BAB III
LANDASAN TEORI

3.1 Smartphone

Smartphone merupakan ponsel yang memiliki banyak fungsi yang menggabungkan beberapa fungsi seperti sebuah PDA, misalnya kalender dan buku panggilan. Smartphone bisa mengakses internet, mengirim dan menerima e-mail, bermain game, menulis, dan mengedit dokumen seperti Microsoft Word dan Excell. Seperti komputer mini, smartphone dapat digunakan untuk membuat aplikasi. (Riduwan, 2008)

Perkembangan smartphone terus mengalami peningkatan sistem operasi seluler dan dalam lima tahun terakhir telah terjadi beberapa peningkatan di *Apple* iOS, *Android* dan *Blackberry* OS. Sistem Operasi seluler yang paling populer (iOS, Android, Blackberry OS, Windows Mobile) dan vendor Smartphone utama (*Apple*, *Samsung*, HTC, Motorola, Nokia, LG, Sony, dll.) berkonsentrasi untuk menghadirkan fitur terbaik dalam sistem operasi dan perangkat yang akan digunakan untuk konsumen (perusahaan dan umum). Peran Android telah memberikan manfaat besar bagi penggunanya hal itu dikarenakan android memberikan peluang besar bagi semua vendor untuk membangun perangkat menggunakan teknologi *Android open source* yang hebat (Sarwar, 2013)

3.2 Framework

Framework dalam sistem berorientasi objek, merupakan kumpulan *class* yang melambangkan bentuk abstrak untuk pemecahan sejumlah masalah yang berhubungan

Framework adalah sebuah *software* untuk memudahkan para *programmer* untuk membuat sebuah aplikasi *website* maupun aplikasi *mobile phone* yang didalamnya berbagai fungsi diantaranya *plugin*, dan konsep untuk membentuk suatu sistem tertentu agar tersusun dan terstruktur dengan rapi.

Dengan menggunakan *framework* bukan berarti akan terbebas dari pengkodean. Karena sebagai pengguna *framework* harus menggunakan fungsi-fungsi dan *variable* yang ada di dalam sebuah *framework* yang digunakan. Saat ini ada beberapa *framework* untuk pembuatan *website* dan *framework* untuk pembuatan aplikasi (Idcloudhost, 2017)

3.3 CodeIgniter

CodeIgniter adalah sebuah *framework* untuk membuat sebuah *website* yang bersifat *open source. Framework* ini menggunakan sistem MVC (Model, View, Controller) yang dikembangkan oleh Rick Ellis pada tahun 22006 untuk membangun *website* dinamis dengan menggunakan PHP juga mempercepat pengembang dalam melakukan pembuatan aplikasi *website.* Selain ringan dan cepat, CodeIgniter juga memiliki dokumentasi yang lengkap karena itu menjadi salah satu alasan kenapa banyak digunakan di seluruh dunia. Adapun kelebihan *codeigniter* yaitu (Budiyanto & Surya, 2018).

1) *Open Source*

   Framework yang gratis untuk digunakan dan dapat dikembangkan secara legal. *Codeigniter* berlisensi *Apache open source* sehingga dapat dikembangkan dan digunakan sesuai dengan kebutuhan.

2) *Multiplatform*

   Codeigniter sangat ringan ketika dijalankan pada berbagai *platform*, pada bagian sistem utama *codeigniter* hanya memerlukan kapasitas yang sedikit untuk server dalam *class library.*

3) Efisensi Waktu

   Dengan adanya struktur dan *library* yang telah disediakan *framework* maka tidak lagi memikirkan hal-hal tersebut, hal ini dapat membuat *programmer* fokus pada proses pengembangan sistem yang akan dibangun.

4) Menggunakan Metode *MVC*

Metode *MVC* adalah metode dengan cara *Model, View, Controller*. Dengan *model framework* ini maka dapat mempermudah dalam membedakan antara tampilan dan program.

5) *User Friendly*

Codeigniter sangat mudah digunakan oleh berbagai kalangan *user* atau *programmer*, hal ini dikarenakan bentuk dan tampilan yang sudah dirancang secara terstruktur.

6) URL yang *User Friendly*

Ketika *URL* di panggil maka akan langsung membentuk sebuah alamat contohnya *http://namasitus.com/blog/contoh/alamat-blog* hal ini dikarenakan *CodeIgniter* telah dirancang dengan rapi.

7) *Framework* yang Lengkap

CodeIgniter telah dikemas secara lengkap karena didalamnya terdapat kumpulan *class* yang ada didalam *library* yang telah tersedia sehingga sangat menunjang dalam pembuatan *session*, penggunaan *XML-RPC*, dan lain-lain

8) *User Guide*

Didalam *CodeIgniter* telah dilengkapi dengan pendokumentasian yang cukup baik dan lengkap, sehingga dapat membantu para *programmer* dalam mempelajari lebih jauh tentang *CodeIgniter*

9) *Reuse of Code*

Dengan menggunakan *framework* maka program yang dibuat akan memiliki struktur yang baku, sehingga dapat digunakan kembali pada proyek-proyek lainnya.

10) Komunitas

Saat ini komunitas *CodeIgniter* sudah mulai berkembang sehingga dapat dijadikan acuan dalan mencari ilmu dalam memecahkan suatu masalah yang dihadapi, selain itu dapat untuk meningkatkan pengetahuan akan kemampuan pemograman dengan menggunakan *framework CodeIgniter*

11) Kumpulan *Best Practice*

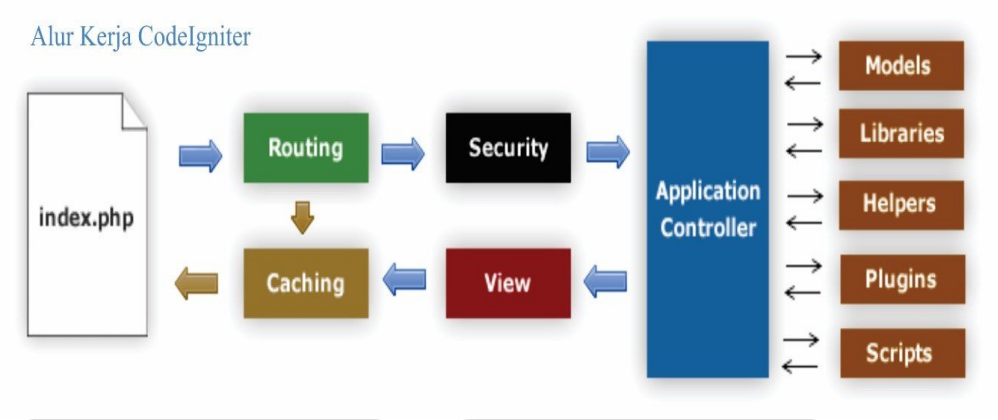
Sebuah *framework* merupakan kumpulan *best practice* yang sudah teruji. Jadi, dapat digunakan untuk meningkatkan kualitas program atau sistem yang dibangun.

12) Berjalan di *PHP* Versi 5 dan 7

Sekarang ini *PHP* sudah mencapai versi 7, meskipun begitu masih banyak orang tetap menggunakan *PHP* versi 5, oleh sebab itu *CodeIgniter* dikembangkan agar tetap kompatibel dengan *PHP* versi 5 dan dapat dijalankan dengan *PHP* versi 7.

3.3.1 Sistem Pada CodeIgniter

Pada sistem *framework codeigniter* memiliki keunikan tersendiri dimana tampilan *website* merupakan bagian akhir ketika halaman *website* ditampilkan dengan menggunakan browser. Perhatikan alur kerja *CodeIgniter* berikut ini:



Gambar 3.1. Alur Kerja *CodeIgniter*

Sumber(*https://idcloudhost.com/panduan/mengenal-apa-itu-framework-codeigniter/*).

Berdasarkan Gambar 3.1 menjelaskan bahwa alur kerja pada *CodeIgniter* terbagi menjadi beberapa alur yaitu:

1) *Index.php*

Index.php berfungis sebagai *file* pertama dalam program yang akan dibaca oleh program.

2) *Routing*

Berfungsi sebagai menerima permintaan dari HTTP *request* untuk menentukan hal apa yang harus dilakukan oleh program.

3) *Cache*

Jika *file cache* ada (aktif) maka sistem akan langsung menuju ke bagian *caching* yang kemudian akan tampilkan pada halaman *website* tanpa melalui sistem *CodeIgniter.*

4) *Security*

Seluruh permintaan HTTP dan *form* yang dikirim oleh *user* akan difilter atau disaring pada bagian *security*, hal ini dilakukan untuk menjaga keamanan data.

5) *Controller*

Pada *Controller* membuat *model*, *core, libraries, plugins, helpers,* dan semua *resource* yang diperlukan untuk memproses *request* yang telah dilakukan oleh *user*.

6) *View*

Request yang dihasilkan akan dikirimkan ke *browser*, jika *cache* telah bersedia atau aktif maka *view* akan disimpan ke dalam *cache* terlebih dahulu dan *request* berikutnya akan langsung ditampilkan.

3.3.2 MODEL, VIEW, CONTROLLER (MVC)

MVC adalah konsep dasar yang harus dipelajari sebelum mengenal *CodeIgniter*. *MVC* merupakan sebuah *pattern* atau teknik pemograman yang memisahkan antar pengembang aplikasi berdasarkan komponen utama pada sebuah aplikasi, seperti manipulasi data, *user interface* dan bagian yang menjadi kontrol aplikasi. Secara sederhana dapat dikatakan bahwa anatara desain dan proses data berada pada tempat yang terpisah. Saat ini *MODEL, VIEW, CONTROLLER* merupakan suatu konsep yang cukup populer dalam pembangunan aplikasi *website* yang hal ini berawal pada bahasa *small talk.* Terdapat 3 jenis komponen yang membangun suatu *MVC pattern,* yaitu

1) *Model*

Model berhubungan dengan data dan interaksi ke *database* atau *webservice*. Model juga memperesentasikan stuktur data dari aplikasi yang bisa berupa basis data maupun data lain, misalnya dalam bentik *file* teks, XML maupun *webservice.* Biasanya didalam *model* berisi *class* dan fungsi yaitu untuk melakukan manipulasi data seperti *insert, update, delete* dan *search,* namun tidak ada berhubungan dengan bagian *view* secara langsung. Sebuah aplikasi *website* biasanya menggunakan *database* dalam penyimpanan data, oleh karena itu *model* biasanya akan berhubungan dengan perintah-perintah *query sql.*

2) *View*

View berhubungan dengan segala sesuatau yang akan ditempatkan ke *end-user*. Biasa berupa halaman *website*, RSS, *JavaScript*, *JQuery.* Didalam *view* juga harus menghindari adanya kode untuk melakukan koneksi ke *database*. *View* hanya dikhususkan untuk menampilkan data-data hasil dari *model* dan *controller*. Bagian ini tidak memiliki akses secara langsung terhadap bagian *model.*

3) *Controller*

Controller merupakan penghubung antara *model* dan *vew.* Didalam *Controller* inilah terdapat *class* dan fungsi-fungsi yang memproses permintaan dari *View* kedalam struktur data didalam *model.* Controller juga tidak boleh berisi kode untuk mengakses basis data. Tugas *Controller* adalah menyediakan penangan *error*, mengerjakan proses logika dari aplikasi, serta melakukan *validasi* atau pengecekan terhadap *input*

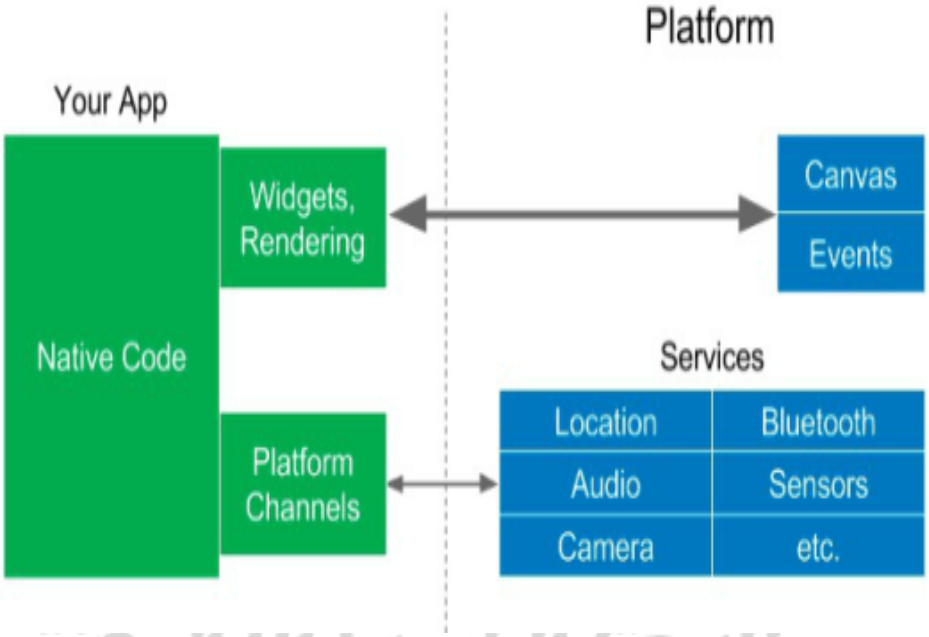
3.4 Flutter

Flutter adalah sebuah *software development kit* (SDK) buatan *Google* yang berfungsi untuk membuat aplikasi *mobile phone* menggunakan Bahasa pemograman *Dart,* baik untuk *android* maupun IOS. Dengan *flutter,* aplikasi *android* dan IOS dapat dibuat menggunakan basis kode dan Bahasa pemrograman yang sama, yaitu *dart*, Bahasa pemrograman yang juga diproduksi oleh *google* pada

tahun 2011. Sebelumnya, aplikasi murni (*native*) untuk *android* perlu dibuat menggunakan Bahasa *Java* atau *Kotlin*, sedangkan aplikasi IOS perlu dibuat menggunakan Bahasa pemrograman *Objective-C* atau *swift*. *Flutter* ditujukan untuk mempermudah dan mempercepat proses pengembangan aplikasi *mobile* yang dapat berjalan di atas *android* dan IOS, tanpa harus mempelajari dua Bahasa pemgrograman secara terpisah.

Flutter dapat dikatakan sebagai produk *google* yang masih relatif baru. Rilis perdana *flutter*, versi *Alpha* (v.0.0.6), dipublikasikan pada bulan Mei 2017. Dan versi v.1.0 merupakan yang stabil saat diterbitkan oleh *google* (Raharjo.2019).

3.4.1 Sistem Flutter



Gambar 3.2. Kinerja *Flutter*

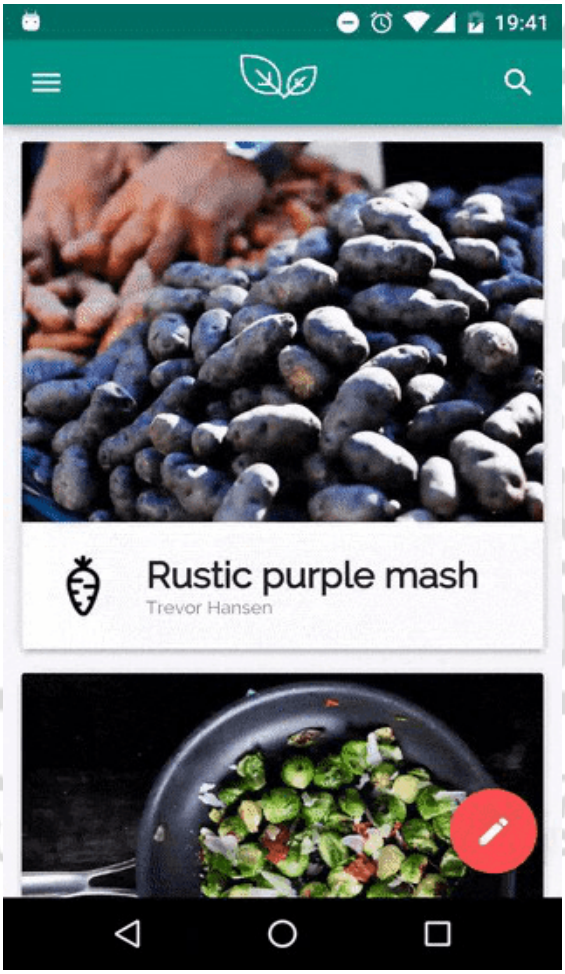
Sumber (http://omahti.web.id/blog/flutter/)

Salah satu keunggulan *flutter* daru pada teknologi *multiplatform* lainnya seperti *react native* adalah dalam hal kinerja. *Flutter* menjanjikan aplikasi yang dibuat akan mendapatkan tingkat sebesar 60 *frame per second*. Kinerja ini bisa didapatkan karena cara kerja dari *flutter* sedikit berbeda. Kode-kode yang ditulis dengan menggunakan bahasa *dart* akan diubah menjadi kode *C/C++* kemudian dikompilasi secara *native*. Hal inilah yang menyebabkan *flutter* memilii *performa*

yang hampir setara dengan aplikasi *native*. *Flutter* bisa berjalan pada sistem operasi *android* mulai dari versi 4.1 dan *IOS* mulai dari versi 8. Serta dijalankan di perangkat asli maupun simulator.

3.4.2 Widget Flutter

Widget dalam *flutter* adalah elemen yang sangat penting digunakan untuk mengontrol tampilan antarmuka suatu aplikasi. Pada kode tampilan diatas semua adalah *widget* termasuk tata letak. Pusat *widget* memusatkan anaknya dalam induknya (misalnya layar).



Gambar 3.3. *Widget Flutter*

Sumber (*http://omahti.web.id/blog/flutter/*)

Widget tata letak kolom mengatur anak-anaknya (daftar *widget*) secara vertikal. Kolom berisi *widget* teks dan *widget* ikon (yang memang memiliki

prorperti warnanya). Pada *flutter*, *padding* dan *margin* adalah *widget*. Tema adalah *widget*. Ini memungkinkan kemudahan dalam mengatur tampilan antarmuka karena setiap elemen bisa diatur menggunakan *widget* (Adminoti, 2018).

3.5 HTML

Hypertext Markup Language (HTML) adalah sebuah bahasa markah yang digunakan untuk membuat sebuah halaman *web*, menampilkan berbagai informasi di dalam sebuah penjelajah *web Internet* dan pemformatan *hypertext* sederhana yang ditulis dalam berkas format *ASCII* agar dapat menghasilkan tampilan wujud yang terintegrasi. Dengan kata lain, berkas yang dibuat dalam perangkat lunak pengolah kata dan disimpan dalam format *ASCII* normal sehingga menjadi halaman *web* dengan perintah-perintah *HTML*. Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan *SGML* (*Standard Generalized Markup Language*), *HTML* adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman *web*. *HTML* saat ini merupakan standar *Internet* yang didefinisikan dan dikendalikan penggunaannya oleh *World Wide Web Consortium* (*W3C*). *HTML* dibuat oleh kolaborasi Caillau TIM dengan Berners-lee Robert ketika mereka bekerja di *CERN* pada tahun 1989 (*CERN* adalah lembaga penelitian fisika energi tinggi di Jenewa) (Anonim, 2006).

3.5.1 Sejarah HTML

Pada tahun 1980 seorang ahli fisika, Tim Berners-Lee, dan juga seorang kontraktor di *CERN* (Organisasi Eropa untuk Riset Nuklir) mengusulkan dan menyusun ENQUIRE, sebuah sistem untuk ilmuwan *CERN* dalam membagi dokumen. Sembilan tahun kemudian, Berners-Lee mengusulkan adanya sistem markah berbasis *internet*. Berners-Lee menspesifikasikan *HTML* dan menulis jaringan beserta perangkat lunaknya di akhir 1990. Pada tahun yang sama, Berners-Lee dan Robert Cailliau, insinyur sistem data *CERN* berkolaborasi dalam sebuah permintaan untuk pendanaan, namun tidak diterima secara resmi oleh *CERN*. Di catatan pribadinya sejak 1990 dia mendaftar beberapa dari banyak daerah yang menggunakan *hypertext* dan pertama-tama menempatkan sebuah ensiklopedia.

Penjelasan pertama yang dibagi untuk umum dari HTML adalah sebuah dokumen yang disebut "Tanda *HTML*", pertama kali disebutkan di *Internet* oleh Tim Berners-Lee pada akhir 1991. Tanda ini menggambarkan 18 elemen awal mula, versi sederhana dari HTML. Kecuali untuk tag *hyperlink*, yang sangat dipengaruhi oleh *SGMLguid*, *in-house Standard Generalized Markup Language* (*SGML*) berbasis format dokumen di *CERN*. Sebelas elemen ini masih ada di *HTML* 4. *HTML* adalah bahasa markah yang digunakan peramban untuk menafsirkan dan menulis teks, gambar dan bahan lainnya ke dalam halaman *web* secara *visual* maupun suara. Karakteristik dasar untuk setiap item dari markah HTML didefinisikan di dalam peramban, dan karakteristik ini dapat diubah atau ditingkatkan dengan menggunakan tambahan halaman *web* desainer *CSS*. Banyak elemen teks ditemukan di laporan teknis *ISO* pada tahun 1988 TR 9537 Teknik untuk menggunakan *SGML*, yang pada gilirannya meliputi fitur bahasa format teks awal seperti yang digunakan oleh komandan RUNOFF dikembangkan pada awal 1960-an untuk sistem operasi: perintah-perintah format ini berasal dari perintah yang digunakan oleh pengetik untuk memformat dokumen *CTSS* secara manual. Namun, konsep *SGML* dari markah umum didasarkan pada unsur-unsur daripada hanya efek cetak, dengan pemisahan struktur dan markah juga; *HTML* telah semakin bergerak ke arah ini dengan CSS (Berners-Lee, 1989).

3.5.2 Kegunaan HTML

Dokumen HTML mirip dengan dokumen tulisan biasa, hanya dalam dokumen ini sebuah tulisan bisa memuat instruksi yang ditandai dengan kode atau lebih dikenal dengan TAG tertentu. Sebagai contoh jika ingin membuat tulisan ditampilkan menjadi tebal seperti: **TAMPIL TEBAL**, maka penulisannya dilakukan dengan cara: `<b> TAMPIL TEBAL</b>`. Tanda `<b>` digunakan untuk mengaktifkan instruksi cetak tebal, diikuti oleh tulisan yang ingin ditebalkan, dan diakhiri dengan tanda `</b>` untuk menonaktifkan cetak tebal tersebut. HTML lebih menekankan pada penggambaran komponen-komponen struktur dan format di dalam halaman *web* daripada menentukan penampilannya. Sedangkan penjelajah *web* digunakan untuk menginterpretasikan susunan halaman ke gaya *built-in*

penjelajah *web* dengan menggunakan jenis tulisan, tab, warna, garis, dan perataan text yang dikehendaki ke komputer yang menampilkan halaman *web*. Salah satu hal Penting tentang eksistensi HTML adalah tersedianya *Lingua franca* (bahasa Komunikasi) antar komputer dengan kemampuan berbeda. Pengguna *Macintosh* tidak dapat melihat tampilan yang sama sebagaimana tampilan yang terlihat dalam pc berbasis *Windows*. Pengguna *Microsoft Windows* pun tidak akan dapat melihat tampilan yang sama sebagaimana tampilan yang terlihat pada pengguna yang menggunakan Produk-produk *Sun Microsystems*. namun demikian pengguna-pengguna tersebut dapat melihat semua halaman *web* yang telah diformat dan berisi Grafika dan Pranala (Andi, 2001)

3.5.3 Tanda HTML

Secara garis besar, terdapat beberapa 4 jenis elemen dari *HTML* yang sering digunakan yaitu (Shelly, 2001):

1) Struktural. Tanda yang menentukan *level* atau tingkatan dari sebuah tulisan (contoh, *<h1>Golf</h1>* akan memerintahkan peramban untuk menampilkan "Golf" sebagai tulisan tebal besar yang menunjukkan sebagai *Heading* 1.
2) Presentasional. Tanda yang menentukan tampilan dari sebuah tulisan tidak peduli dengan *level* dari tulisan tersebut (contoh, *boldface* akan menampilkan *bold*. Tanda presentasional saat ini sudah mulai digantikan oleh *CSS* dan tidak direkomendasikan untuk mengatur tampilan tulisan,
3) Hiperteks. Tanda yang menunjukkan pranala ke bagian dari dokumen tersebut atau pranala ke dokumen lain (contoh, <a *href*="*http*://*www.wikipedia*.org/"> Wikipedia akan menampilkan Wikipedia sebagai sebuah *hyperlink* ke *URL* tertentu),
4) Elemen *widget* yang membuat objek-objek lain seperti tombol (<*button*>), daftar (<*li*>), dan garis horizontal (<*hr*>). Konsep hiperteks pada *HTML* memungkinkan pembuatan tautan pada suatu kelompok kata atau frasa untuk menuju ke bagian manapun dalam *World Wide Web*.

3.6 PHP

PHP adalah bahasa pemrograman *script server-side* yang didesain untuk pengembangan *web*. Selain itu, *PHP* juga bisa digunakan sebagai bahasa pemrograman umum. *PHP* di kembangkan pada tahun 1995 oleh Rasmus Lerdorf, dan sekarang dikelola oleh The PHP Group. Situs resmi *PHP* beralamat di *http://www.php.net*.

PHP disebut bahasa pemrograman *server side* karena *PHP* diproses pada komputer *server*. Hal ini berbeda dibandingkan dengan bahasa pemrograman *client-side* seperti *JavaScript* yang diproses pada *web browser* (*client*).

Pada awalnya *PHP* merupakan singkatan dari *Personal Home Page*. Sesuai dengan namanya, *PHP* digunakan untuk membuat *website* pribadi. Dalam beberapa tahun perkembangannya, *PHP* menjelma menjadi bahasa pemrograman *web* yang *powerfull* dan tidak hanya digunakan untuk membuat halaman *web* sederhana, tetapi juga *website* populer yang digunakan oleh jutaan orang seperti wikipedia, *wordpress*, *joomla*, dll. (Leerdorf, 2007)

Pada awalnya *PHP* merupakan kependekan dari *Personal Home Page* (Situs personal). *PHP* pertama kali dibuat oleh Rasmus Lerdorf pada tahun 1995. Pada waktu itu *PHP* masih bernama *Form Interpreted* (*FI*), yang wujudnya berupa sekumpulan skrip yang digunakan untuk mengolah data formulir dari *web*. Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya *PHP/FI*. Dengan perilisan kode sumber ini menjadi sumber terbuka, maka banyak pemrogram yang tertarik untuk ikut mengembangkan *PHP*. Pada November 1997, dirilis *PHP/FI* 2.0. Pada rilis ini, interpreter *PHP* sudah diimplementasikan dalam program *C*. Dalam rilis ini disertakan juga modul-modul ekstensi yang meningkatkan kemampuan *PHP/FI* secara signifikan. Pada tahun 1997, sebuah perusahaan bernama Zend menulis ulang interpreter *PHP* menjadi lebih bersih, lebih baik, dan lebih cepat. Kemudian pada Juni 1998, perusahaan tersebut merilis interpreter baru untuk *PHP* dan meresmikan rilis tersebut sebagai PHP 3.0 dan singkatan *PHP* diubah menjadi akronim berulang *PHP*: *Hypertext Preprocessing*. Pada pertengahan tahun 1999, Zend merilis interpreter *PHP* baru dan rilis tersebut dikenal dengan *PHP* 4.0. *PHP* 4.0 adalah versi *PHP* yang paling banyak dipakai

pada awal abad ke-21. Versi ini banyak dipakai disebabkan kemampuannya untuk membangun aplikasi *web* kompleks tetapi tetap memiliki kecepatan dan stabilitas yang tinggi. Pada Juni 2004, Zend merilis *PHP* 5.0. Dalam versi ini, inti dari interpreter *PHP* mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam *PHP* untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek. *Server web* bawaan ditambahkan pada versi 5.4 untuk mempermudah pengembang menjalankan kode *PHP* tanpa menginstall *software* server. Versi terbaru dan stabil dari bahasa pemograman *PHP* saat ini adalah versi 7.0.16 dan 7.1.2 yang resmi dirilis pada tanggal 17 Februari 2017 (Hadi, 2013).

3.7 REST API

REST (*Representational State Transfer*) merupakan standar arsitektur komunikasi berbasis *web* yang sering diterapkan dalam pengembangan layanan berbasis *web*. Umumnya menggunakan *HTTP* (*Hypertext Transfer Protocol*) sebagai *protocol* untuk komunikasi data. *REST* pertama kali diperkenalkan oleh Roy Fielding pada tahun 2000. Pada arsitektur *REST*, *REST server* menyediakan *resources* (sumber daya/data) dan *REST client* mengakses dan menampilkan *resource* tersebut untuk penggunaan selanjutnya. Setiap *resource* diidentifikasi oleh *URIs* (*Universal Resource Identifiers*) atau *global* ID. *Resource* tersebut direpresentasikan dalam bentuk format teks, *JSON* atau *XML*. Pada umumnya formatnya menggunakan *JSON* dan *XML*.

1) Keuntungan REST

   Bahasa dan *platform* agnostic lebih sederhana/simpel untuk dikembangkan ketimbang SOAP mudah dipelajari, tidak bergantung pada tools ringkas, tidak membutuhkan layer pertukaran pesan (*messaging*) tambahan secara desain dan filosofi lebih dekat dengan *web.*

2) Kelemahan REST

   Mengasumsi *model point-to-point* komunikasi - tidak dapat digunakan untuk lingkungan komputasi terdistribusi di mana pesan akan melalui satu atau lebih perantara. Kurangnya dukungan standar untuk keamanan,

kebijakan, keandalan pesan, dll, sehingga layanan yang mempunyai persyaratan lebih canggih lebih sulit untuk dikembangkan ("dipecahkan sendiri"). Berkaitan dengan *model transport* HTTP.

Berikut ini beberapa metode *HTTP* yang umum digunakan dalam arsitektur berbasis *REST* yaitu:

a) *GET*, menyediakan hanya akses baca pada *resource.*
b) *PUT*, digunakan untuk menciptakan *resource* baru.
c) *DELETE*, digunakan untuk menghapus *resource.*
d) *POST*, digunakan untuk memperbarui *resource* yang ada atau membuat *resource* baru.
e) *OPTIONS*, digunakan untuk mendapatkan operasi yang di *support* pada *resource.*

Web service adalah standar yang digunakan untuk melakukan pertukaran data antar aplikasi atau sistem, karena aplikasi yang melakukan pertukaran data bisa ditulis dengan bahasa pemrograman yang berbeda atau berjalan pada platform yang berbeda. Contoh implementasi dari web *service* antara lain adalah *SOAP* dan *REST*. *Web service* yang berbasis arsitektur *REST* kemudian dikenal sebagai *RESTFUL web services*. Layanan *web* ini menggunakan metode *HTTP* untuk menerapkan konsep arsitektur *REST*.

Sebuah *client* mengirimkan sebuah data atau *request* melalui *HTTP Request* dan kemudian *server* merespon melalui *HTTP Response*. Komponen dari *HTTP request*: *Verb*, *HTTP method* yang digunakan misalnya *GET, POST, DELETE*, *PUT* dll. *Uniform Resource Identifier* (*URI*) untuk mengidentifikasikan lokasi *resource* pada *server*. *HTTP Version*, menunjukkan versi dari *HTTP* yang digunakan, contoh HTTP v1.1. *Request Header*, berisi metadata untuk *HTTP Request*. Contoh, *type client*/*browser*, format yang didukung oleh *client*, format dari *body* pesan, seting *cache* dll. *Request Body*, konten dari data. Sedangkan komponen dari *http response*: Status/*Response Code*, mengindikasikan status *server* terhadap *resource* yang di *request*. misal: 404, artinya *resource* tidak ditemukan dan 200 *response OK*. *HTTP Version*, menunjukkan versi dari *HTTP* yang digunakan, contoh *HTTP* v1.1. *Response Header*, berisi metadata untuk *HTTP Response*. Contoh, *type server*,

panjang *content*, tipe *content*, waktu *response*, dll. *Response Body*, konten dari data yang diberikan (Feridi, 2019).

3.8 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 (Hyderabad, 2012).

3.9 Android Studio

Android Studio adalah *Integrated Development Enviroment* (*IDE*) untuk sistem operasi *Android*, yang dibangung diatas perangkat lunak *JetBrains IntelliJ* IDEA dan didesain khusus untuk pengembangan *Android*. *IDE* ini merupakan pengganti dari *Eclipse Android Development Tools* (*ADT*) yang sebelumnya merupakan *IDE* utama untuk pengembangan aplikasi *android*. *Android studio* sendiri pertama kali diumumkan di *Google* I/O *conference* pada tanggal 16 Mei 2013. Ini merupakan tahap *preview* dari versi 0.1 pada Mei 2013, dan memasuki tahap beta sejak versi 0.8 dan mulai diliris pada Juni 2014. Versi liris stabil yang pertama diliris pada *December* 2014, dimulai sejak versi 1.0. Sedangkan versi stabil yang sekarang adalah versi 3.13 yang diliris pada Juni 2018. Fitur Fitur yang tersedia saat ini dalam *stable version* (Dobie, 2013).

3.10 Emulator

Emulator atau lebih tepatnya peranti lunak *emulator* memungkinkan suatu program atau peranti lunak yang dibuat pada awalnya oleh suatu sistem komputer (arsitektur dan sistem operasi) dan untuk dijalankan dalam sistem itu (atau

dijalankan dalam suatu sistem yang didedikasikan), dapat dijalankan dalam sistem komputer yang sama sekali berbeda. Sebagai contoh suatu program *Windows* dapat dijalankan di sistem operasi *Linux* dengan menggunakan peranti lunak *emulator Wine*. Ada pula program yang mengemulasikan suatu komputer dalam komputer, misalnya *VMware*. Contoh lain adalah program-program *emulator* untuk menjalankan permainan komputer yang awalnya hanya bisa dijalankan pada konsolnya masing-masing, misalnya *Nintendo*, *Atari*, *PlayStation*, *XBox* dan lain-lain (Seebach, 2004)

3.11 MYSQL

MYSQL adalah sebuah perangkat lunak sistem manajemen basis data *SQL* (bahasa Inggris: *database management system*) atau *DBMS* yang multi alur, multi pengguna, dengan sekitar 6 juta instalasi di seluruh dunia. *MySQL* tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (*GPL*), tetapi mereka juga menjual di bawah lisensi komersial untuk kasus-kasus di mana penggunaannya tidak cocok dengan penggunaan *GPL*. Tidak sama dengan proyek-proyek seperti *Apache*, di mana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, *MySQL* dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia, di mana memegang hak cipta hampir atas semua kode sumbernya (Kadir, 2013).

3.12 JAVASCRIPT

JavaScript adalah bahasa pemrograman tingkat tinggi dan dinamis. *JavaScript* populer di *internet* dan dapat bekerja di sebagian besar penjelajah *web* populer seperti *Google Chrome*, *Internet Explorer* (*IE*), *Mozilla Firefox*, *Netscape* dan Opera. Kode *JavaScript* dapat disisipkan dalam halaman *web* menggunakan tag *SCRIPT*. *JavaScript* merupakan salah satu teknologi inti *World Wide Web* selain *HTML* dan *CSS*. *JavaScript* membantu membuat halaman *web* interaktif dan merupakan bagian aplikasi *web* yang esensial. Awalnya hanya diimplementasi sebagai *client-side* dalam penjelajah *web*, kini *engine JavaScript* disisipkan ke dalam perangkat lunak lain seperti dalam *server-side* dalam *server web* dan basis data, dalam program *non web* seperti perangkat lunak pengolah kata dan pembaca

PDF, dan sebagai *runtime environment* yang memungkinkan penggunaan *JavaScript* untuk membuat aplikasi *desktop* maupun *mobile* (Flanagan, 2011)

3.13 AJAX

Asynchronous JavaScript and XMLHTTP, atau disingkat *AJaX*, adalah suatu teknik pemrograman berbasis *web* untuk menciptakan aplikasi *web* interaktif. Tujuannya adalah untuk memindahkan sebagian besar interaksi pada komputer *web surfer*, melakukan pertukaran data dengan *server* di belakang layar, sehingga halaman *web* tidak harus dibaca ulang secara keseluruhan setiap kali seorang pengguna melakukan perubahan. Hal ini akan meningkatkan interaktivitas, kecepatan, dan *usability* (Sunyoto, 2007).

3.14 FIREBASE

Firebase adalah suatu layanan dari *Google* yang digunakan untuk mempermudah para pengembang aplikasi dalam mengembangkan aplikasi. Dengan adanya *Firebase*, pengembang aplikasi bisa fokus mengembangkan aplikasi tanpa harus memberikan usaha yang besar. Dua fitur yang menarik dari *Firebase* yaitu *Firebase Remote Config* dan *Firebase Realtime Database*. Selain itu terdapat fitur pendukung untuk aplikasi yang membutuhkan pemberitahuan yaitu *Firebase Notification* (Firebase, 2016)

3.15 GMAIL

Gmail adalah layanan surel milik *Google*. Pengguna dapat mengakses Gmail dalam bentuk surat *web HTTPS*, protokol POP3 atau IMAP4. *Gmail* diluncurkan dengan sistem undangan dalam bentuk Beta pada 1 April 2004 dan tersedia untuk publik pada 7 Februari 2007 meski masih menyandang status Beta. Bersama seluruh produk *Google Apps*, layanan ini tidak lagi Beta pada 7 Juli 2009. Dengan kapasitas penyimpanan awal 1 GB per pengguna, *Gmail* berhasil meningkatkan standar penyimpanan gratis surat *web* dari 2-4 MB yang ditawarkan para pesaingnya pada waktu itu. Pesan pribadi, termasuk lampiran, dibatasi hingga 25 MB, lebih besar daripada layanan surat *web* lainnya. *Gmail* memiliki antar muka berorientasi pencarian dan "tampilan percakapan" yang mirip dengan forum

Internet. Sejumlah pengembang *web* mengakui Gmail adalah layanan pertama yang memakai metode pemrograman Ajax. *Gmail* beroperasi dengan *Google* GFE/2.0 di *Linux*. Hingga Juni 2012, *Gmail* adalah layanan surat elektronik berbasis web terbesar dengan 425 juta pengguna aktif di seluruh dunia (Smith, 2014)

3.16 FACEBOOK

Facebook, *Inc*. adalah sebuah layanan jejaring sosial berkantor pusat di Menlo Park, California, Amerika Serikat yang diluncurkan pada bulan Februari 2004. Hingga September 2012, Facebook memiliki lebih dari satu miliar pengguna aktif, lebih dari separuhnya menggunakan telepon genggam. Pengguna harus mendaftar sebelum dapat menggunakan situs ini. Setelah itu, pengguna dapat membuat profil pribadi, menambahkan pengguna lain sebagai teman, dan bertukar pesan, termasuk pemberitahuan otomatis ketika mereka memperbarui profilnya (Sengupta, 2012).

3.17 HTTP

Hypertext Transfer Protocol (*HTTP*) adalah sebuah protokol jaringan lapisan aplikasi yang digunakan untuk sistem informasi terdistribusi, kolaboratif, dan menggunakan hipermedia. Penggunaannya banyak pada pengambilan sumber daya yang saling terhubung dengan tautan, yang disebut dengan dokumen hiperteks, yang kemudian membentuk *World Wide Web* pada tahun 1990 oleh fisikawan Inggris, Tim Berners-Lee. Hingga kini, ada dua versi mayor dari protokol *HTTP*, yakni *HTTP*/1.0 yang menggunakan koneksi terpisah untuk setiap dokumen, dan *HTTP*/1.1 yang dapat menggunakan koneksi yang sama untuk melakukan transaksi. Dengan demikian, *HTTP*/1.1 bisa lebih cepat karena memang tidak perlu membuang waktu untuk pembuatan koneksi berulang-ulang. Pengembangan standar *HTTP* telah dilaksanakan oleh Konsorsium *World Wide Web* (*World Wide Web Consortium*/W3C) dan juga *Internet Engineering Task Force* (*IETF*), yang berujung pada publikasi beberapa dokumen *Request for Comments* (*RFC*), dan yang paling banyak dirujuk adalah *RFC* 2616 (yang dipublikasikan pada bulan Juni 1999), yang mendefinisikan *HTTP*/1.1. Dukungan untuk *HTTP*/1.1 yang belum disahkan, yang pada waktu itu RFC 2068, secara cepat diadopsi oleh banyak

pengembang penjelajah *Web* pada tahun 1996 awal. Hingga Maret 1996, *HTTP*/1.1 yang belum disahkan itu didukung oleh *Netscape* 2.0, *Netscape Navigator Gold* 2.01, Mosaic 2.7, Lynx 2.5, dan dalam *Microsoft Internet Explorer* 3.0. Adopsi yang dilakukan oleh pengguna akhir penjelajah *Web* pun juga cepat. Pada bulan Maret 2006, salah satu perusahaan *Web hosting* melaporkan bahwa lebih dari 40% dari penjelajah *Web* yang digunakan di *Internet* adalah penjelajah *Web* yang mendukung *HTTP*/1.1. Perusahaan yang sama juga melaporkan bahwa hingga Juni 1996, 65% dari semua penjelajah yang mengakses server-server mereka merupakan penjelajah *Web* yang mendukung *HTTP*/1.1. Standar *HTTP*/1.1 yang didefinisikan dalam *RFC* 2068 secara resmi dirilis pada bulan Januari 1997. Peningkatan dan pembaruan terhadap standar *HTTP*/1.1 dirilis dengan dokumen *RFC* 2616 pada bulan Juni 1999 (Fielding, 2009).

3.18 Analisis Deskriptif

Analisis Deskriptif adalah analisis yang dilakukan untuk menilai karakteristik dari sebuah data. Karakterisitik itu banyak sekali, antara lain: nilai *Mean*, *Median*, *Sum*, *Variance*, Standar error, *standar error of mean*, *mode*, *range* atau rentang, minimal, maksimal, *skewness* dan kurtosis (Hidayat, 2012)

3.19 Metode Waterfall

Metode air terjun atau yang sering disebut metode *waterfall* sering dinamakan siklus hidup klasik (*classic life cycle*), dimana hal ini menggambarkan pendekatan yang sistematis dan juga berurutan pada pengembangan perangkat lunak, dimulai dengan spesifikasi kebutuhan pengguna lalu berlanjut melalui tahapan-tahapan perencanaan (*planning*), permodelan (*modeling*), konstruksi (*construction*), serta penyerahan sistem ke para pelanggan/pengguna (*deployment*), yang diakhiri dengan dukungan pada perangkat lunak lengkap yang dihasilkan (Pressman, 2012).