihatkan berbagai kemampuan sistem operasi Android kepada pengembang aplikasi sehingga dapat menggunakannya di dalam aplikasi yang dikembangkan.

5. *Application*

Pada lapisan teratas ini akan ditemukan aplikasi yang disertakan bersama perangkat Android, seperti telepon, kontak, browser, dan lain-lain, serta aplikasi yang di-download dan install dari Android Market. Aplikasi yang dibuat berada pada lapisan ini.



Gambar 2.1 Arsitektur Android
(Sumber : http://developer.android.com/about/versions/index.html)

2.2. HTTP

Protokol HTTP merupakan sebuah protokol jaringan pada level aplikasi yang ringan dan cepat dibutuhkan untuk system informasi terdistribusi, kolaboratif, dan hypermedia. HTTP telah telah digunakan di jaringan seluruh dunia sejak 1990 (Bernesrs-Lee, et al., 1996). Pada penelitian ini protokol HTTP digunakan untuk aplikasi mobile dan web administrator sebagai protokol yang dapat mendistribusikan data yang bersumber dari database atau server.

2.3. Library

Library merupakan kumpulan resource program yang sering digunakan para developer untuk mengembangkan software aplikasi. *Library* dapat berisi konfigurasi data, dokumentasi, data penunjang, maupun class (Chen, 2015). Salah satu dari *library* adalah *library HTTP*. *Library HTTP* adalah *library* yang digunakan sebagi koneksi antara *client* dan *server.* Adapun contoh dari beberapa *library HTTP* adalah:

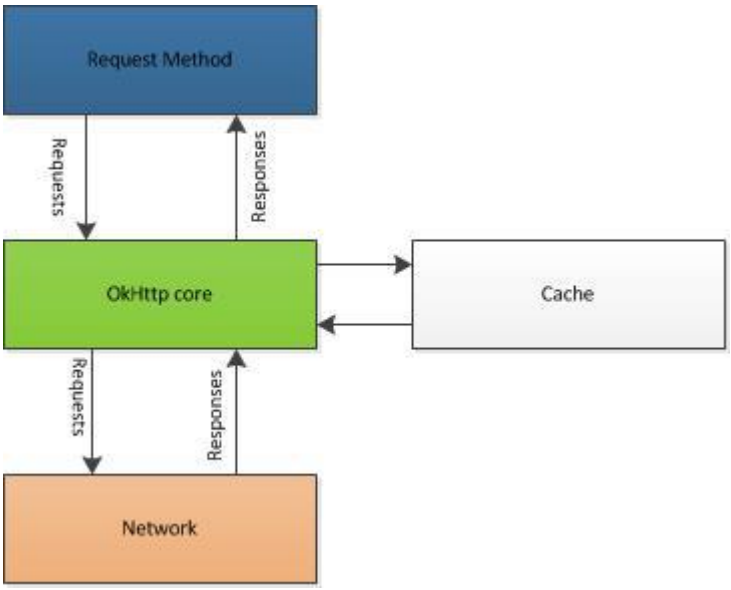
2.3.1. Okhttp

OkHttp merupakan suatu *HTTP client* untuk android dan aplikasi *java* buatan Square. Ok-http adalah library third-party yang dikembangkan oleh Square yang dapat digunakan untuk mengirim dan menerima HTTP-based network request. Library OkHTTP dibuat dari Okio library, yang dikembangkan dengna tujuan agar lebih efisien dalam mengirim dan menerima data dibandingkan dengan standar Java I/O libary.

OkHttp siap ketika jaringan terdapat gangguan. Secara perlahan *OkHttp* akan pulih dari masalah koneksi. Jika suatu layanan terdapat beberapa *IP address*, *OkHttp* akan memilih alamat alternatif jika koneksi pertama gagal. Ini dibutuhkan untuk *IPv4 + IPv6* dan untuk layanan dengan *data center* yang besar. *OkHttp* memulai dengan fitur *TLS* yang baru (*SNI*, *ALPN*) dan jika gagal akan kembali ke *TLS*1.0. *Request/response API* telah dirancang dengan kuat sekaligus mendukung *synchronous blocking call* dan *async call with callback* (OkHttp, 2016).

Fitur Okhttp secara *default* (Okhttp, 2016):

1. *HTTP/2* mendukung semua *request* pada host yang sama untuk membagikan sebuah socket.
2. *Connection polling* mengurangi permintaan latency (jika *HTTP/2* tidak ada).
3. *Response caching* menghindari jaringan yang melakukan permintaan berulang.



Gambar 2.2. Workflow Okhttp

2.3.2. Retrofit

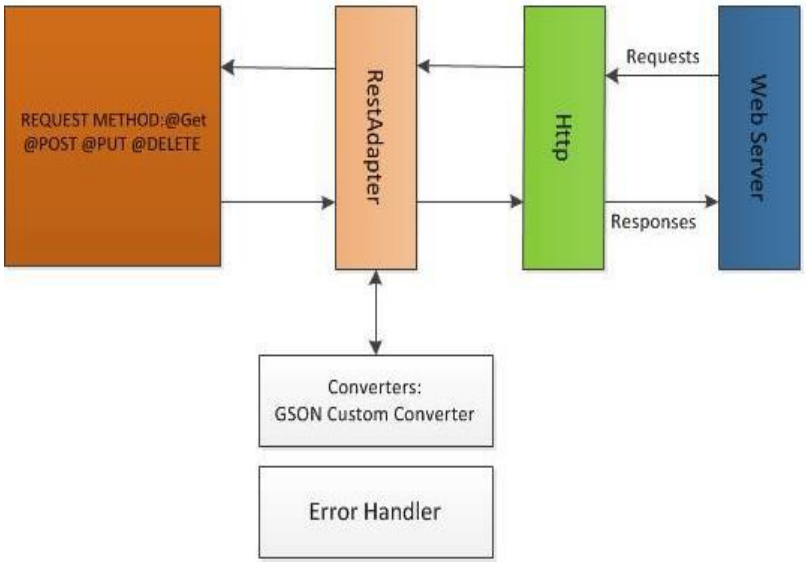
Retrofit merupakan REST client library yang aman untuk android dan java. Retrofit dibuat oleh square. Retrofit menyediakan cara yang aman untuk autentikasi dan interaksi dengan berbagai API lainnya. Sehingga memungkinkan pengiriman permintaan jaringan dengan Okhttp. Retrofit mengambil data JSON dan XML Dario web API dan saat data diterima akan langsung dikonversi atau diubah ke Plain Old Java Object (POJO). Sehingga harus ditentukan setiap class yang dipakai saat response diterima (Cosepath, 2015).

Retrofit juga bekerja dengan REST API menggunakan implementasi java interface, yang dapat dihasilkan dengan bantuan *RestAdapter*.

Implementasi dalam hal ini bertindak sebagai *local Instance* dari layanan dan setiap panggilan sesuai dengan permintaan HTTP (Maskov, 2015).

Retrofit menggunakan anotasi untuk mendeskripsikan HTTP request (Retrofit, 2016):

1. Penggantian parameter *URL* dan dukungan parameter *query.*
2. Pengubah objek ke *request body* contohnya *JSON, protocol buffer.*
3. Mengubah file.
4. *Request body* dalam part yang banyak.



Gambar 2.3. Workflow Retrofit

2.3.3. Volley

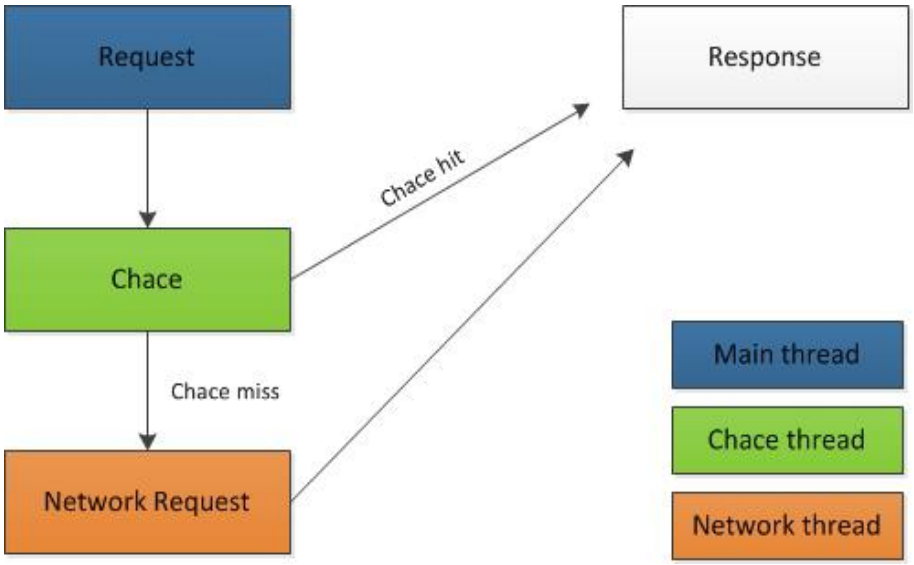
Volley Library merupakan merupakan produk yang diperkenalkan oleh Google untuk mempermudah pertukaran data tanpa harus membuat deretan kode yang sangat panjang. Secara *default* volley menggunakan metode singkronisasi.

Volley Library memiliki fitur seperti melakukan sebuah *request quering and prioritization* (Mengutamakan prioritas dalam sebuah antrian), Sangat efektif untuk melakukan *chace* dan efesiensi penyimpanan (memory), dapat melakukan perubahan *class* sesuai dengan kebutuhan, serta mampu melakukan pembatalan dalam sebuah *request*.

Adapun fitur yang ditawarkan Volley library adalah sebagai berikut:

1. Melakukan sebuah *request queuing and prioritization* (Mengutamakan prioritas dalam sebuah antrian).
2. Sangat efektif untuk melakukan *chace* dan efesiensi penyimapan (memory).
3. Dapat melakukan perubahan *class* sesuai dengan kebutuhan.
4. Dapat melakukan pembatalan dalam sebuah *request.*

Dari beberapa fitur diatas, ada kata yang menjadi point yaitu *"request"* Jadi Volley ini sangat baik digunakan untuk melakukan permintaan data ke server melalui web service dengan memperhatikan antrian dalam permintaan data data skala prioritas.



Gambar 2.4. Workflow Volley

2.4. JSON

JSON (*JavaScript Object Notation*) 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 pemprograman apapun karena menggunakan gaya bahasa yang umum digunakan oleh *programmer* keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dll. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data (Ecma International, 2013).

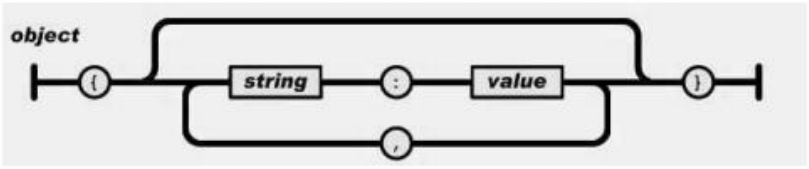
JSON terbuat dari dua struktur:

- 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*.
- 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 modern mendukung struktur data ini dalam bentuk yang sama maupun berlainan. Hal ini pantas disebut demikian karena format data mudah dipertukarkan dengan bahasa-bahasa pemprograman yang juga berdasarkan pada struktur data ini.

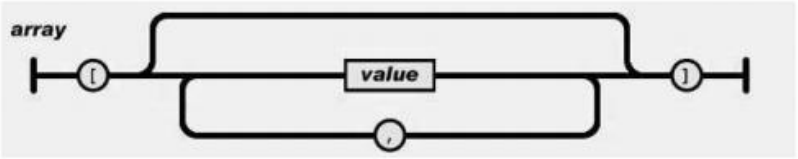
JSON menggunakan bentuk sebagai berikut :

- Objek merupakan sepasang nama/nilai yang tidak terurutkan.



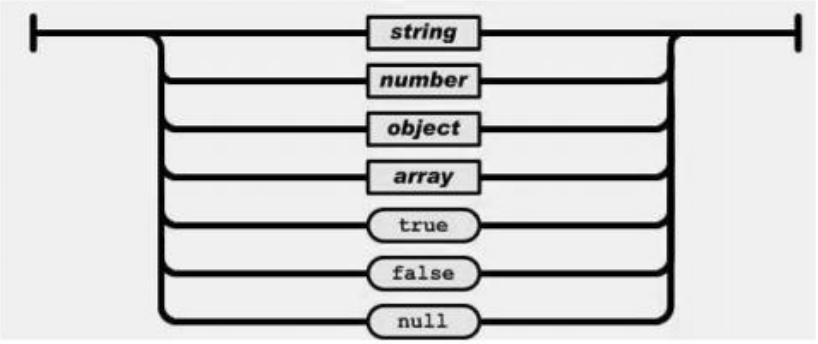
Gambar 2.5. Objek pada JSON

- Ray merupakan kumpulan nilai yang terurutkan.



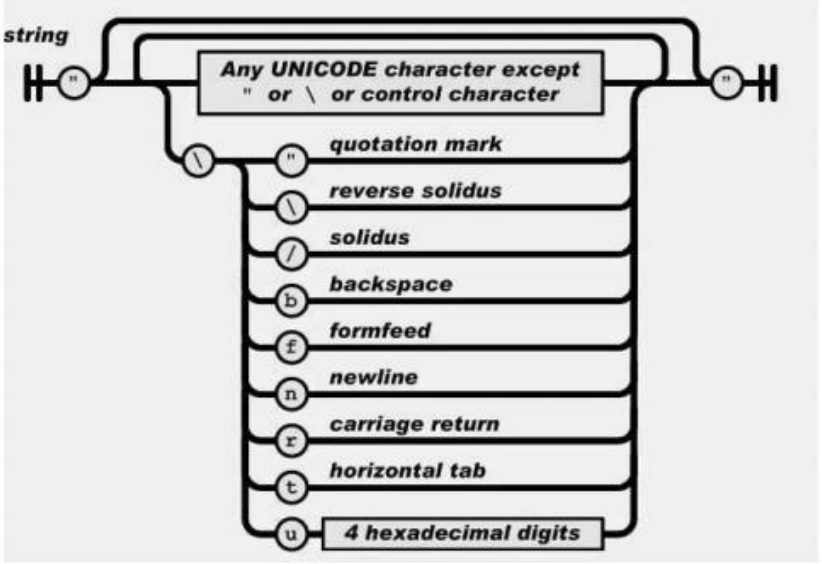
Gambar 2.6. Array pada JSON

- Nilai dapat berupa salah *string*, angka, *true*, *false*, *null*,objek, atau juga dapat berupa sebuah larik.



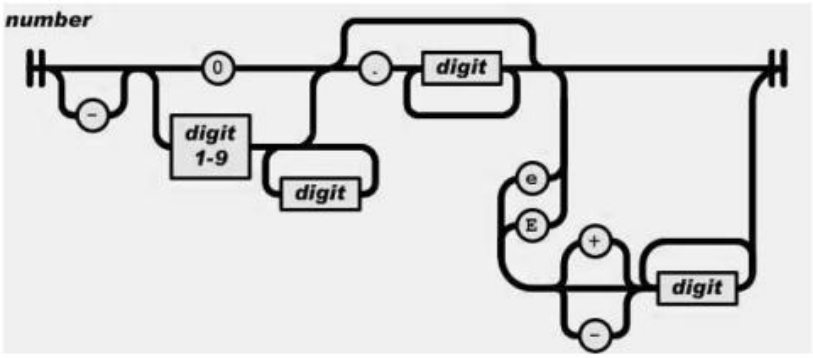
Gambar 2.7. Nilai pada JSON

- *String* terdiri dari kumpulan nol atau lebih karakter *Unicode* dimana menggunakan *backslash* (\) *escape* untuk membentuk karakter khusus.



Gambar 2.8. *String* pada JSON

- Angka yang ada di JSON mirip dengan angka yang di C dan Java akan tetapi tidak angka yang pada JSON tidak memiliki format oktal dan heksadesimal.



Gambar 2.9. Angka pada JSON

Contoh penulisan JSON :

```
{
  "addressbook" : {
     "contact":[
     {
        "name":"Wira Setiawan",
        "address":"bandung"
     },
     {
        "name":"John Doe",
        "address":"Los Angeles"
     }
     ]
  }
}
```

2.5. XML

XML merupakan singkatan dari eXtensible Markup Language. Bahasa markup adalah sekumpulan aturan-aturan yang mendefinisikan suatu sintaks yang digunakan untuk menjelaskan, dan mendeskripsikan teks atau data dalam sebuah dokumen melalui penggunaan tag. XML adalah sebuah bahasa markup yang digunakan untuk mengolah meta data (informasi tentang data) yang menggambarkan struktur dan maksud atau tujuan data yang

terdapat dalam dokumen XML, namun bukan menggambarkan format tampilan data tersebut. *(http://www.xml.com)*

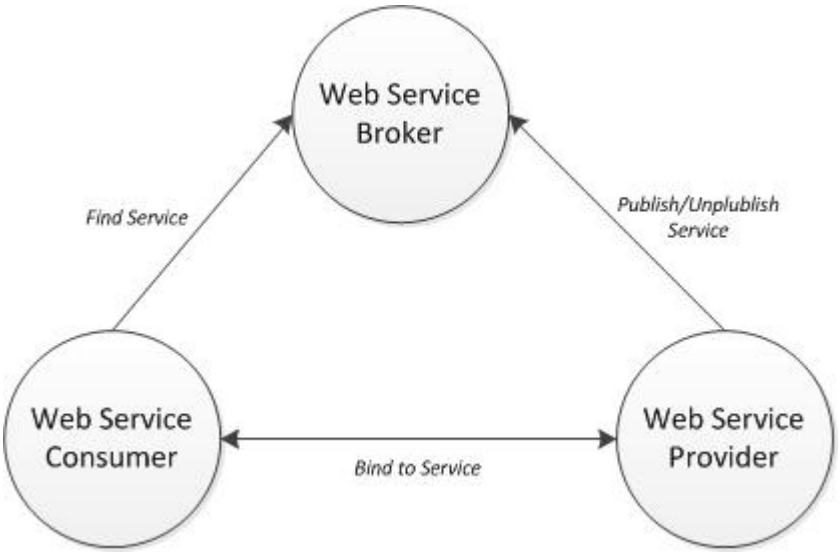
Contoh Penulisan XML

```
<?xml version="1.0" encoding="UTF-8" ?>
    <addressbook>
        <contact>
            <name>Wira Setiawan</name>
            <address>bandung</address>
        </contact>
        <contact>
            <name>John Doe</name>
            <address>Los Angeles</address>
        </contact>
    </addressbook>
```

2.6. Web Service

Web service menyediakan standar operasi antara aplikasi perangkat lunak yang berbeda dan berjalan pada platform atau framework yang berbeda (Lee, 2014). Web service adalah suatu standar yang mengintegrasikan aplikasi berbasis web dengan menghubungkan dan berbagi proses bisnis di seluruh jaringan yang memiliki aplikasi berbeda vendor, bahasa, dan platform (Al-Fedaghi, 2011).

Struktur web service memiliki tiga komponen yaitu web service broker, web service consumer, dan web service provider (Lee, 2014). Struktur web service ditunjukkan pada Gambar:



Gambar 2.10. Struktur web service

Penjelasan struktur web service pada gambar x.x adalah sebagai berikut:

1. Web service broker sebagai lokasi utama yang mendeskripsikan layanan yang telah terdaftar dan sebagai tempat web service consumer mencari layanan yang diinginkan.
2. Web Service Provider sebagai penyedia layanan dan mem-publish maupun unpublish spesifikasi layanan ke web service broker. Serta mengirimkan hasil pencarian ke web service consumer saat terjadi permintaan.
3. Web Service Consumer sebagai konsumen yang mencari layanan yang diinginkan pada web service broker. Selanjutnya melakukan proses binding ke web service provider untuk mendapatkan layanan.

Pada awalnya implementasi webs service menggunakan SOAP (Simple Object Access Protocol) dan WSDL (Web Service Description Language). SOAP adalah suatu standar protokol yang saling beroprasi untuk menghubungkan aplikasi. Sedangkan WSDL adalah suatu standar interface yang digunakan untuk mengakses aplikasi. Dalam beberapa tahun terakhir, terdapat suatu pendekatan alternatif menggunakan REST (Representational State Transfer). REST telah banyak digunakan dalam aplikasi berskala internet.

2.6.1. SOAP

SOAP (Simple Object Access Protocol) merupakan suatu protocol berbasis yang digunakan untuk pertukaran informasi antar aplikasi. Awalnya *SOAP* dikembangkan oleh *Microsoft* untuk menggantikan metode tradisional seperti *DCOM* dan *COBRA*. Keuntungan uatama dari penggunaan SOAP adalah pertukaran informasi antar aplikasi yang berjalan pada *platform* yang berbeda dan bahasa pemograman yang berbeda pula. Selain itu *SOAP* memungkinkan pola pertukaram *rich message* mulai dari tradisional *request-and-response* ke korelasi pesan *broadcasting* (Ramanathan, 2014).

Dokumen *SOAP* terdiri dari elemen teratas *XML* yang disebut *envelope*, yang terdiri dari suatu *header* dan *body*. *Header SOAP* bersifat fleksibel untuk informasi infrastruktur *message-layer* yang digunakan untuk tujuan *roating* dan konfigurasi *QoS (Quality of Service). Body SOAP* berisi *payload* dari pesan (Pautasso, et al., 2008).

2.6.2. REST

REST (Representational State Transfer) merupakan suatu model arsitektur yang terdiri dari beberapa *resource*. Setiap *resource* pada *REST* diidentifikasi dengan *URI (Uniform Resource Identifier)*. Dasar ide dari *REST* adalah menggunakan mekanisme *HTTP* untuk menghubungkan aplikasi daripada menggunakan mekanisme yang kompleks seperti *COBRA, RPC, dan SOAP. REST* focus pada interaksi *resource* dan perubahan pada *state* daripada fokus pada pengiriman dan penerimaan *message* seperti pada *SOAP* (Ramanathan, 2014).

Model arsitektur REST terdiri dari 4 prinsip utama yaitu (Pautasso, et al., 2008) :

1. Identifikasi *resource* melalui *URI*. Suatu *RESTful* web *service* terdiri dari sekumpulan *resource* yang mengidentifikasi target dari interaksi dengan *client*. *Resource* diidentifikasi melalui *URI* yang menyediakan ruang pengalaman untuk *resource* dan pencarian *service*.
2. *Uniform interface. Resource* dimanipulasi menggunakan empat operasi yaitu *PUT, GET, POST,* dan *DELETE. PUT* untuk membuat suatu *resource* baru dan dapat dihapus menggunakan perintah

DELETE. GET untuk mendapatkan *state* terbaru dari suatu *resource* pada beberapa representasi. Sedangkan *POST* utnutk mengirimkan state baru ke suatu *resource.*

3. Pesan *self-descriptive. Resource* dipisahkan dari representasinya sehingga konten dapat diakses dalam beberapa bentuk *seperti HTML, XML, plain text, PDF, JPG,* dan lainnya. *Meta-data* tentang *resource* ada dan digunakan seperti mengatur *caching*, mendeteksi transmisi yang *error*, mengatur sesuai format representasi, dan melakukan autentikasi atau akses kontrol.
4. Interaksi *stateful* melalui *hyperlink*. Setiap interaksi dengan suatu *resource* bersifat *stateless* misalnya permintaan pesan bersifat lengkap. Interaksi *stateful* berdasar dari konsep pengiriman *state* secara eksplisit. Beberapa teknik yang ada untuk bertukar *state* contohnya penulisan ulang *URI, cookies,* dan kolom formulir tersembunyi. Suatu *state* dapat diletakkan dalam *response* message untuk menunjukkan *state* yang valid dalam interaksi.