BAB II
TINJAUAN PUSTAKA

2.1. Teori Umum

2.1.1. Pengertian Komputer

Menurut Wahyudin dan Munir (2018:1), "Komputer adalah suatu alat elektronik yang mampu melakukan beberapa tugas, yaitu menerima *input*, memproses *input* sesuai dengan instruksi yang diberikan, menyimpan perintah-perintah dan hasil pengolahannya, serta menyediakan *output* dalam bentuk informasi."

Menurut William dalam Irma (2016:2), "Komputer adalah suatu pemrosesan data yang dapat melakukan perhitugan yang benar dan cepat, termasuk perhitungan aritmatika yang besar atau operasi logikan tanpa campur tangan dari manusia dalam pengoperasiannya selama pemrosesan."

Dari definisi diatas maka Komputer dapat diartikan sebagai suatu alat elektronik yang dapat melakukan beberapa tugas secara cepat dan benar dengan cara menyimpan *input*, melakukan pemrosesan, dan menghasilkan *output* berdasarkan intruksi program yang tersimpan.

2.1.2. Pengertian Informasi

Menurut Rusmawan (dalam Gaol, 2019:31), "Informasi adalah data yang telah diproses atau diolah ke dalam bentuk yang sangat berarti untuk penerimnya dan merupakan nilai yang sesungguhnya atau dipahami dalam tindakan atau keputusan yang sekarang atau nantinya."

Selain itu Menurut Hutahaean (2015:9), "Informasi adalah data yang diolah menjadi bentuk yang lebih berarti bagi penerimanya."

Dari definisi diatas maka Informasi adalah data yang telah diproses menjadi bentuk yang lebih berarti, dan dapat dipahami dalam tindakan atau keputusan bagi penerimanya.

2.1.3. Pengertian Data

Menurut Rusmawan (2019:34), “Data adalah catatan atas kumpulan fakta. Data merupakan bentuk jamak dari datum, berasal dari bahasa latin yang berarti sesuatu yang diberikan.”

Menurut Hutahean (2015:8), “Data adalah bahan mentah bagi informasi, dirumuskan sebagai kelompok lambing-lambang tidak acak menunjukkan jumlah-jumlah, tindakan-tindakan, hal-hal dan sebagainya.”

Dari definisi diatas maka Data merupakan bahan mentah yang belum diolah dan harus melalui proses sehingga dapat menghasilkan informasi yang dapat digunakan oleh pengguna.

2.1.4. Pengertian *Database*

Menurut Pamungkas (2017:2). “*Database* atau basis data merupakan suatu kumpulan data terhubung yang disimpan secara bersama-sama pada suatu media, yang diorganisasikan berdasarkan sebuah skema atau struktur tertentu, dan dengan *software* untuk melakukan manipulasi untuk kegunaan tertentu. Database dapat diartikan juga sebagai sekumpulan data yang disusun dalam beberapa tabel yang saling memiliki relasi maupun berdiri sendiri.”

Menurut Sukamto (2018:43), “Basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat.”

Dari definisi diatas maka *Database* adalah sekumpulan data yang disusun dalam beberapa tabel untuk menyimpan data agar dapat diakses dengan mudah dan cepat.

2.2. Teori Khusus

2.2.1. Pengenalan *Unified Modelling Language*

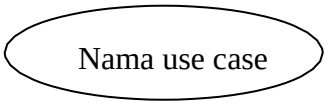
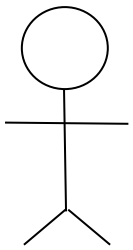
Menurut Sukamto (2018:13), *Unified Modeling Language* (UML*)* adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan

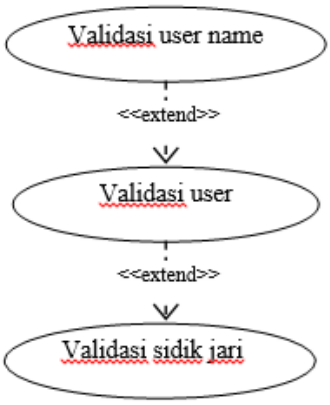
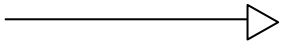
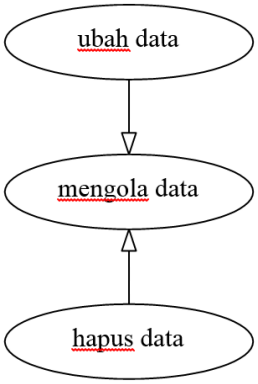
diagram dan teks-teks pendukung. UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek.

2.2.2. *Usecase Diagram*

Sukamto (2018:155), *use case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Berikut adalah simbol-simbol yang ada pada diagram *use case* :

Tabel 2.1. Simbol-simbol diagram *use case*

No.	Simbol	Deskripsi
1.	*Use case* 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal frase nama *use case*.
2.	Aktor/*actor* 	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
3.	Assosiasi/*association* 	Komunikasi antara aktor dan *use case* yang berpartisilpasi pada *use case* atau *use case* memiliki interaksi dengan aktor.

4.	Exstensi/*extend* <<extend>>	Relasi *use case* tambahan kesebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal  Arah panah mengarah pada *use case* yang ditambahkan, biasanya *use case* yang menjadi *extend*-nya merupakan jenis yang sama dengan *use case* yang menjadi induknya.
5.	Generalisasi/*generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua *buah use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya :  arah panah mengarah pada *use case*

		yang menjadi generalisasinya (umum)

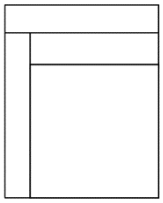
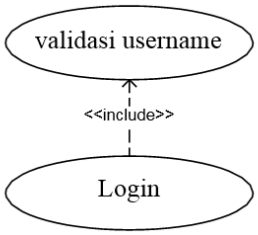
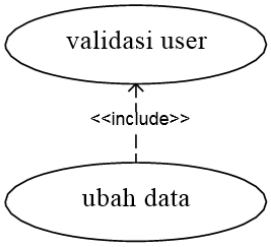
Sumber : Sukamto (2018:156-158)

2.2.3. Activity Diagram

Sukamto (2018:161), diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Berikut adalah simbol-simbol yang ada pada diagram aktivitas :

Tabel 2.2. Simbol-simbol *activity diagram*

No.	Simbol	Deskripsi
1.	Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
2.	Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
3.	Percabangan/*decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
4.	Penggabungan/join	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
5.	Status akhir 	Status akhir yang dilakukan oleh sistem, sebuah diagram aktivitas memiliki sebuah status akhir.

6.	Swimlane 	Memisahkan organisasi bisnis yang bertanggunga jawab terhadap aktivitas yang terjadi.
7.	Menggunakan / *include* / *uses* <<include>> «uses»	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* : - *Include* berarti *use case* yang ditambahkan akan selalu di panggil saat *use case* tambahan dijalankan, missal pada kasus berikut :  - *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang di tambahkan telah dijalankan sebelum *use case* tambahan di jalankan, misal pada kasus berikut :  Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

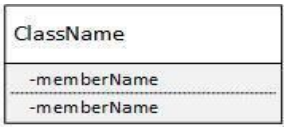
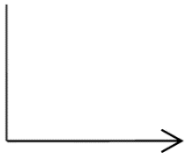
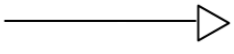
Sumber : Sukamto (2018:162-163)

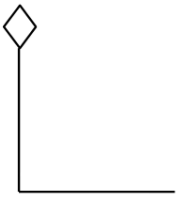
2.2.4. *Class Diagram*

Sukamto (2018:141), diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan *method* atau operasi. Berikut penjelasan atribut dan *method* :

1. Atribut merupakan variable-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau *method* adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Tabel 2.3 Simbol-simbol *class diagram*

No.	Simbol	Deskripsi
1.	Kelas 	Kelas pada struktur sistem
2.	Antarmuka/*interface* 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
3.	Asosiasi/*association* 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
4.	Asosiasi berarah/ *directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
5.	Generalisasi 	Relasi antar kelas dengan makna generalisasi – spesialisasi (umum - khusus)

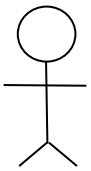
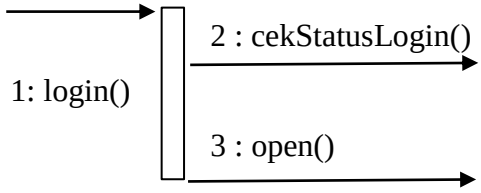
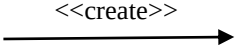
6.	Kebergantungan/*dependensi* 	Relasi antar kelas dengan makna kebergantungan antar kelas
7.	Agregasi/*aggregation* 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

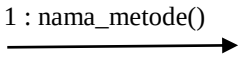
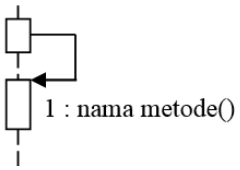
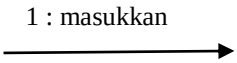
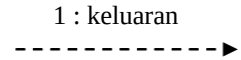
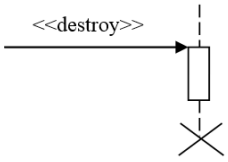
Sumber : Sukamto (2018:146-147)

2.2.5. *Sequence Diagram*

Sukamto (2018:165), "diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dengan *massage* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*. Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup dalam diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak. Berikut adalah simbol-simbol yang ada pada diagram sekuen:"

Tabel 2.4. Simbol-simbol *sequence diagram*

No.	Simbol	Deskripsi
1.	Aktor  Atau Nama aktor Tanpa waktu aktif	Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya dinyatakan dalam menggunakan kata benda diawal frase nama aktor.
2.	Garis hidup/*lifeline*	Menyatakan kehidupan suatu objek
3.	Objek Nama objek : nama kelas	Menyatakan objek yang berinteraksi pesan
4.	Waktu aktif 	Menyatakan objek dalam keadaan aktif dan berinteraksi, semuanya yang terhubungdengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  Maka cekStatusLogin() dan open() dilakukan didalam metode login(). Aktor tidak memiliki waktu aktif
5.	Pesan tipe *create* 	Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat

6.	Pesan tipe *call* 	Menyatakan suatu objek memanggil operasi/metode yang ada pada objek lain atau dirinya sendiri,  Arah panah mengarah pada objek yang memiliki operasi/metode, karena ini memanggil operasi/metode maka operasi/metode yang dipanggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi.
7.	Pesan tipe *send* 	Menyatakan bahwa suatu objek mengirimkan data/masukkan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.
8.	Pesan tipe *return* 	Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.
9.	Pesan tipe *destroy* 	Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaliknya jika ada *create* maka ada *destroy*

Sumber: : Sukamto (2018:165-167)

2.3. Teori Judul

2.3.1. Pengertian Aplikasi

Menurut Santoso (2015:9) "Aplikasi adalah suatu kelompok *file (form, class, report)* yang bertujuan untuk melakukan aktivitas tertentu yang saling terkait.*"*

Menurut Kadir (2017:3) "Aplikasi lebih sering disebut untuk menyatakan perangkat lunak. Di kalangan profesional teknologi informasi, istilah program biasa digunakan untuk menyatakan hasil karya mereka yang berupa instruksi-instruksi untuk mengendalikan komputer. Di sisi pemakai, hal seperti itu biasa disebut sebagai aplikasi."

Dari definisi diatas maka Aplikasi adalah suatu kelompok *file* dalam komputer yang bertujuan untuk melakukan aktivitas tertentu dengan sarana komputer.

2.3.2. Pengertian Pemantauan *(Monitoring)*

Menurut Nasir (2013:124) "Pemantauan (*Monitoring)* adalah pengumpulan informasi secara teratur yang akan membantu menjaga agar pekerjaan tetap pada jalurnya dan dapat memperingatkan anda ketika terjadi suatu salah."

Manurut Mulyono dan Yumari (2017:170) "Pemantauan (*Monitoring)* adalah ketentuan-ketentuan yang disepakati dan diberlakukan, selanjutnya kegiatanya harus terjaga, dalam pelaksanaannya objektivitas sangat diperhatikan dan orientasi utamanya adalah pada tujuan program itu sendiri."

Dari definisi diatas maka Pemantauan (*Monitoring)* adalah pengumpulan informasi secara teratur yang kegiatanya harus terjaga agar pekerjaan tetap pada jalurnya.

2.3.3. Pengertian Kinerja

Menurut Colquitt, LePine, dan Wesson dalam Wibowo (2016:2) “Kinerja adalah nilai serangkaian perilaku pekerja yang memberikan kontribusi, baik secara positif maupun negatif, pada penyelesaian tujuan organisasi.”

Menurut Donelly, Gibson and Ivancevich dalam Wibowo (2016:2), “Kinerja adalah hasil pekerjaan yang berkaitan dengan tujuan organisasi seperti kualitas, efisiensi dan kriteria lain dari efektivitas.”

Dari definisi diatas maka Kinerja adalah perilaku pekerja yang memberikan kotribusi untuk mencapai tujuan organisasi.

2.3.4. *Rational Unified Process*

RUP merupakan sebuah pengembangan rekayasa perangkat lunak dengan pendekatan yang disiplin dalam melakukan tiap tugas dan tanggung jawabnya pada sebuah organisasi.

Kelebihan metode Rational Unified Process menurut Sukamto (2018:126) adalah sebagai berikut:

1. RUP mngakomodasikan perubahan kebutuhan perangkat lunak. Kebutuhan untuk mengubah dan menambah fitur karena perubahan teknologi atau keinginan pelanggan merupakan salah satu kendala yang sering dialami pengembangan perangkat lunak yang brimbas pada terlambatnya waktu penyelesaian perangkat lunak.
2. Integrasi bukanlah sebuah proses besar dan cepat di akhir proyek. Pendekatan secara iteratif (pengulangan) dapat memecah proyek menjadi bagian iterasi kecil yang diakhiri dengan integrasi kecil yang nantinya digabungkan menjadi integrasi besar.
3. Resiko biasanya ditemukan atau dialamatkan selama pada proses integrasi awal. Pendekatan integrasi pada RUP mengurangi resiko pada iterasi awal dimana saat semua komponen diuji.
4. Manajemen berarti membuat perubahan taktik pada produk. Taktik produk misalnya pengembangan dengan waktu singkat akan menghasilkan produk dengan fungsi yang terbatas akan dapat cepat digunakan oleh user.

5. Mendukung fasilitas penggunaan kembali. Peninjauan kembali pada iterasi awal dapat membuat arsitek perangkat lunak untuk menandai peluang penggunaan kembali *(reuse)* dan kemudian mengembangkan kode umum yang lebih baik atau mapan pada iterasi berikutnya.
6. Kecacatan dapat ditemukan dan diperbaiki pada beberapa iterasi menghasilkan arsitetur yang baik dan aplikasi berkualitas tinggi.
7. Lebih baik menggunakan "anggota proyek" dibandingkan susunan secara seri pada tim proyek. Seorang analis bekerja untuk menganalisis kebutuhan sistem lalu memberikan hasil analisis ke desainer untuk melakukan desain sistem, kemudiann desainer memberikan desain ke programmer dan programmer mengirimkan aplikasi ke pelanggan.

2.4. Teori Program

2.4.1. Pengertian PHP

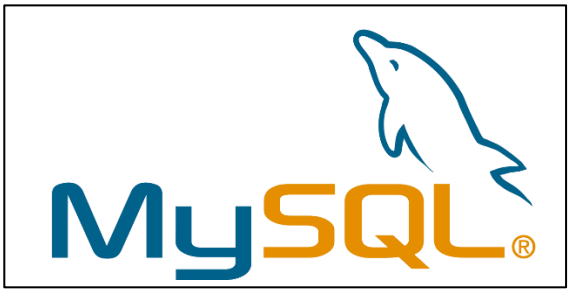
Menurut Sidik (2017:5), PHP: *HyperText Preprocesssor* merupakan bahasa utama *script serverside* yang disisipkan pada HTML yang dijalankan di *server*, dan juga bisa digunakan untuk membuat aplikasi desktop.

Menurut Raharjo (2016:38), PHP adalah salah satu bahasa pemrograman skrip yang dirancang untuk membangun aplikasi. Ketika dipanggil dari *web browser*, program yang ditulis dengan PHP akan di-*parsing* di dalam *webserver* oleh *interpreter* PHP dan diterjemahkan ke dalam dokumen HTML, yang selanjutnya akan kembali ke *web browser*.

Sumber : *https://en.wikipedia.org/wiki/File:PHP-logo.svg*

Gambar 2.2 Logo PHP

2.4.2. Pengertian *MySql*



Sumber : *https://en.wikipedia.org/wiki/MySQL*

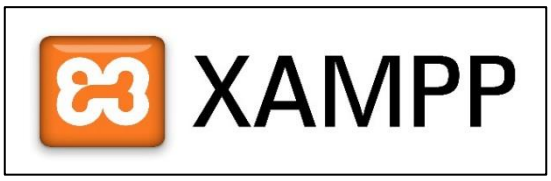
Gambar 2.3 Logo MySQL

Menurut Murya (2014:46), MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (*Database Management System*) atau DBMS yang *mutiheard*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. Keistimewaan yang dimiliki MySQL, antara lain:

1. MySQL dapat berjalan stabil pada berbagai sistem operasi.

2. MySQL didistribusikan sebagai perangkat lunaksumber terbuka, dibawah lisensi GPL sehingga dapat digunakan secara gratis.
3. MySQL dapat digunakan beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah dan konflik.
4. MySQL memiliki kecepatan yang menakjubkan dalam menagani *query*.
5. MySQL memiliki ragam tipe data yang sangat kaya, seperti signed/unsigned integer, float, double char, text, date, timestamp, dan lain-lain.
6. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah *query*.
7. MySQL memiliki beberapa lapisan keamanan seperti subnetmassk, nama host, dan izin akses user dengan sistem perizinan yang mendetail serta sandi terenkripsi.
8. MySQL mampu menangaini basis data dalam skla besar, dengan jumlah rekaman lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris.
9. MySQL dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, Unix soket, atau Named Pipes.
10. MySQL dapat mendeteksi pesan kesalahan pada klien.
11. MySQL memiliki antarmuka terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*)
12. MySQL dilengkapi dengan berbagai peralatan yang dapat digunakan untuk administrasi basis data.
13. MySQL memiliki struktur tabel yang lebih fleksibel dalm menangani ALTER TABLE.

2.4.3. XAMPP



Sumber : *https://commons.wikimedia.org/wiki/File:Xampp_logo.svg*

Gambar 2.4 Logo XAMPP

Menurut Hidayatullah dan Jauhari (2014:127), *XAMPP* adalah fasilitas untuk banyak sistem operasi seperti Windows, Linux, Mac, dan Solaris yang memungkinkan sebuah *web* dinamis bisa diakses secara local menggunakan *web server* local. Kata *XAMPP* sendiri terdiri dari :

1. X yang berarti *Cross Platform* karena *XAMPP* bisa dijalankan di Windows, Linux, Mac, dan Solaris.
2. A yang berarti Apache sebagai *web-server*-nya.
3. M yang berarti *MySQL* sebagai *Database Management System (DBMS)*
4. PP yang berarti *PHP* dan Perl sebagai bahasa yang didukungnya.

2.4.4. *HyperText Markup Language* (HTML)

Menurut Winarno (2014:1) *Hyper Text Markup Language (HTML)* adalah sebuah bahasa untuk menampilkan konten di *web*. *HTML* mirip dengan teks biasa, hanya dalam dokumen ini tulisan mengandung beberapa instruksi yang ditandai dengan kode tertentu yang dikenal dengan tag tertentu.

Elemen *HTML* dimulai dengan *tag* awal, yang diikuti dengan isi elemen dan *tag* akhir. *Tag* terakhir termasuk simbol/diikuti oleh tipe elemen, misalnya </HEAD>. Sebuah elemen *HTML* dapat bersarang di dalam elemen lainnya.