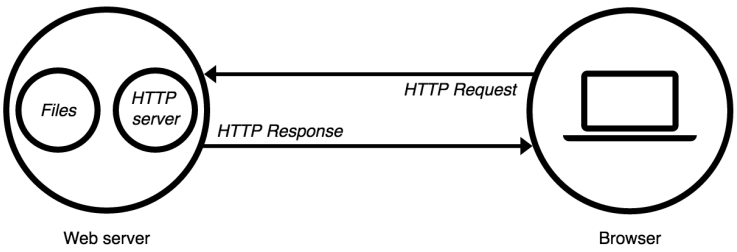
BAB II
LANDASAN TEORI

2.1 Web Server

Web server merujuk pada perangkat keras dan perangkat lunak yang berkolaborasi untuk memberikan layanan berbasis data dan berfungsi menerima permintaan dari HTTP (*Hyper Teks Transfer Protocol*) atau HTTPS (*Hyper Teks Transfer Protocol Secure*) pada klien yang menggunakan web *browser* dan mengirimkannya kembali yang hasilnya dalam bentuk beberapa halaman web dan pada umumnya akan berbentuk dokumen HTML (https://developer.mozilla.org, 2016). Dari sisi *hardware* web *server* bertugas menyipan berkas-berkas yang diperlukan oleh aplikasi web seperti dokumen HTML statis, gambar, berkas CSS, berkas JavaScript, berkas PHP, dll. Sementara dari sisi *software* web *server* mengontrol bagaimana pengguna dapat mengakses berkas yang disimpan melalui sebuah protokol HTTP. Server HTTP ini sendiri merupakan software tertentu yang dapat mengenali URL (*Universal Resource Locator*) / alamat web dan protokol HTTP yakni protokol yang digunakan *browser* untuk menampilkan halaman web.

Secara sederhana ketika pengguna mengakses suatu alamat menggunakan *browser* dan berhasil menjangkau web *server*, maka *server* HTTP akan merespon permintaan *user* dan mengirimkannya kembali melalui protokol HTTP. Skema komunikasi yang berjalan dapat dilihat pada Gambar 2.1.

Terdapat 2 jenis web *server*, yakni statis dan dinamis. *Web server* yang statis yang memberikan berkas yang diminta oleh *browser* seperti yang ada tanpa proses lebih lanjut. *Web server* yang dinamis memiliki perangkat lunak tambahan yang memproses berkas yang diminta *browser*. Perangkat lunak tambahan ini sering disebut dengan *application server*.



Gambar 2.1 Skema komunikasi web *server* dan *browser*
(sumber: http://developer.mozilla.org, 2016)

Berkas tersebut memiliki format standar yang disebut dengan format SGML (*Standard General Markup Language*) Data yang berupa format ini kemudian akan ditampilkan oleh *browser* sesuai dengan kemampuan *browser* tersebut. Contohnya, bila data yang dikirim berupa gambar, *browser* yang hanya mampu menampilkan teks (misalnya *link*) tidak akan mampu menampilkan gambar tersebut, dan jika ada akan menampilkan teks alternatifnya saja. Sehingga disini web *server* berfungsi pula untuk mentransfer seluruh aspek pemberkasan konten pada suatu halaman web yang tidak hanya berisi teks, tetapi juga gambar, suara, dan video.

Komunikasi antara web *server* dengan *client* diatur oleh sebuah protokol komunikasi, yakni HTTP. Protokol ini akan melakukan fungsi *request-response* terhadap permintaan dari *user agent* yang dapat berupa *browser* atau web *crawl*. HTTP akan merespon permintaan dari *client* sesuai dengan permintaannya. HTTP didesain agar jaringan dengan skala menengah dapat berkomunikasi dengan mekanisme *client* dan *server*. Permintaan dari client berupa dokumen dengan ekstensi tertentu seperti *.htm*, *.html*, *.php*, dan lain sebagainya (https://msdn.microsoft.com, 2016).

Dokumen tersebut diakses dengan menyertakan URI (*Uniform Resource Identifier*). Secara khusus URL mewakili alamat sebuah *resource* konten yang dapat diakses menggunakan protokol HTTP atau HTTPS. URL (*Universal Resources Locator*) pada http atau URI pada https bersama dengan *hyperlink* yang ada pada dokumen HTML dapat membetuk konten yang terhubung satu sama lain.

Selain memberikan respon berupa konten ke *client*, web *server* juga dapat menerima konten dari *client* melalui sebuah form dalam *website* seperti form untuk mengupload berkas. *Web server* juga mendukung keberadaaan *server side scripting* menggunakan ASP (*Active Server Page*), PHP (*PHP Hyperteks Processor*), dan lain lain. Melalui bahasa pemprograman di sisi *server* tersebut respon dari web *server* dapat dikontrol sesuai dengan *script* bahasa pemprograman. Hal ini biasa digunakan untuk menghasilkan konten HTML secara dinamis (*on the fly*). Seperti kegiatan pengambilan dan atau modifikasi informasi dalam *database*.

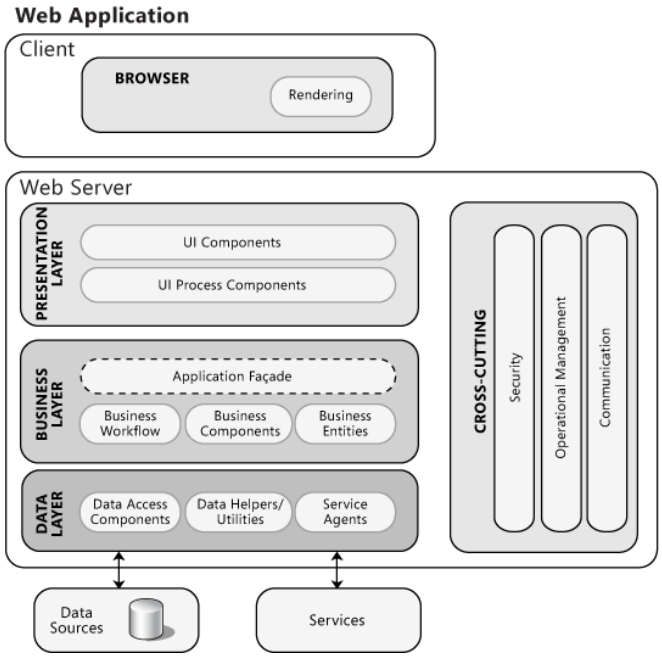
Web server tidak selalu hanya melayani kebutuhan *world wide web.* Lebih dari itu web *server* dapat ditemukan di *embedded device* seperti printer, router, webcam and juga dapat berjalan pada jaringan lokal. *Web server* juga dapat menghubungkan penggunanya dengan dunia *mobile* wireless internet atau yang sering disebut sebagai WAP (*Wireless Access Protocol*). Akses terhadap teknologi ini dapat dilakukan dengan *handphone* yang memiliki fitur WAP. Sehingga web *server* tidak hanya melayani dokumen berbasis HTML tetapi juga WML (*Wireless Markup Language*).

2.2 Aplikasi Web

Defini aplikasi web yang sederhana dipaparkan oleh Rouse (2011), yakni sebuah program aplikasi yang disimpan dalam *remote server* dan diakses melalui jaringan internet dengan antarmuka dalam sebuah *browser*. Sementara program aplikasi sendiri adalah sebuah program yang didesain untuk melakukan fungsi yang spesifik bagi pengguna atau program aplikasi lain dalam kasus tertentu (Rouse, 2011). Beberapa contoh aplikasi web antara lain layanan peta *online*, webmail, situs lelang *online*, situs wiki, pengirim pesan *online*, situs media sosial, dll.

Aplikasi web juga merupakan perangkat lunak komputer yang dibuat dengan bahasa pemprograman web, seperti PHP, ASP, Phyton, JAVA, dll. Dan dalam realisasinya diperlukan beberapa atribut lain seperti HTML dan CSS yang masing-masing digunakan untuk mendeskripsikan konten halaman aplikasi web

dan memformatnya sedemikian rupa. Pemprosesan data terjadi di *server* yang membuatnya diistilahkan *server-side*. Sehingga dengan mudah pula pengakses aplikasi web disebut *client-side*. Struktur dari aplikasi web berlapis-lapis dan memiliki fungsi masing-masing di setiap lapisan (http://msdn.microsoft.com, 2016). Struktur tipikal aplikasi web ini dapat dilihat pada Gambar 2.2 berikut.



Gambar 2.2 Struktur lapisan aplikasi web
(sumber: http://msdn.microsoft.com, 2016)

Terlihat pada gambar diatas bahwa inti dari aplikasi web dijalankan pada sisi *server*. Struktur tipikal lapisan aplikasi web memiliki tiga lapis arsitektur yang terdiri dari lapisan *presentation*, *bussiness*, dan data. Lapisan presentasi meliputi *User Interface* (UI) dan komponen lainnya. Sementara *bussiness layer* berisi bussiness workflow, komponen dan entitas. Lapisan data biasanya meliputi *data access* dan *service agent*. Lebih dari itu proses yang dijalankan di disi *server* adalah hasil permintaan dari sisi client melalui *browser.* Resource pemprosesan apa saja yang diperlukan oleh client akan ditangani sepenuhnya oleh web *server*.

2.3 Website

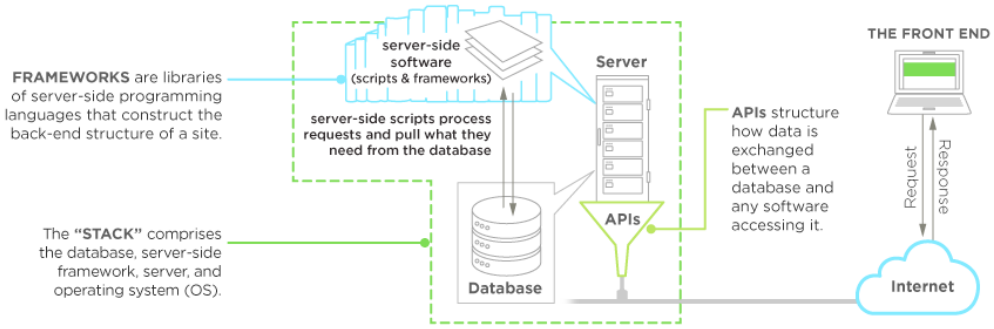
Sebuah *website* adalah kumpulan dari halaman web yang dimulai dengan halaman awal yang disebut *home page* dan diakses melalui web *browser* (Rouse, 2015). *Website* berisi berbagai dokumen yang berhubungan dengan instansi tertentu. Tiap *website* memiliki alamat web yang unik dan dapat dijangkau melalui koneksi internet. Home page yang dibuka oleh pengguna dapat berisi *link* ke halaman lain dalam *website* yang sama sehingga dokumen dalam *website* terhubung satu sama lain. *Website* disimpan dalam sebuah web *server* bersama ratusan / ribuan *website* lainnya.

Terdapat perbedaan cukup mendasar antara *website* dengan aplikasi web dalam hal fungsionalitas dan penggunaannya secara umum. Sebuah *website* sering kali difungsikan sebagai portal konten informasi tertentu. Contoh sebuah *website* adalah portal berita online, blog, dll. Sementara aplikasi web difungsikan secara khusus untuk berinteraksi dengan pengguna dalam rangka menyelesaikan suatu permasalahan. Seperti *Google SpreadSheet* yang digunakan untuk membuat lembar sebar secara online atau Google Map untuk memperoleh informasi spasial suatu daerah, dll. Dimana kedua hal tersebut diakses dengan metode yang sama, yakni via web *browser*.

2.4 Server-side Scripting

Server-side scripting adalah sebuah teknik yang digunakan dalam pengembangan web yang melibatkan skrip di web *server* yang menghasilkan respon yang berbeda pada tiap permintaan data pengguna (Wodehouse, 2016). Hal ini sebagai ganti dari konten statis yang tidak dapat memberikan interaktifitas dan personalisasi kebutuhan data pengguna. Skrip dapat ditulis dengan berbagai bahasa pemprograman web. Sesuai namanya skrip ini disimpan di web *server* dimana skrip

ini akan berinteraksi dengan *database*. Interaksi *server-side scripting* dengan *database*, *server* dan sebagainya ditunjukkan pada Gambar 2.2.



Gambar 2.3 Interaksi *server-side scripting* dengan *database*, *server*, API dan internet (upwork.com, 2016)

Berkas program di *server* bekerja jika terdapat *request* dari pengguna yang akan memproses data yang diminta. Permintaan ini dapat melalui URL yang diakses pengguna di *browser* atau via AJAX (*Asynchronous JavaScript And XML*) ketika halaman web diakses. Skrip ini juga memfasilitasi transfer data dari *server* ke *browser* dalam bentuk hasil yang bisa dipresentasikan ke pengguna. Skrip pada sisi *server* juga difugnsikan untuk aplikasi web yang dinamis seperti menjalankan validasi form, menyimpan dan mengambil data dan navigasi ke halaman lain.

2.5 Client-side Scripting

Client-side scripting umumnya mengacu pada program komputer di web yang dijalankan pada sisi client oleh web *browser* pengguna bukan di *server* (*server-side*). Jenis pemrograman komputer merupakan bagian penting dari konsep Dynamic HTML (DHTML), yang memungkinkan halaman web yang akan ditulis yaitu, memiliki konten yang berbeda dan berubah tergantung pada input pengguna, kondisi lingkungan seperti waktu hari, atau variabel lainnya. *Script* ini baru dijalankan ketika telah dimuat pada *browser*.

Secara umum *client-side scripting* ditulis dengan JavaScript sebagai tambahan dari kode HTML dan CSS. Hal fungsional yang dapat dilakukan skrip ini adalah menambah interaktifitas pengguna tanpa harus berkomunikasi kembali dengan *server*. Sehingga meringankan beban *request* berlebih di *server*.

2.6 Database

Database adalah kumpulan informasi yang terorganisasi sehingga dapat dengan mudah diakses, dikelola, dan diperbarui. Komponen penyusunnya terdiri dari koleksi skema, tabel, *query*, laporan dan lainnya. Data diorganisasi dalam metode tertentu yang mendukung proses yang membutuhkan informasi kemudian diakses dan dimanipulasi menggunakan perangkat lunak tertentu. Secara logikal *database* ini merupakan kumpulan dari arsip / berkas seperti transaksi penjualan, katalog produk, inventaris, dll. Dan diperlukan sebuah *database* manager yang menungkinkan pengguna untuk mengontrol aksi baca/tulis, sistem pelaporan dan analisa penggunaan.

Proses memasukkan dan mengambil data di *database* ke dan dari media penyimpanan data memerlukan perangkat lunak yang disebut dengan *Database Management System* (DBMS). Perangkat lunak ini memainkan tiga perananan besar, yakni mengatur sirkulasi data, mengatur *database engine* sehingga data dapat diakses, dikunci atau dimodifikasi dan menjaga skema *database* yang menentukan struktur logikal dari *database*. Secara spesifik DBMS bertugas untuk mengolah pendefinisian data, menangani permintaan pemakai untuk mengakses data, memeriksa sekuriti dan integriti data yang didefinisikan oleh *Database Administrator* (DBA), menangani kegagalan dalam pengaksesan data yang disebabkan oleh kerusakan sistem maupun disk dan menangani unjuk kerja semua fungsi secara efisien (Hindrianto, 2012).

Secara umum terdapat dua jenis *database* ditinjau dari relasi data dalam *database*, yakni *database flat-file* dan *database* relasional. Basis data *flat-file* secara fisik tersusuk dari berkas yang barisi sekumpulan *string* yang dipisahkan

dengan *delimiter* tertentu. Tipe basis data ini tepat digunakan untuk menyimpan data yang sederhana dan berukuran kecil. Sementara pada basis data relasional mempunyai struktur yang lebih logis terkait cara penyimpanan dimana tabel-tabel yang berada di basis data dapat dihubungkan satu dengan lainnya.

Basis data relasional menggunakan sekumpulan tabel dua dimensi yang masing-masing tabel tersusun atas baris dan kolom. Untuk membuat hubungan antara dua atau lebih tabel, digunakan *key* (atribut kunci) yaitu *primary key* di salah satu tabel dan *foreign key* di tabel yang lain. Saat ini, basis data relasional menjadi pilihan karena keunggulannya. Beberapa kelemahan yang mungkin dirasakan untuk basis data jenis ini adalah implementasi yang lebih sulit untuk data dalam jumlah besar dengan tingkat kompleksitasnya yang tinggi dan proses pencarian informasi yang lebih lambat karena perlu menghubungkan tabel-tabel terlebih dahulu apabila datanya tersebar di beberapa tabel. Beberapa contoh basis data relasional adalah Microsoft Access, MySQL, Oracle, Microsoft SQL Server dan PostgreSQL (Hindrianto, 2012).

2.6.1 Entity Relationalship Diagram (ERD)

ERD adalah *tool* untuk memodelkan data secara semantik dengan tujuan untuk mendeskripsikan abstaksi atau menggambarkan data. Model ER berisi komponen-komponen Himpunan Entitas dan Himpunan Relasi yang masing-masing dilengkapi dengan atribut-atribut yang merepresentasikan seluruh fakta dari dunia nyata yang kita tinjau. Digambarkan dengan lebih sistematis dengan menggunakan Diagram *Entity Relationship* (Diagram E-R) yang kali pertama dikembangkan oleh Chen pada tahun 1976.

Terdapat sejumlah konvensi mengenai notasi ERD. Notasi klasik sering digunakan untuk model konseptual. Berbagai notasi lain juga digunakan untuk menggambarkan secara logis dan fisik dari suatu basis data. Notasi-notasi simbolik yang digunakan dalam ERD adalah sebagai berikut :

a. **Entitas**

Entitas adalah segala sesuatu yang dapat digambarkan oleh data. Entitas juga dapat diartikan sebagai individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain (Fathansyah, 1999). Ada dua macam entitas yaitu entitas kuat dan entitas lemah. Entitas kuat merupakan entitas yang tidak memiliki ketergantungan dengan entitas lainnya. Contohnya entitas anggota. Sedangkan entitas lemah merupakan entitas yang kemunculannya tergantung pada keberadaaan entitas lain dalam suatu relasi.

b. **Atribut**

Atribut merupakan pendeskripsian karakteristik dari entitas. Atribut digambarkan dalam bentuk lingkaran atau elips. Atribut yang menjadi kunci entitas atau *key* diberi garis bawah.

c. **Relasi atau Hubungan**

Relasi menunjukkan adanya hubungan diantara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

Antar entitas dalam sebuah *database* memiliki tingkat relasi atau kardinalitas yang menunjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Macam-macam kardinalitas adalah:

- **Satu ke satu *(one to one),*** Setiap anggota entitas A hanya boleh berhubungan dengan satu anggota entitas B, begitu pula sebaliknya.
- **Satu ke banyak *(one to many)***, Setiap anggota entitas A dapat berhubungan dengan lebih dari satu anggota entitas B tetapi tidak sebaliknya.
- **Banyak ke banyak *(many to many),*** Setiap entitas A dapat berhubungan dengan banyak entitas himpunan entitas B dan demikian pula sebaliknya.

2.6.2 Normalisasi

Normalisasi data merupakan sebuah konsep dalam mengorganisasi data dalam struktur yang terintegrasi dan minimalisasi data yang belebihan (Robert Sheldon dan Geoff Moes, 2005). Normalisasi merupakan suatu teknik desain *database* yang mengorganisasi kolom, tabel , dan relasi basis data dengan cara

tertentu untuk mengurangi kelebihan dan ketergantungan data. Secara umum kegiatan dalam normalisasi meliputi pemecahan berbagai tabel menjadi beberapa tabel yang lebih kecil dan mendefinisikan kembali dengan hubungan tertentu. Proses normalisasi membantu pengembang untuk menciptakan *database* yang fleksibel dan efisien.

Acuan untuk normalisasi data adalah bentuk normal dimana untuk mencapainya harus mengikuti aturan dalam normalisasi. *Database* yang mentah sangat rentan terhadap isu keamanan, penggunaan ruang *hard disk*yang berlebihan, kecepatan *query*, efisiensi saat meng-*update database*, dan integritas data yang rendah. Suatu *database* dikatakan normal jika mengikuti suatu kaidah bentuk normal.

Tingkatan bahwa suatu *database* dikatakan normal tergantung pada level normal yang diaplikasikan. Sebagai contoh, beberapa desain *database* yang bertujuan hanya bentuk normal kedua, namun, beberapa *database* berusaha untuk mencapai kesesuaian terhadap bentuk normal keempat atau kelima. Sering kali ada *trade-off* antara bentuk normal dan kinerja sistem. Jika normalisasi dilakukan secara berlebihan justru akan mendapatkan efek yang berkebalikan. Sehingga desain *database* harus menjaga keseimbangan antara sepenuhnya bentuk normal dan kinerja sistem. Dalam kebanyakan situasi, tiga bentuk normal pertama memberikan keseimbangan yang dimaksud. Menurut Gerald V. Post (1999) proses normalisasi data memiliki tahap-tahap berikut:

1. **Bentuk tidak normal (0NF)**

   Yaitu bentuk berkas yang masih memiliki grub-grub berulang (*repeating-grub*) ketergantungan data secara parsial (*partial dependency*) atau ketergantungan transitif (*transitif dependency*).

2. **Bentuk Normal 1 (1NF)**

   Pada bentuk normal 1 grub-grub yang berulang harus dipecah ke dalam tabel baru. Desainer *database* harus memastikan untuk menyertakan salinan kunci dari tabel asal ke tabel yang baru. Sementara tabel yang

memegang grub berulang harus mempunyai kunci gabungan (*composite keys*) sehingga dapat dikombinasi kembali saat mengeksekusi *query*.

3. **Bentuk Normal 2 (2NF)**

   Bentuk normal kedua mensyaratkan bahwa tabel harus memenuhi bentuk normal pertama. Setiap kolom yang bukan *primary key* harus tergantung ada *key* tersebut.

4. **Bentuk Normal 3 (3NF)**

   Bentuk normal ketiga mensyarakatkan bahwa tabel harus memenuhi bentuk normal kedua. Syarat lainnya adalah kolom yang bukan *primary key* harus tergantung pada kolom yang jadi *key* pada satu tabel. Sementara *primary key* masing-masing tabel sendiri independen satu sama lain.

2.6.3 Model Data

Sebuah tabel tesusun dari baris dan kolom sehingga membentuk struktur tabel yang utuh. Baris diidentifikasi dengan penggunaan *primary key* dan data dinormalisasi berdasakan aturan normalisasi. Dengan berdisiplin dengan dua paramter tersebut sebuah proses desain *database* dapat dilaksanakan dengan baik. Terdapat banyak *tool* yang dapat digunakan untuk membuat permodelan *database*. Model data sendiri merupakan representasi fisik dari komponen yang menyusun *database* dan hubungan antar komponennya (Robert Sheldon dan Geoff Moes, 2005).

Namun demikian secara umum terdapat langkah umum yang dapat ditempuh untuk membuat model data yang menampilkan informasi berikut:

1. Tabel yang menyusun *database*
2. Kolom yang menyusun tabel
3. Tipe data yang didefinisikan tiap tabel
4. *Primary key* yang mengidentifikasi tiap baris
5. Relasi yang dibangun antar tabel

Permodelan data ini penting untuk diketahui sebelum mengimplementasikannya secara fisik pada RDBMS seperti MySQL.

2.6.4 Diagram Arus Data

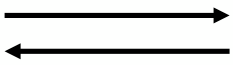
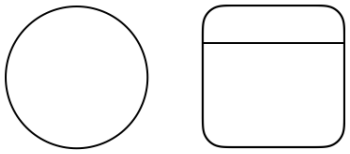
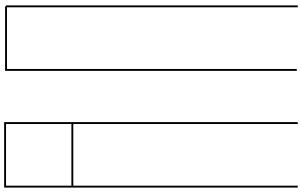
Diagram Alir Data (DAD) atau *Data Flow Diagram* (DFD) adalah diagram yang memodelkan sistem dengan entitias eksternal dimana data mengalir ke proses yang mengubah data tersebut dan membuat output aliran data yang mengalir ke proses atau entitas lain atau penyimpanan data (Rajaraman, 2004). DAD menggambarkan arus dari data sistem, yang penggunaannya sangat membantu untuk memahami sistem secara logika, terstruktur dan jelas.

DAD merupakan alat bantu dalam menggambarkan atau menjelaskan sistem yang sedang berjalan logis. DFD ini sering disebut juga dengan nama *Bubble Chart*, *Bubble Diagram*, model proses, diagram alur kerja, atau model fungsi. DFD merupakan salah satu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks dari pada data yang dimanipulasi oleh sistem. Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem.

Orientasi dari DFD adalah untuk menggambarkan analisa maupun rancangan sistem yang mudah dikomunikasikan oleh pembuat sistem kepada pemakai maupun pembuat program.sebuah sistem kontekstual DFD yang akan pertama kali muncul adalah interaksi antara sistem dan entitas luar. DFD didisain untuk menunjukkan sebuah sistem yang terbagi-bagi menjadi suatu bagian sub-sistem yang lebih kecil adan untuk menggarisbawahi arus data antara kedua hal yang tersebut diatas. Diagram ini lalu dikembangkan untuk melihat lebih rinci sehingga dapat terlihat model-model yang terdapat di dalamnya.

Secara fisik diagram ini digambarkan dengan simbol-simbol yang memiliki fungsi khusus yang ditunjukkan pada Tabel 2.1.

Tabel 2.1 Simbol-Simbol Diagram Arus Data

Simbol	Arti
	Terminator/Kesatuan Luar di DFD
	Arus Data
	Proses
	Penyimpanan Data

2.7 Diagram Alir

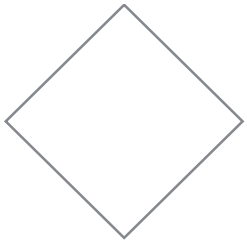
Diagram alir merupakan sebuah diagram dengan simbol-simbol grafis yang menyatakan aliran algoritma atau proses yang menampilkan langkah-langkah yang disimbolkan dalam bentuk kotak, beserta urutannya dengan menghubungkan masing masing langkah tersebut menggunakan tanda panah.

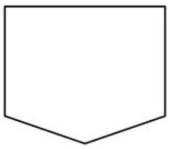
Diagram alir adalah dasar dari pemrograman dari pemrograman bahasa tingkat rendah sampai bahasa pemrograman tinggi. Pemrograman fungsional

ataupun pemrograman berorientasi objek, semuanya menggunakan diagram alir dalam analisis pembuatan desaiannya maupun proses *reverse engineering*-nya.

Berikut ini merupakan simbol-simbol yang digunakan untuk menggambarkan diagram alir yang disajika pada tabel 2.2.

Tabel 2.2 Simbol-Simbol Diagram Alir

Simbol	**Arti**
	Awal/Akhir Program
	Input/Output
	Proses
	Keputusan
	Garis Alir
	*Predefined Process*

Simbol	Arti
	Preparasi
	*Off-page Connector*

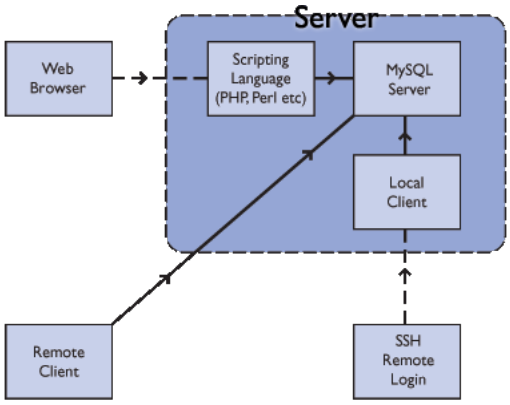
2.8 MySQL

MySQL merupakan *software* RDBMS (*Relational Database Management System*) yang didistribusikan secara gratis dibawah lisensi GNU GPL (General Public License). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basis data yang telah ada sebelumnya. SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basis data, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Sebagai penyedia basis data, MySQL mendukung operasi basis data transaksional maupun operasi basis data non-transaksional. Pada modus operasi non-transaksional, MySQL dapat dikatakan unggul dalam hal unjuk kerja dibandingkan perangkat lunak penyedia basis data kompetitor lainnya. Namun demikian pada modus non-transaksional tidak ada jaminan atas reliabilitas terhadap data yang tersimpan, karenanya modus non-transaksional hanya cocok untuk jenis aplikasi yang tidak membutuhkan reliabilitas data seperti aplikasi *blogging* berbasis web (Wordpress), CMS, dan sejenisnya. Untuk kebutuhan sistem yang ditujukan

untuk bisnis sangat disarankan untuk menggunakan modus basis data transaksional, hanya saja sebagai konsekuensinya unjuk kerja MySQL pada modus transaksional tidak secepat unjuk kerja pada modus non-transaksional.

MySQL dapat dipasang pada komputer *server* sehingga dapat melayani permintaan *database* dari berbagai client yang terhubung. Server MySQL tersebut dapat diakses secara langsung dengan memanfaaatkan berbagai antarmuka *client* yang mengirim *SQL statement* ke *server* kemudian mengembalikan hasilnya ke pengguna.



Gambar 2.4 Cara kerja Server MySQL

Local client merupakan sebuah program pada mesin yang sama sebagai *server*. Sebuah contoh dari hal ini adalah perangkat lunak *MySQLclient*. MySQL. *Scripting language*, dapat melewatkan *query* SQL ke *server* dan menampilkan hasilnya. *Remote Client* adalah sebuah program pada mesin yang berbeda yang dapat terhubung ke *server* dan menjalankan *SQL stetements*. *Remote login* memungkinkan pengguna terhubung ke mesin *server* untuk menjalankan salah satu klien lokal. Sementara web *browser* dapat digunakan untuk melakukan permintaan terhadap *scripting language*.

Dari interaksi berbagai komponen diatas, sebuah sistem yang melibatkan *database* MySQL terbentuk. Interaksi ini dapat berjalan sangat dinamis dengan menggunakan SQL. Baik program *MySQL client* atau bahasa pemprograman dapat menjalankan SQL untuk berintaraksi dengan MySQL.

2.9 PHP (PHP Hyperteks Processor)

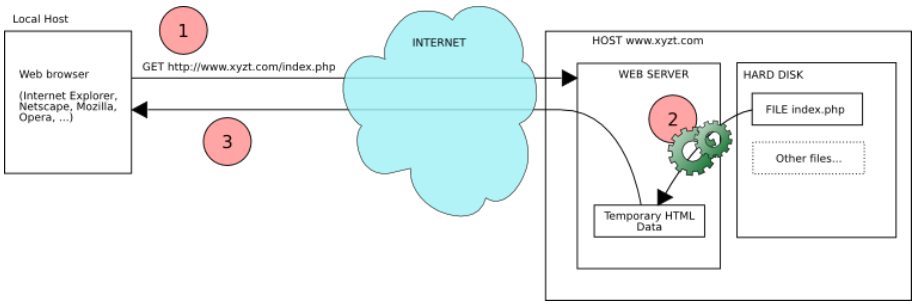
PHP merupakan *server-side scripting language* yang dapat disisipkan dalam HTML. PHP banyak digunakan sebagai bahasa pemprograman untuk pembuatan *website* (*web development*). Sebagian besar sitnak PHP mirip dengan C, Java, dan Perl ditambah dengan beberapa fungsi PHP yang khusus. Penggunaan PHP lebih memudahkan pengembang web untuk membuat *website* yang dinamis karena kemampuan PHP yang dapat disisipkan ke dokumen HTML.

Salah satu fitur yang signifikan dalam PHP adalah dukungan untuk berbagai *database*. Menulis sebuah halaman web *database* menjadi sangat mudah menggunakan salah satu ekstensi *database* tertentu (misalnya, untuk MySQL), atau menggunakan lapisan abstraksi seperti PDO, atau terhubung ke *database* yang mendukung standar *Open Database Connection* melalui perpanjangan ODBC. *Database* lain dapat memanfaatkan cURL atau soket, seperti CouchDB.

PHP juga memiliki dukungan untuk berkomunikasi dengan layanan lain menggunakan protokol seperti LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (pada Windows), dll. Programer juga dapat membuka soket jaringan raw dan berinteraksi dengan menggunakan protokol lainnya. PHP memiliki dukungan untuk pertukaran data yang kompleks antara hampir semua bahasa pemrograman Web. Berbicara tentang interkoneksi, PHP memiliki dukungan untuk Instansiasi objek Java dan menggunakannya secara transparan sebagai objek PHP.

Salah satu hal yang paling membedakan antara PHP dengan bahasa pemprograman *client-side* seperti Java adalah kode PHP dieksekusi di *server*. Proses yang terjadi didalamnya tersembunyi. Pengguna yang melakukan

permintaan halaman akan menerima hasil berupa dokumen HTML yang dapat dibaca di *browser*. Berkas PHP yang diminta oleh *client* melalui URL pada *browser* akan melalui proses *parsing* terlebih dahulu agar dapat menghasilkan dokumen HTML.



Gambar 2.5 Respon web *server* saat melayani permintaan berkas PHP

Pemprosesan pada sisi *server* akan menjamin bahwa hasil pemprosesan berupa data HTML akan dikirim ke *browser* untuk ditampilkan. *Interpreter* pada sisi *server* akan mengenali permintaan berkas PHP dari *client* kemudian melakukan semua hal diperlukan sesuai instruksi program yang ada pada berkas tersebut. Termasuk diantaranya adalah akses terhadap layanan *database*.

Mulai versi PHP 5 pengembang dapat menggunakan fitur OOP (*Object Oriented Programming)* yang memungkinkan programer PHP untuk menghasilkan kode yang fleksibel, *reuseable* dan *modular*.

2.10 Java, JavaScript, JQuery dan AJAX

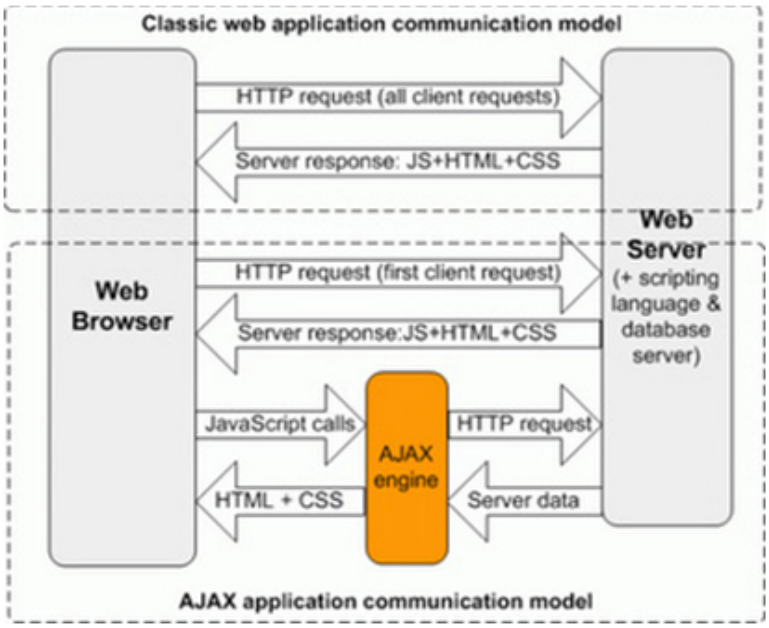
Java adalah bahasa pemprograman tingkat tinggi yang dikembangkan oleh Sun Microsystems pada sejak tahun 1991. Bahasa pemprograman ini dikembangkan dengan model yang mirip dengan bahasa C++ dan *Smalltalk*, namun dirancang agar lebih mudah dipakai dan *platform independent*, yaitu dapat dijalankan di berbagai jenis sistem operasi dan arsitektur komputer. Bahasa ini juga dirancang untuk pemrograman di Internet sehingga memperhatikan faktor keamanan dan portabilitas.

Bahasa pemprograman ini bersifat *platform independent* yang berarti dapat dengan mudah dipindahkan antar berbagai jenis sistem operasi dan berbagai jenis arsitektur komputer. JVM (*Java Virtual Machine*) yang terpasang di setiap komputer pengguna akan mengintrepretasikan kode sumber program yang berkekstensi *.class*. Selain itu bahasa pemprograman ini memiliki pustaka yang dapat digunakan untuk membuat aplikasi dengan cepat. Dan kemampuan *object oriented* memungkinkan skrip program terorganisir dan terskala dengan baik.

Berbeda dengan Java, JavaScript adalah *scripting language* yang bekerja dengan *interpreter* bukan dikompilasi. Javacript biasanya ditanam di kode HTML dalam satu berkas atau diembed dari berkas eksternal yang diterjemahkan oleh *browser*. Bahasa pemprograman ini diciptakan oleh Netspace dan bukan bagian dari platform Java yang dibuat oleh Sun Microsystems. JavaScript dibuat untuk memenuhi interaktifitas yang ada di halaman web yang tidak dapat dicapai dengan kode HTML dan bahasa pemprograman di sisi *server*.

Sementara itu JQuery adalah salah satu pustaka dari JavaScript yang menyederhanakan proses pembuatan skrip JavaScript yang meliputi seleksi dan manipulasi DOM (*Document Object Model*), *event handling*, animasi dan AJAX. JQuery harus disertkaan dalam dokumen HTML agar pustaka ini dapat digunakan dan diterjemahkan oleh *browser* seperti halnya skrip JavaScript. Pada JQuery kita dapat menjalankan AJAX (*Asyoncronous JavaScript And Xml*) yang memungkinkan pertukaran data dengan *server* dan memperoleh data teks, HTML, XML dan JSON melalui HTTP *get* atau *post*. Data ini selanjutkan dapat digunakan untuk memutakhirkan sebagian porsi dari halaman web untuk menyediakan

interaktifitas kepada pengguna. Perbandingan model komunikasi konvensional dan AJAX dapat dilihat pada Gambar 2.6.



Gambar 2.6 Model komunikasi konvensional dan AJAX

Permintaan data secara konvesional membutuhkan *reload* halaman web yang berarti meminta ulang data ke *server*. Setelah data tersedia data akan diterima oleh *browser* dengan menerjemahkan tag HTML secara utuh satu halaman. Pada permintaan data menggunakan AJAX, *reload* tidak diperlukan karena ada *engine* yang dapat meminta porsi data yang spesifik untuk dihadirkan kembali ke halaman web dan diproses lebih lanjut oleh JavaScript.

2.11 HTML dan CSS

HTML (*Hyperteks Markup Language*) merupakan kumpulan dari simbol atau kode tertentu yang digunakan untuk menyusun berkas yang ditampilkan di halaman *browser*. Kumpulan kode ini akan diterjemahkan oleh *browser* yang membentuk tampilan halaman *web*. HTML merupakan rekomendasi dari W3C

(*World Wide Web Consortium*) sebagai simbol standar dokumen yang juga digunakan acuan semua *browser*.

Sebuah dokumen HTML tersusun dari elemen HTML yang mendefinisikan link, gambar, *list item*, dll. Tiap elemen didefinisikan dengan *tag* HTML yang memiliki atribut tertentu. Elemen HTML ini dapat bekerjasama dengan CSS (*Cascading Style Sheets*) dan JavaScript untuk meningkatkan pengalaman pengguna.

CSS sendiri merupakan bahasa untuk mendeskripsikan presentasi halaman web seperti warna, tata letak, dan *font*. Dengan CSS halaman web dapat menjadi lebih responsif untuk beradaptasi dengan berbagai jenis perangkat yang memiliki ukuran layar berbeda. Kode CSS tidak memiliki ketegantungan terhadap HTML karena dapat digunakan dengan bahasa *markup* berbasis XML. Sebuah berkas CSS dapat ditulis di dokument HTML atau tulis secara parsial dalam sebuah berkas yang ditautkan ke dalam dokumen HTML.

2.12 JSON

JSON (*JavaScript Object Notation*) merupakan format penyimpanan informasi dalam cara tertentu sehingga terorganisir dan mudah diakses yang digunakan untuk pertukaran data. Data diformat dalam cara yang mudah dimengerti dan diakses dengan cara logis. Secara umum JSON tersusun atas dua sturktur, yakni:

- Koleksi dari pasangan nama dan nilai. Dalam berbagai bahasa pemprograman secara fisik direalisasikan dengan sebuah objek, record, daftar yang memiliki kunci atau array asosiatif.
- Kumpulan nilai yang tersusun dalam cara tertentu.

Struktur data ini universal dan dapat digunakan di banyak bahasa pemprograman.

Dalam integrasinya dengan JavaScript di halaman web JSON berupa teks yang datanya dapat diminta secara tidak sinkron menggunakan AJAX. Data teks ini dapat dikonversi menjadi objek yang selanjutnya dikirim ke *server* atau

sebaliknya, yakni diterima dari *server* dan dimanipulasi menjadi objek JavaScript. Kemudahan dalam transaksi data membuat JSON dipilih dalam sebagai format pertukaran data pada halaman web.

2.13 Aplikasi Mobile

Sesuai namanya aplikasi *mobile* adalah program aplikasi yang dikembangkan untuk perangkat genggam (*handled device*) seperti telepon genggam (*mobile device*), *smartphone*, PDA dan sebagainya. Sering kali aplikasi ini telah tertanam pada telepon genggam dari prabriknya atau diunduh oleh pengguna dari internet (Viswanathan, 2016). Tiap *mobile device* telah memiliki sistem operasi tertentu sebagai jembatan antara pengguna dengan program aplikasi. Terdapat beberapa sistem operasi *mobile device* yang populer seperti Android, IOS, Windows Phone dan BlackBerry.

Tiap vendor telepon genggam biasanya telah memiliki *marketplace* untuk mengunduh aplikasi bagi gratis maupun *proprietary*. Seperti *App Store* pada sistem operasi IOS dan *Google Play* pada sistem operasi Android. Untuk menggunakan aplikasi *mobile* pengguna yang telah mengunduh aplikasi harus menginstall di perangkatnya kemudian digunakan sesuai fungsi aplikasi.

2.14 Android

Android merupakan sebuah sistem operasi telepon seluler dan komputer tablet layar sentuh (*touch-screen*) berbasis Linux yang dikembangkan oleh Google. Sistem operasi ini adalah yang paling populer untuk menjembatani pengguna dengan *gadget* seperti telepon genggam. Pengembangan aplikasi pada platform Android menggunakan bahasa pemrograman Java. Terdapat aplikasi *built-in* Android yang preinstal pada saat pembelian sebuah *smartphone* antara lain klien *email*, program SMS, kalender, peta, *browser*, kontak, dan lain-lain.

Kode sumber Android disebarkan secara bebas oleh Google dengan lisensi *Open Source* meskipun banyak juga parangkat berbasis Android yang terinstal aplikasi berbayar yang menggunakan layanan Google. Dukungan terhadap pengembangan aplikasi Android terus dilakukan oleh Google dengan menciptakan *tool* dan IDE (*Integrated Development Environment*) untuk para pengembang.

2.15 PhoneGap

PhoneGap merupakan perangkat lunak kerangka kerja (*framework*) yang dikembangkan Adobe System yang digunakan untuk mengembangkan aplikasi *mobile*. Pengembangan aplikasi *mobile* dengan *PhoneGap* tidak mengharuskan developer menguasai bahasa pemprograman aplikasi *mobile* namun cukup pengetahuan web *development* seperti HTML, CSS dan Java*Script*. *PhoneGap* dapat menghasilkan aplikasi untuk banyak platform operasi sistem seperti Android, iOS, BlackBerry dan Windows *mobile*. Dengan satu kode sumber *PhoneGap* dapat menghasilkan aplikasi untuk banyak platform sistem operasi *mobile*.

PhoneGap bukan merupakan *framework* untuk pengembangan aplikasi *native* namun menyediakan beberapa fitur menggunakan layer abstraksi. Sehingga dengan menggunakan *PhoneGap developer* dapat mengakses beberapa fungsi device seperti: *Camera, Geolocation, Compass, Contacts, Media, Accelerometer, Network, Notification, Storage*, dll.

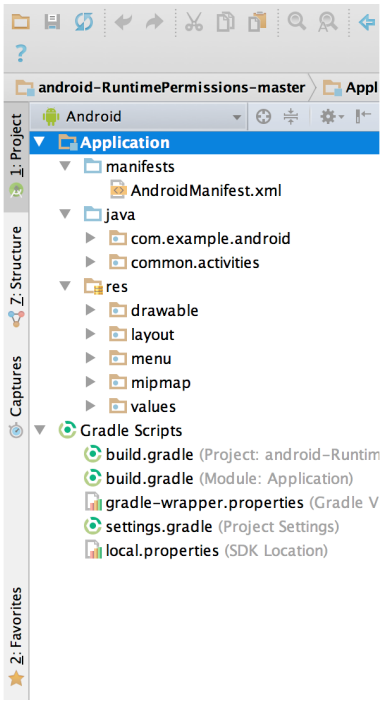
2.16 Android Studio

Android Studio adalah IDE resmi yang dibuat untuk pengembangan platform Android. Android Studio tersedia secara bebas di bawah Lisensi Apache 2.0. Pengembangan aplikasi di perangkat lunak ini dibuat dala sebuah proyek. Dimana proyek di Android Studio berisi satu atau lebih modul dengan berkas kode sumber dan berkas sumber daya. Jenis modul tersebut meliputi: modul aplikasi Android, modul pustaka dan modul *Google App Engine*.

Secara *default*, Android Studio menampilkan berkas proyek dalam tampilan proyek Android, seperti yang ditunjukkan pada Gambar 2.7. Tampilan ini diorganisasi sedemikian rupa oleh modul untuk menyediakan akses cepat ke berkas kunci sumber proyeknya. Semua berkas terlihat di tingkat atas di bawah *Script Gradle* dan setiap modul aplikasi berisi folder berikut:

- **manifest**: Berisi berkas AndroidManifest.xml.
- **java**: Berisi berkas kode sumber Java, termasuk kode uji JUnit.
- **res**: Berisi semua sumber daya non-kode, seperti layout XML, *string* UI, dan gambar bitmap.

Namun demikian terfapat perbedaan struktur berkas dalam pengembangan aplikasi yang berbeda.



Gambar 2.7 Struktur berkas dalam proyek Android Studio

2.17 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) adalah API (*Application Programming Interface*) yang digunakan untuk pengiriman pesan *cross-platform* yang memungkinkan pengiriman pesan kepada pengguna Android secara gratis. Layanan ini dihadirkan oleh Google untuk mendukung platform Android dalam pengiriman notifikasi (https://*Firebase*.google.com, 2016). Dengan menggunakan FCM dapat dilakukan pemberitahuan pada aplikasi klien jika yang *email* baru atau data lain yang tersedia untuk sinkronisasi. API ini juga dapat mengirimkan pesan untuk mendorong pengguna agar terlihat dalam aplikasi kembali. Kemudian untuk kasus penggunaan seperti instant messaging, pesan dapat mentransfer muatan hingga 4Kb ke aplikasi client.

Terdapat tiga tipe pesan yang dapat dikirimkan menggunakan FCM, yakni *Notification Message*, *Data Message* dan pesan dengan *Notification & Data Payload*.

2.17.1 Jenis Notifikasi FCM

Terdapat tiga jenis notifikasi yang dapat dilakukan menggunkan FCM, yakni sebagai berikut (Tamada, 2016).

1. Notification Message

Pada jenis pesan ini pemberitahuan akan ditangani oleh SDK *Firebase* sendiri. Biasanya pesan pemberitahuan berisi judul, pesan, ikon dll. Pesan ini dapat dikirim dari *User Interface Firebase* konsol. Dengan mengirimkan pesan ini tidak akan mendapatkan banyak kontrol atas pemberitahuan. pemberitahuan akan ditampilkan secara otomatis ketika aplikasi tersebut pada *background process*. Untuk mengirim pesan pemberitahuan diperlukan *notification key* dalam data JSON.

2. Data Message

Jenis pesan ini dapat dikirimkan untuk mengirim beberapa data tambahan bersama dengan notifikasi meskipun secara *default* pesan data telah

ditangani oleh aplikasi Android. Tapi mengirimkan pesan tersebut melalui konsol *Firebase* tidak dapat dilakukan. Aplikasi harus memiliki logika sisi *server* untuk mengirim pemberitahuan menggunakan API Firebase. Dan diperlukan *data key* saat mengirimkan pesan ini.

3. Notification and Data Payload Message

Sebuah pesan juga dapat berisi notifikasi dan data. Tipe pesan ini ditangani oleh dua skenario tergantung pada kondisi aplikasi Android pengguna (*background/foreground*). Dimana dibutuhkan dua *key*, yakni data dan notification *key*. Ketika aplikasi berada pada background proses maka aplikasi akan menerima notifikasi pada area notifikasi dan hanya menangani datanya ketika *user* membuka notifikasi. Sementara ketika dalam foreground process maka aplikasi akan menerima sebuah pesan objek dengan pesan notifikasi dan data sekaligus.

2.17.2 Penargetan Pesan

Pengiriman notifikasi dapat dilakukan secara segmentif dengan pemilihan target *audience*. Pesan dapat dikirimkan kepada seorang pengguna atau grup pengguna menggunakan nama topik (Tamada, 2016). Pengiriman pesan ke *single user* harus menyertakan ID registrasi *Firebase* dalam ini pesan yang diformat dalam JSON seperti Gambar 2.8.

```
{
  "to": "/topics/news",
  "data": {
    "message": "This is a Firebase Cloud Messaging Topic Message!",
   }
}
```

Gambar 2.8 Format data JSON pada pengiriman pesan *single user*

Kemudian untuk pengiriman pesan dengan pengguna yang tertarget dapat menggunakan *topic message* dimana pengguna dikelompokkan dalam grup tertentu

untuk dikirimkan pesan secara masal. Format data JSON yang valid ditunjukkan pada Gambar 2.9.

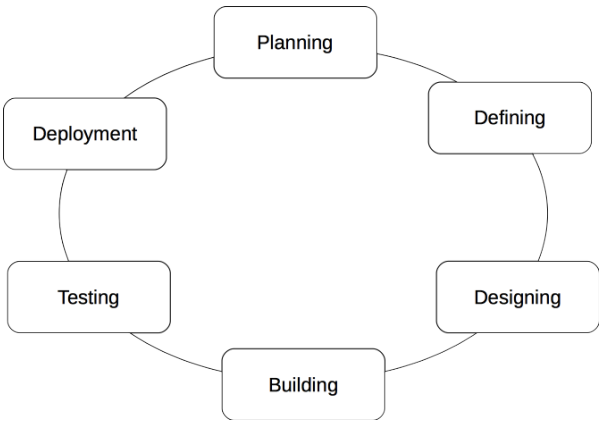
```
{
  "to": "e1w6hEbZn-8:APA91bEUIb2JewYCIiApsMu5JfI5Ak...",
  "data": {
    "message": "This is a Firebase Cloud Messaging Topic Message!",
   }
}
```

Gambar 2.9 Format data JSON pada pengiriman pesan *topic messaging*

2.18 Software Development Lifecycle

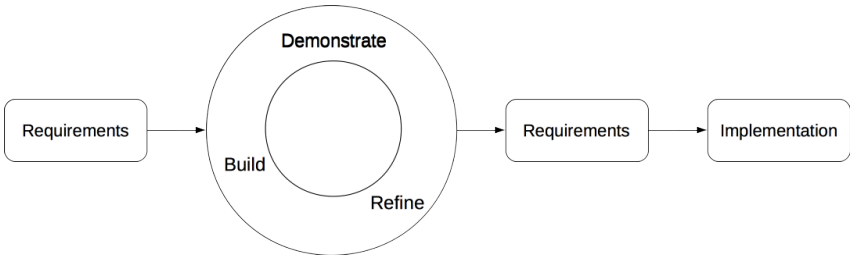
Software Development Lifecycle (SDLC) merupakan serangkaian tahapan proses yang digunakan secara luas pada industri pengembangan perangkat lunak untuk menghasilkan perangkat lunak yang berkualitas (http:// tutorialspoint.com). Metodologi yang digunakan dalam proses SDLC dapat berbeda antara satu industri atau organisasi. Namun terdapat standar seperit ISO/IEC 12207 yang mewakili proses yang membuat sebuah siklus kerja untuk pengembangan perangkat lunak dan menyediakan mode untuk pengembangan dan konfigurasi dari sistem perangkat lunak.

Tujuan utama dari proses SDLC adalah untuk menghasilkan sebuah produk yang *cost-efficient*, *effective* dan pada saat yang sama berkualitas tinggi. Sebuah SDLC standar memiliki tahapan proses berikut: *analysis* (kebutuhan dan desain), *construction/implementation*, *testing*, *release*/*deployment* dan *maintenance*. Dalam hal ini terdapat beberapa model *lifecycle* yang dikenal, yakni: *Rapid Application Development* (RAD), *WaterFall, Iterative,* dll. Fase standar yang dimiliki oleh SDLC ditunjukkan oleh Gambar 2.10.



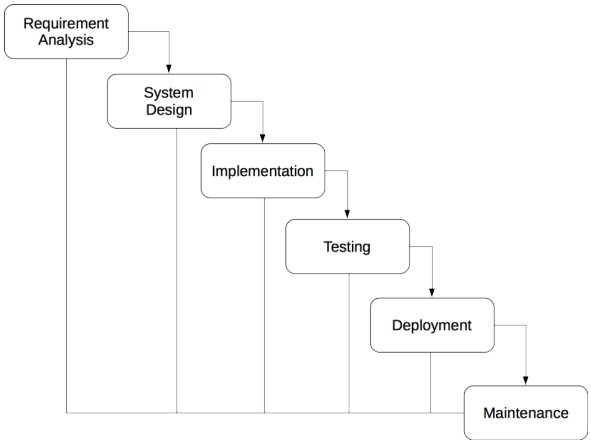
Gambar 2.10 Siklus SDLC standar

RAD adalah model proses perkembangan software sekuensial linier yang menekankan siklus perkembangan yang sangat pendek. Hal ini merupakan adaptasi dari model siklus pengembangan konvensional. Model ini menitik beratkan pada kecepatan proses pengembangan perangkat lunak dengan cara analisa kebutuhan secara cepat dan proses desain yang secara aktif melibatkan pengguna (demonstrasi). Seperti yang terlihat pada Gambar 2.11.



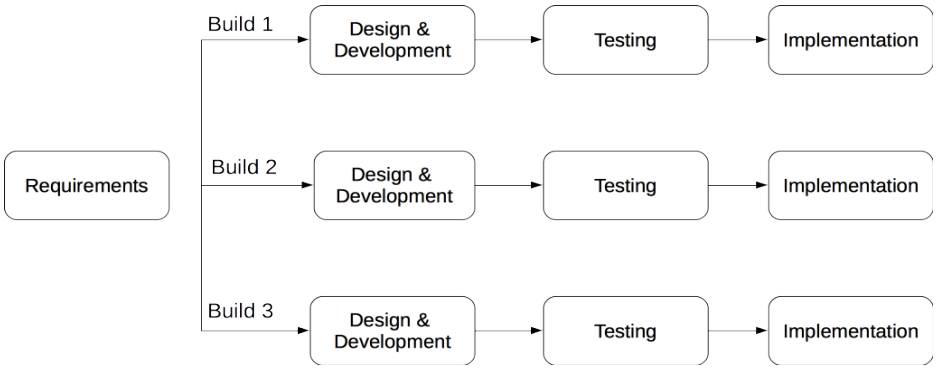
Gambar 2.11 Tahapan proses SDLC pada model RAD

Model *Waterfall* merupakan model pengembangan linear-sekuensial dimana terdapat serangkaian proses dan tiap proses dilalui secara urut dan bertahap satu per satu. Tiap tahapan harus terlesesaikan sebelum melanjutkan ke tahapan berikutnya sehingga tidak ada *overlapping* tahapan. Model *Waterfall* ini ditunjukkan oleh Gambar 2.12.



Gambar 2.12 Tahapan proses SDLC pada model *Waterfall*

Sementara itu model iteratif merupakan pecahan dari fase SDLC tradisional dimana implentasi perangkat lunak dipecah menjadi beberapa bagian dengan tiap bagian yang memiliki proses *lifecycle* yang utuh mulai dari *plan* hingga *maitenance*. Sehingga terdapat beberapa iterasi dalam keseluruhan waktu pengembangan perangkat lunak. Tiap iterasi akan menyelesaikan bagian tertentu dari perangkat lunak seperti ditunjukkan pada Gambar 2.13.



Gambar 2.13 Tahapan proses SDLC pada model *Iterative*