BAB II TINJAUAN PUSTAKA DAN DASAR TEORI

2.1 Tinjauan Pustaka

Penelitian ini menggunakan beberapa sumber pustaka yang berhubungan dengan kasus atau metode yang akan diteliti. Diantaranya sebagai berikut :

Alam Rahmatulloh, Heni Sulastri dan Rizal Nugroho (2018) dalam penelitiannya membangun sistem keamanan RESTful web service menggunakan JSON Web Token (JWT) HMAC SHA-512 dengan tujuan membuktikan bahwa JWT dengan algoritma HMAC SHA-512 lebih baik dari pada JWT dengan algoritma HMAC SHA-256.

Edy, Ferdiansyah, Wahyu Pramusinto dan Sejati Waluyo (2019) dalam penelitiannya pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order dengan tujuan menerapkan teknologi REST yang dapat digunakan oleh berbagai client seperti aplikasi mobile, aplikasi web dan aplikasi desktop.

Mochammad Rizky Royani dan Arief Wibowo (2020) dalam penelitiannya mengimplementasikan web service pada Perusahaan Logistik menggunakan JSON Web Token dan Algoritma Kriptografi RC4 dengan tujuan membangun web service dengan pendekatan RESTful.

Andi Setiawan dan Ade Irma Purnamasari (2020) dalam penelitiannya mengimplementasikan JSON Web Token Berbasis Algoritma SHA-512 untuk Otentikasi Aplikasi BatikKita yang bertujuan untuk mengimplementasikan JWT dengan algoritma SHA-512 pada aplikasi BatikKita.

Ilyas Mahfud, Putranto Hadi Utomo (2021) dalam penelitiannya mengimplementasikan sistem kriptografi RSA Signature dengan SHA-256 pada Mekanisme Autentikasi REST API dengan tujuan menerapkan sistem kriptografi RS256 pada mekanisme autentikasi REST API.

Perbandingan dari penelitian ini dengan penelitian sebelumnya adalah adanya kesamaan dalam penggunaan JWT (JSON WEB Token), namun pada penelitian ini lebih difokuskan kepada pengamanan autentikasi RESTful API aplikasi food delivery.

Tabel 2. 1 Tinjauan Pustaka

NO	Penulis	Obyek Penelitian	Metode/ Teknologi	Keterangan
1	Alam Rahmatulloh, Heni Sulastri, Rizal Nugroho (2018)	Keamanan RESTful Web Service Menggunakan JSON Web Token (JWT) HMAC SHA-512	JWT, Algoritma HMAC SHA-512	Jurnal Penelitian
2	Edy, Ferdiansyah, Wahyu Pramusinto, Sejati Waluyo (2019)	Pengamanan Restful API menggunakan JWT untuk Aplikasi Sales Order	Waterfall, MariaDB, JWT, Algoritma SHA-256	Jurnal Penelitian
3	Mochammad Rizky Royani, Arief Wibowo (2020)	Implementasi Web Service pada Perusahaan Logistik menggunakan JSON Web Token dan Algoritma Kriptografi RC4	Studi literatur, JWT, Algoritma kriptografi RC-4	Jurnal Penelitian
4	Andi Setiawan, Ade Irma Purnamasari (2020)	Implementasi JSON Web Token Berbasis Algoritma SHA-512 untuk Otentikasi Aplikasi BatikKita	RAD, Laravel, JWT, Algoritma HMAC SHA-512	Jurnal Penelitian
5	Ilyas Mahfud, Putranto Hadi Utomo (2021)	Implementasi Sistem Kriptografi RSA Signature dengan SHA-256 pada Mekanisme Autentikasi REST API	Java Spring Boot, JWT, Algoritma RSA signature SHA 256 (RS-256)	Jurnal Penelitian
6	Wildan Ibnu Sina	Implementasi Keamanan Authentication Restful Web Service Menggunakan JWT Pada Aplikasi Food Delivery	Laravel, JWT, Algoritma RSA signature SHA 256 (RS-256), MySQL.	Jurnal Penelitian

2.2 Dasar Teori

2.1.1 Web Service

Layanan *web service* merupakan sebuah sistem yang didesain untuk mendukung interaksi antara sistem pada aplikasi di sebuah jaringan. Layanan *web service* memberikan layanan berupa informasi antar sistem sehingga antara bagian di dalam sistem layanan *web service* dapat saling berinteraksi. Layanan *web service* dapat juga diartikan sebagai antarmuka yang mengilustrasikan beberapa operasi yang dapat diakses melalui jaringan. Contoh arsitektur layanan web termasuk *Simple Object Access Protocol* (SOAP) dan *Representational State Transfer* (REST).

2.1.2 RESTful

RESTful atau REST (*Representational State Transfer*) adalah standar arsitektur atau gaya komunikasi yang menggunakan penamaan jalur API yang menggunakan protokol Hypertext *Transfer Protocol* (HTTP) untuk mengirim dan menerima data. REST bersifat *stateless* yang artinya pesan tidak bergantung pada keadaan percakapan.

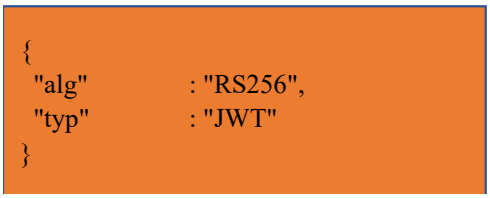
Arsitektur REST digunakan untuk memanipulasi data pada sebuah sistem dengan menggunakan metode protokol HTTP seperti GET, POST, DELETE dan lain sebagainya. Data diidentifikasi dengan *Uniform Resource Locator* (URL) untuk digunakan sebagai antar muka dalam memanipulasi sumber daya. Dalam arsitektur REST kembalian sumber daya dapat berupa format XML, HTML, JSON ataupun format yang lain. Dengan menggunakan *protocol* HTTP/HTTPS yang

bersifat *stateless*, arsitektur REST ditujukan untuk *performance*, *reliability* dan *scalability*.

2.1.3 JWT

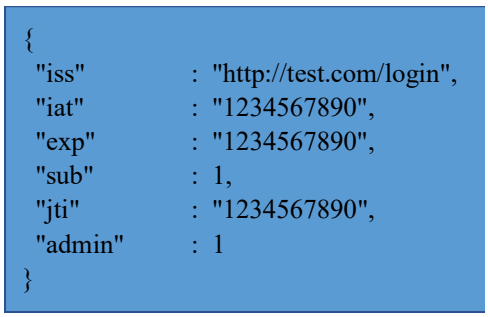
JWT adalah format standar dalam mengamankan informasi pribadi menjadi sebuah klaim yang dienkripsi hingga berbentuk JSON dan menjadi muatan untuk *JSON Web Token* (JWT). Klaim akan terlindungi dengan adanya tanda tangan digital seperti *Message authentication code* (MAC) atau di enkripsi. JWT tersusun atas 3 bagian yaitu *header*, *payload* dan *signature*.

1. Bagian *header* berisi informasi tentang *signature* JWT untuk proses validasi seperti algoritma kriptografi yang dipakai dan tipe token.



Gambar 2. 1 Header Pada JWT

2. *Payload* pada JWT merupakan data entitas pengguna yang dimasukkan ke dalam JWT.



Gambar 2. 2 Payload Pada JWT

3. Pada *Signature* JWT digunakan untuk membuat nilai *digest* dari *header* dan *payload*. Bentuk akhir dari token JWT berupa *string* yang menggunakan *encoding* base64url.

```
RSASHA256 (
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key,
  Private Key
)
```

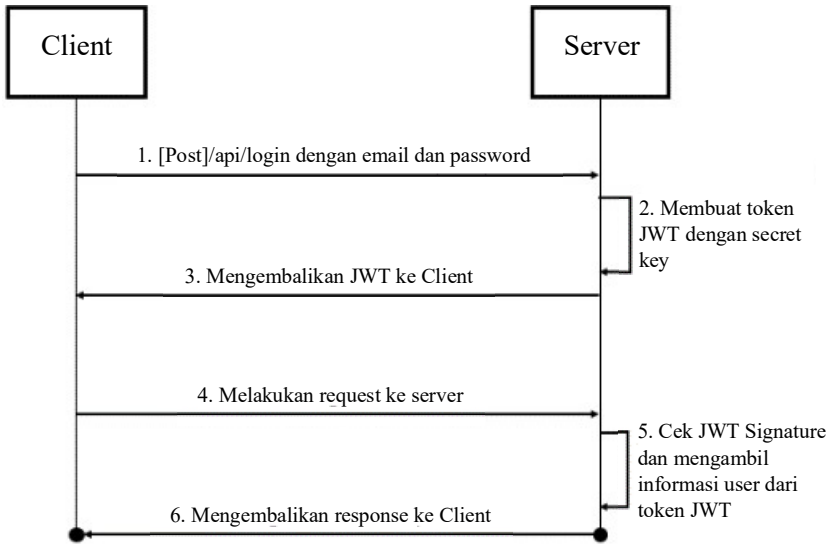
Gambar 2. 3 Signature Pada JWT

Header, Payload dan *Signature* yang sudah di *encode* digabungkan dengan titik dan dikirim ke *client* untuk disimpan di *client*.

Gambar 2.4 merupakan gambar ilustrasi dari bagian-bagian token JWT.

Gambar 2. 4 Tiga Bagian JWT Token

Cara kerja dari JWT dapat dilihat pada Gambar 2.5 di bawah ini.



Gambar 2. 5 Cara Kerja JWT

Proses yang terjadi pada Gambar 2.5 bisa dijelaskan sebagai berikut :

1) Pertama *client* melakukan *request login* dengan data *username* dan *password.*

2) Jika data benar, server akan membuat *token* JWT menggunakan *secret key.*

3) Kemudian server akan mengembalikan *token* JWT sebagai *response* ke *client*.

4) *Token* disimpan di *client*, jika *client* ingin melakukan *request* lain maka *token* juga dikirim melalui *request*.

5) Server akan memverifikasi token JWT dan mengambil informasi pengguna dari token.

6) Jika Token benar, server mengembalikan *response* yang di minta *client*.

2.1.4 Kriptografi

Kriptografi menurut terminologi adalah ilmu dan seni dalam melindungi pesan. kriptografi sebagai ilmu yang mempelajari teknik matematika yang berkaitan dengan aspek keamanan informasi seperti aspek kerahasiaan, integritas data, autentikasi pengirim atau penerima pesan, dan validitas pesan.

Data atau pesan dalam kriptografi disebut plain text. Kemudian proses menyamarkan data/pesan disebut enkripsi. Data yang dienkripsi disebut dengan ciphertext. Kebalikan dari enkripsi atau proses pengembalian ciphertext ke plaintext disebut dekripsi. Dalam proses enkripsi dan dekripsi diperlukan kunci rahasia untuk mendapatkan ciphertext dan begitu juga pada proses dekripsi.

2.1.5 RS-256

RS-256 adalah kependekan dari RSA signature SHA-256 dan merupakan algoritma asimetris yang menggunakan pasangan kunci publik-pribadi, dengan

penyedia layanan memiliki kunci pribadi (rahasia) untuk menghasilkan signature atau tanda tangan, dan konsumen JWT mendapatkan kunci publik untuk memvalidasi tanda tangan.

2.1.6 RSA

Algoritma RSA (Rivest-Shamir-Adleman) adalah algoritma kriptografi kunci publik yang pengaplikasiannya sudah sangat luas terutama dalam mengamankan pengiriman data. Algoritma RSA diciptakan oleh tiga orang peneliti yaitu Ronald Rivest, Adi Shamir, dan Len Adleman.

Pada algoritma kunci publik enkripsi data dilakukan dengan menggunakan sebuah kunci publik yang dapat disebarkan ke siapapun khususnya pengirim data, sedangkan dekripsi data dilakukan dengan sebuah kunci privat yang hanya diketahui oleh pihak penerima data. Algoritma RSA memanfaatkan tingkat kerumitan dalam memfaktorkan dua bilangan prima yang sangat besar untuk menjaga kerahasiaan data.

Inti dari algoritma RSA merupakan penurunan dari Teorema Euler yaitu :

$$a^{\phi(n)} \equiv 1(mod\ n)$$

Dengan syarat yaitu:

1. Nilai α harus relatif prima dengan n.
2. Totient Euler atau $\phi(n)$ adalah fungsi yang menentukan jumlah bilangan yang relatif prima terhadap n.

Dalam implementasinya, algoritma RSA memiliki dua prosedur penting. Pertama terdapat prosedur pembangkitan kunci yang akan menghasilkan kunci privat dan kunci publik serta prosedur enkripsi dan dekripsi data.

Agar dapat melakukan enkripsi dan dekripsi data, maka diperlukan sepasang kunci yang akan menjadi salah satu masukan pada algoritma enkripsi dan dekripsi. Berikut merupakan tahapan prosedur pembangkitan kunci pada algoritma RSA.

1) Pilih dua buah bilangan prima acak yang akan disimpan sebagai variabel p dan q. Nilai bilangan prima acak ini bersifat rahasia dan dicari nilai sebesar mungkin agar semakin sulit untuk ditebak.
2) Hitung nilai n dengan menggunakan rumus:

   $$n = p * q$$

3) Lalu, hitung nilai Totient Euler dengan menggunakan rumus :

   $$\phi(n) = (p - 1) * (q - 1)$$

   Nilai Totient Euler ini merupakan properti yang bersifat rahasia.
4) Pilih sebuah nilai e yang akan menjadi kunci publik. Nilai e harus relatif prima dengan nilai Totient Euler. Relatif prima berarti nilai FPB dari Totient Euler dan e = 1.
5) Hitung nilai d yang akan menjadi kunci privat. Mencari nilai d dengan menggunakan rumus:

   $$d \equiv e^{-1} mod\ \phi(n)$$

Dari algoritma tersebut, sudah berhasil dibangkitkan kunci publik dan kunci privat. Kunci publik terdiri dari (e) dan kunci privat terdiri dari (d). Prosedur

selanjutnya pada algoritma RSA adalah prosedur enkripsi. Berikut merupakan tahapan prosedur enkripsi pada algoritma RSA.

1) Pisahkan *plaintext* atau data yang akan dienkripsi menjadi blok-blok *plaintext* atau data.

   $m_1, m_2, m_3, \dots, m_i$

   Dengan syarat $0 \leq m_1 < n - 1$.

2) Hitung blok *ciphertext* c_i untuk setiap blok *plaintext* m_i dengan menggunakan kunci publik (e) dengan rumus :

   $c_i = \mathrm{m}_i^e \ mod \ n$

Untuk melakukan dekripsi dari *ciphertext* kembali menjadi *plaintext*. Dapat digunakan algoritma yang sama dengan menggunakan kunci privat (d). Berikut merupakan tahapan prosedur dekripsi pada algoritma RSA.

1) Pisahkan cipherteks yang akan didekripsi menjadi blokblok pesan atau data.

   $c_1, c_2, c_3, \dots, c_i$

2) Hitung blok *plaintext* mi untuk setiap blok *ciphertext* c_i dengan menggunakan kunci privat (d) dengan rumus :

   $\mathrm{m}_i = \mathrm{c}_i^d \ mod \ n$

2.1.7 SHA-256

Secure Hash Algorithm 256-*bit* (SHA-256) adalah fungsi *hash* yang sering dipakai agar tidak ada kemungkinan terjadi *collision resistance* (pemberian input data berbeda namun memiliki nilai hash yang sama). SHA-256 memiliki 8 langkah pengerjaan sebagai berikut.

1) Tambahkan bit *Padding*

   Pesan diubah menjadi desimal dan diisi sehingga panjangnya kongruen dengan 448 *modulus* 512. *Padding* 1 bit ditambahkan di akhir pesan, diikuti oleh banyaknya 0 yang diperlukan sehingga panjang bit sama dengan 448 *modulus* 512.

2) Panjang *Append*

   Representasi panjang pesan 64bit ditambahkan pada hasil akhirnya, langkah ini untuk membuat panjang pesan kelipatan 512 bit.

3) *Parsing* Pesan

   Pesan *padding* diuraikan menjadi N blok pesan 512 bit, M_1, M_2,...M_N, dengan menambahkan blok 64 bit.

4) Inisialisasi Nilai *Hash*

   Nilai *hash* awal sudah diatur dan terdiri dari delapan kata 32 bit, dalam bentuk heksadesimal.

Tabel 2. 2 Initial Hash Value

Variabel	Initial Hash Value
$H_0^{(0)}$	6A09E667
$H_1^{(0)}$	BB67EA85
$H_2^{(0)}$	3C6EF372
$H_3^{(0)}$	A54FF53A
$H_4^{(0)}$	510E527F
$H_5^{(0)}$	9B05688C
$H_6^{(0)}$	1F83D9AB
$H_7^{(0)}$	5BE0CD19

5) Mempersiapkan jadwal pesan

SHA-256 menggunakan jadwal pesan 64 kata 32 bit, kata-kata dari jadwal pesan diberi label W_0, W_1,..., W_{63} dengan rumusan baku SHA-256 sebagai berikut.

$$W_t = \begin{cases} M_t^{(t)} & ,0 \leq t \leq 15 \\ \sigma_1^{(256)}(W_{i-2}) + W_{i-7} + \sigma_0^{(256)}(W_{i-15}) + W_{i-16} & ,16 \leq t \leq 63 \end{cases}$$

Dengan fungsi σ_0 dan σ_1 dirumuskan sebagai berikut. :

$$\sigma_1^{(256)}(W_{i-2}) = ((W_{i-2})ROTR^{17}) \oplus ((W_{i-2})ROTR^{19}) \oplus ((W_{i-2})SHR^{10})$$

$$\sigma_0^{(256)}(W_{i-15}) = ((W_{i-15})ROTR^{7}) \oplus ((W_{i-15})ROTR^{18}) \oplus ((W_{i-15})SHR^{3})$$

Keterangan :

W_t = Blok pesan yang baru

M_t = Blok pesan yang lama

W_{i-2} = Blok data/pesan,dari W ke $i-2$

W_{i-15} = Blok data/pesan,dari W ke $i-15$

$ROTR$ = *Rotate right*

SHR = *Shift right*

$\oplus$ = Operator XOR

6) Inisialisasi delapan variabel kerja *a, b, c, d, e, f, g,* dan *h* dengan nilai *hash* (i-1).

For t = 0 to 63

{

$T_1 = h + \sum_1^{(256)}(e) + Ch(e, f, g) + K_1^{(256)} + W_t$

$T_2 = \sum_0^{(256)}(a) + Maj(a, b, c)$

$h = g$

$g = f$

$f = e$

$e = d + T1$

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

}

Keterangan :

$\sum_1^{(256)}(e)$	= $(e\ ROTR\ 6) \oplus (\ e\ ROTR\ 11) \oplus (\ e\ ROTR\ 25)$
$\sum_0^{(256)}(a)$	= $(a\ ROTR\ 2) \oplus (\ a\ ROTR\ 13) \oplus (\ a\ ROTR\ 22)$
$Ch(e, f, g)$	= $(e \wedge f) \oplus (\sim e \wedge g)$
$Maj(a, b, c)$	= $(a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c)$
a, b, c, d, e, f, g, h	= Variabel yang berisi pesan heksadesimal
$K_1^{(256)}$	= Konstanta SHA-256

ROTR = *Rotate Right*

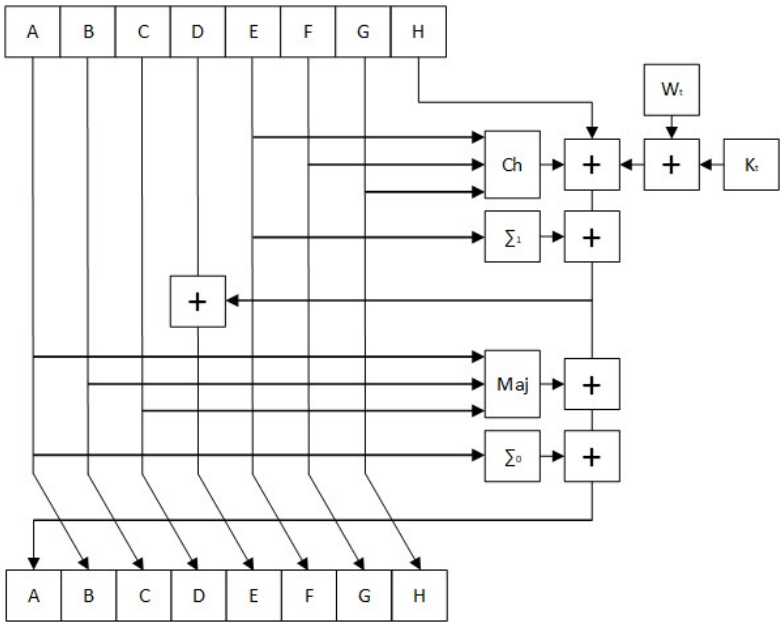
$\oplus$ = Operator XOR

^ = Operator AND

Tabel 2. 3 Konstanta SHA-256

428A2F98	71377791	B5C0FBCF	E9B5DBA5
3956C25B	59F111F1	923F82A4	AB1C5ED5
D807AA98	12835B01	243185BE	550C7DC3
72BE5D74	80DEB1FE	9BDC06A7	C19BF174
E49B69C1	EFBE4786	0FC19DC6	240CA1CC
2DE92C6F	4A7484AA	5CB0A9DC	76F988DA
983E5152	A831C66D	B00327C8	BF597FC7
C6E00BF3	D5A79147	06CA6351	14292967
27B70A85	2E1B2138	4D2C6DFC	53380D13
650A7354	766A0ABB	81C2C92E	92722C85
A2BFE8A1	A81A664B	C24B8B70	C76C51A3
D192E819	D6990624	F40E3585	106AA070
19A4C116	1E376C08	2748774C	34B0BCB5
3 91C0CB3	4ED8AA4A	5B9CCA4F	682E6FF3
74 8F82EE	78A5636F	84C87814	8CC70208
90 BEFFFA	A4506CEB	BEF9A3F7	C67178F2

Di bawah ini merupakan Ilustrasi fungsi yang dieksekusi untuk tiap ronde.



Gambar 2. 6 Ilustrasi Fungsi Yang Dieksekusi Untuk Tiap Ronde

7) Menjumlahkan hasil akhir *a, b, c, d, e, f, g*, *h* dengan inisial *hash value* $H^{(i)}$

$$\mathbf{H_0^{(i)}} = a + H_0^{(i)}$$

$$\mathbf{H_1^{(i)}} = b + H_1^{(i)}$$

$$\mathbf{H_2^{(i)}} = c + H_2^{(i)}$$

$$\mathbf{H_3^{(i)}} = d + H_3^{(i)}$$

$$\mathbf{H_4^{(i)}} = e + H_4^{(i)}$$

$$\mathbf{H_5^{(i)}} = f + H_5^{(i)}$$

$$\mathbf{H_6^{(i)}} = g + H_6^{(i)}$$

$$\mathbf{H_7^{(i)}} = h + H_7^{(i)}$$

8) Output fungsi *hash* yang dihasilkan adalah sebagai berikut :

$$H_0^{(N)} \| H_1^{(N)} \| H_2^{(N)} \| H_3^{(N)} \| H_4^{(N)} \| H_5^{(N)} \| H_6^{(N)} \| H_7^{(N)}$$

2.1.8 Laravel

Laravel merupakan framework yang menggunakan Bahasa pemrograman PHP dalam proses pengembangannya karena PHP menjadi Bahasa pemrograman yang dinamis menjadikan laravel sebuah framework yang lebih *powerfull*, cepat, aman dan *simple*. Setiap perilisan versi laravel terbaru selalu menghadirkan fitur atau teknologi terbaru menjadikan framework yang paling banyak diminati oleh para *developer* di github (tahun 2015). Laravel focus dibagian *end-user* yang berarti fokus pada kejelasan dan kesederhanaan, baik penulisan maupun tampilan, serta menghasilkan fungsionalitas aplikasi web yang bekerja sama sebagaimana mestinya. Hal ini membuat para *developer* maupun perusahaan menggunakan framework ini untuk membangun sebuah proyek website dari skala kecil hingga skala atas. Laravel mengubah pengembangan website menjadi lebih elegan, ekspresif, dan menyenangkan. Selain itu laravel lebih mudah dalam proses pengembangannya dengan menerapkan konsep MVC (*Model, View, Control*).

2.1.9 Framework

Framework adalah sebuah kerangka kerja yang digunakan untuk mempermudah *developer software* dalam membuat dan mengembangkan aplikasi. Framework berisikan fungsi dasar dan perintah yang dipakai untuk membuat dan mengembangkan sebuah aplikasi dengan harapan aplikasi yang dibuat dapat dibangun secara lebih terstruktur, lebih cepat serta tersusun dengan rapi.

2.1.10 PHP

PHP atau *Hypertext Preprocessor* adalah bahasa pemrograman *script server side* yang sengaja dirancang lebih cenderung untuk membuat dan mengembangkan

web. Bahasa pemrograman ini dirancang untuk pengembang web agar dapat menciptakan suatu halaman web yang bersifat dinamis.

PHP diciptakan oleh Rasmus Lerdorf seorang pemrogram C, dan digunakan untuk *mencatat* jumlah pengunjung pada *homepage.* Pada awal tahun 1995 dirilis PHP/FI (*Form Interpreter*) yang memiliki kemampuan dasar membangun aplikasi web, memproses *form* dan mendukung data MySQL.

2.1.11 JavaScript

JavaScript merupakan *script* program berbasis *client* yang dieksekusi oleh browser *sehingga* membuat halaman web melakukan tugas-tugas tambahan yang tidak bisa dilakukan oleh *script* HTML biasa.

2.1.12 CSS

CSS adalah singkatan dari *Cascading Style Sheets*, yaitu bahasa yang digunakan untuk *menentukan* tampilan dan format halaman website. CSS menghemat banyak pekerjaan dan sudah pasti mengontrol tata letak beberapa halaman web sekaligus.

2.1.13 MySQL

MySQL (*My Structured Query Language*) adalah database yang paling favorit saat ini. Program ini berjalan sebagai server yang menyediakan *multi-user*, mengakses ke sejumlah database baik *multithread* maupun *multi-user*. MySQL *termasuk* jenis RDBMS (*Relational Database Management System*). Sehingga istilah seperti tabel, baris, dan kolom tetap digunakan. Pada MySQL sebuah database mengandung beberapa tabel, tabel terdiri dari beberapa baris dan kolom.