BAB II
TINJAUAN PUSTAKA

2.1 Teori Umum

2.1.1 Pengertian Komputer

Paramytha (2016:4), "Komputer berasal dari bahasa latin yaitu *Computer* yang berarti menghitung (*to compute* atau *to reckon).* Kata komputer itu sendiri pada awalnya dipergunakan untuk menggambarkan pekerjaan orang yang melakukan perhitungan aritmatika dengan atau tanpa alaat bantu".

Abdul Kadir (2017:2), "Komputer adalah peralatan elektronik yang bermanfaat untuk melaksanakan berbagai pekerjaan yang dilakukan oleh manusia".

Dari kedua pendapat diatas penulis dapat simpulkan pengertian komputer adalah peralatan elektronik yang dipergunakan untuk menggambarkan pekerjaan yang dilakukan oleh manusia.

2.1.2 Pengertian Internet (*Interconnected Network*)

Pibriana dan Ricoida (2017:105), "Internet adalah penghubung antara organisasi dan pelanggannya, sehingga tercipta sebuah organisasi baru secara visual".

Zabar dan Novianto (2015:69), "Internet adalah suatu jaringan komputer yang saling terhubung untuk keperluan komunikasi dan informasi".

Dari beberapa definisi diatas penulis menyimpulkan bahwa internet adalah suatu jaringan komputer yang saling terhubung sehingga tercipta sebuah organisasi baru secara visual.

2.1.3 Pengertian Perangkat Lunak

Kadir (2017:2), "Perangkat Lunak adalah instruksi-instruksi yang ditujukan kepada komputer agar dapat melaksanakan tugas sesuai kehendak

pemakai. Sistem operasi seperti *Windows, Mac OS,* dan *Linux,* dan aplikasi seperti *Microsoft Word* dan *Microsoft Excel* adalah contoh perangkat lunak."

Sukamto dan Shalahuddin (2018:2), "Perangkat Lunak (*Software*) adalah program komputer yang terasosiasi dengan dokumentasi perangkat lunak seperti dokumentasi kebutuhan, model desain dan cara penggunaan (*user manual*)."

Jadi dapat penulis simpulkan pengertian perangkat Lunak adalah program komputer yang terasosiasi denga dokumentasi perangkat lunak agar dapat melaksanakan tugas sesuai kehendak pemakai.

2.1.4 Pengertian Basis Data

Sukamto dan Shalahudin (2018:43),"Basis data adalah sistem komputerisasi yang tujuan utamanya adalah memelihara data yang sudah ada yang diolah atau informasi dan membuat informasi tersedia saat dibutuhkan."

Fathansyah (2018:2),"Basis data adalah himpunan kelompok data (arsip) yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah."

Jadi dapat penulis simpulkan pengertian basis data adalah sistem komputerisasi yang saling berhubungan yang diorganisasi sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah.

2.1.5 Metode Pengembangan Aplikasi

Penelitian ini menggunakan metode pengembangan perangkat lunak dengan RUP (*Rational Unified Process*). Menurut Sukamto dan Shalahuddin (2018:125), "RUP (*Rational Unified Process*) adalah pendekatan pengembangan perangkat lunak yang dilakukan berulang-ulang (*iterative*), fokus pada arsitektur (*architecture-centric*), lebih diarahkan berdasarkan penggunaan kasus (*use case driven*)". Adapun tahap-tahap (*fase*) dalam metode pengembangan RUP menurut Sukamto dan Shalahuddin (2018:128-131) adalah sebagai berikut:

1. *Inception* (permulaan)

Tahap ini lebih pada memodelkan proses bisnis yang dibutuhkan (*bussiness modeling*) dan mendefinisikan kebutuhan akan sistem yang akan dibuat (*requirements*).

2. *Elaboration* (perluasan/perencanaan)

Tahap ini lebih difokuskan pada perencanaan arsitektur sistem. Tahap ini juga dapat mendeteksi apakah arsitektur sistem yang diinginkan dapat dibuat atau tidak. Mendeteksi resiko yang mungkin terjadi dari arsitektur yang dibuat. Tahap ini lebih pada analisis dan desain sistem serta implementasi sistem yang fokus pada purwarupa sistem (*prototype*).

3. *Construction* (kontruksi)

Tahap ini fokus pada pengembangan komponen dan fitur-fitur sistem. Tahap ini lebih pada implementasi dan pengujian sistem yang fokus pada implementasi perangkat lunak pada kode program. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/tonggak kemampuan operasional awal.

4. *Transition* (transisi)

Tahap ini lebih pada deployment atau instalasi sistem agar dapat dimengerti oleh user. Tahap ini menghasilkan produk perangkat lunak dimana menjadi syarat dari *Initial Operational Capability Milestone* atau batas/tongggak kemampuan operasional awal. Aktifitas pada tahap ini termasuk pada pelatihan user, pemeliharaan dan pengujian sistem apakah sudah memenuhi harapan user.

2.2 Teori Judul

2.2.1 Pengertian Sistem \

Sukamto (2018:1), “Sistem merupakan kumpulan elemen-elemen yang saling terkait dan bekerja sama untuk memroses masukan (*input*) yang ditujukan kepada sistem tersebut dan mengolah masukan tersebut sampai menghasilkan keluaran (*output*) yang diinginkan.”

Fathansyah (2018:11), “Sistem adalah sebuah tatanan (keterpaduan) yang terdiri atas sejumlah komponen fungsional (dengan satuan fungsi dan tugas khusus) yang saling berhubungan dan secara bersama-sama bertujuan untuk memenuhi suatu proses tertentu.”

Jadi dapat penulis simpulkan pengertian sistem adalah sebuah tatanan (keterpaduan) yang terdiri atas sejumlah komponen fungsional yang saling berkaitan dan saling terhubung yang mengolah masukan (input) menjadi keluaran (output) yang diinginkan .

2.2.2 Pengertian Informasi

Menurut Kristanto (2018:7), “Informasi merupakan kumpulan data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerima”.

2.2.3 Pengertian Senat Akademik

Peraturan Pemerintah No. 66/2010 tentang Pengelolaan dan Penyelenggaraan Pendidikan, serta Permendikbud No.54/2011 tentang Statuta Polsri “Senat Akademik adalah badan normatif tertinggi di bidang akademik dan merupakan salah satu dari 4 organ yang ada di Statuta Politeknik Negeri Sriwijaya (Polsri)”.

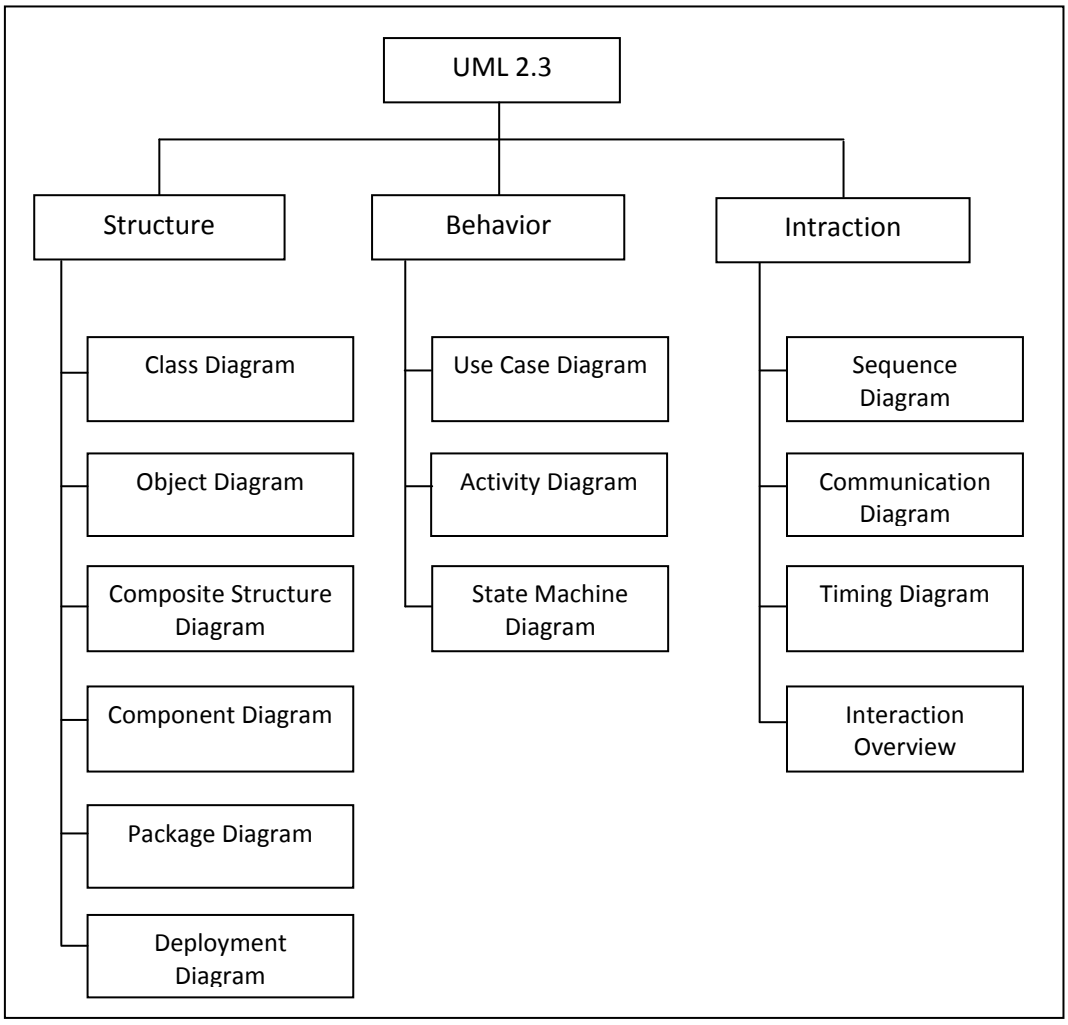
2.2.4 Pengertian Sistem Informasi Senat Akademik pada Politeknik Negeri Sriwijaya

Sistem Informasi Senat Akademik pada Politeknik Negeri Sriwijaya adalah Sistem Informasi yang dibuat untuk membantu Pusat/UPT/anggota senat dan bagian-bagian lainnya dalam mencari kegiatan dan dokumen senat.

2.3 Teori Khusus

2.3.1 Pengertian *UML (Unified Modeling Language)*

Menurut Sukamto dan Shalahuddin (2018:140), "Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori". Pembagian kategori dan macam-macam diagram Menurut Sukamto dan Shalahuddin tersebut dapat dilihat pada gambar dibawah:



Gambar 2.1 Macam-macam Diagram UML

Penjelasan singkat dari pembagian kategori pada diagram UML menurut Sukamto dan Shalahuddin (2016:141) :

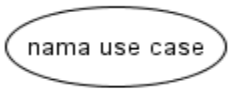
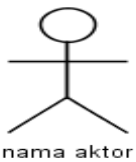
1) *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

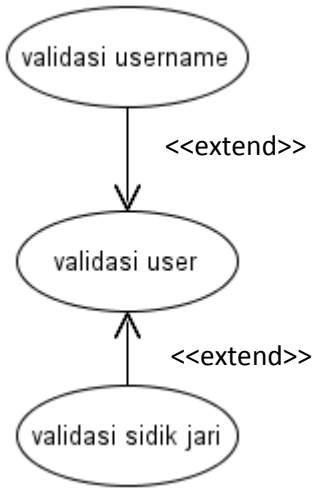
2) *Behavior diagrams* yaitu kumpulam diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahana yang terjadi pada sebuah sistem.
3) *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

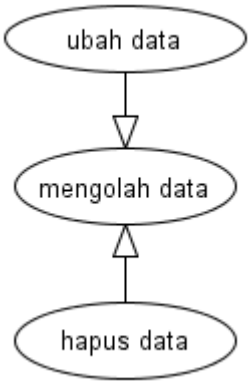
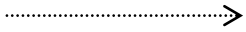
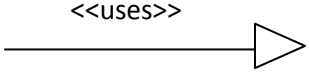
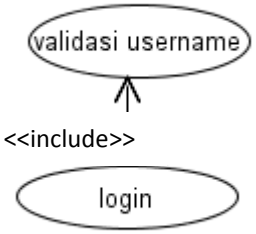
2.3.2 Pengertian *Use Case Diagram*

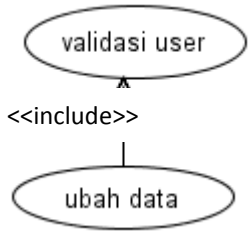
Sukamto dan Shalahuddin (2018:155), menjelaskan tentang *use case* diagram sebagai berikut : *"Use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem." Adapun simbol-simbol yang digunakan dalam *use case* adalah sebagai berikut:Berikut simbol-simbol pada Use Case Diagram :

Tabel 2.1 Simbol-simbol pada *Use case* Diagram

Simbol	**Deskripsi**
Use case 	fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal-awal frase nama *use case*
aktor / *actor* 	orang, proses, atau siatem 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
asosiasi / *association*	komunikasi antar aktor dan *use case* yang

———————	berpartisioasi pada *use case*.
ekstensi / *extend* <<extend>> ⇢	relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang di tambahkan 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, misalnya  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
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)
menggunakan / include / uses <<include>>  <<uses>> 	relasi 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, misalnya 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 dijalankan,

	misal pada kasus berikut:  kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan.

2.3.3 Pengertian *Activity Diagram*

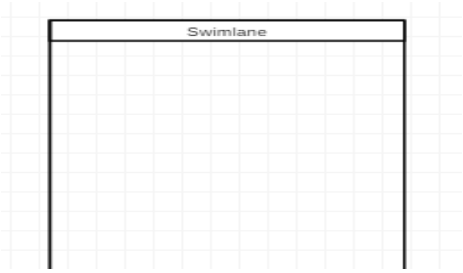
Sukamto dan Shalahuddin (2018:161), menjelaskan tentang *activity diagram* sebagai berikut :

A*ctivity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Adapun simbol-simbol yang digunakan dalam *activity diagram* adalah sebagai berikut:

Tabel 2.2 Simbol-simbol pada *Activity Diagram*

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan

	kata kerja
Percabangan / *decision* 	Asosiasi percabangan di mana jika ada pilihan aktivitas lebih dari satu
Penggabungan / *join* 	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir
Swimlane  atau  	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

2.3.4 Pengertian *Class Diagram*

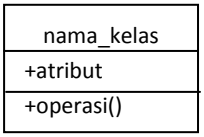
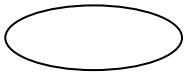
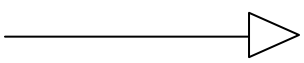
Sukamto dan Shalahuddin (2018:141), menjelaskan tentang *class diagram* sebagai berikut :

Class Diagram menggambarkan struktur sistem dari segi pendefinisian

kelas-kelas yang akan dibuat untuk membangun sistem. *Diagram Class* dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

Adapun simbol-simbol yang digunakan dalam *class diagram* adalah sebagai berikut:

Tabel 2.3 Simbol-simbol pada *Class Diagram*

Simbol	Deskripsi
kelas 	Kelas pada struktur sistem
antarmuka / interface  nama_interface	Sama dengan konsep interface dalam pemrograman berorientasi objek
asosiasi / association	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai *multiplicity*
asosiasi berarah / *directed assosiation*	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
generalisasi 	Relasi antarkelas dengan makna generalisasi – spesialisasi (umum khusus)
kebergantungan / *dependency*	Relasi antarkelas dengan

	makna kebergantungan antar kelas
agregasi / *aggregation* 	Relasi antarkelas dengan makna semua-bagian (*whole-part*)

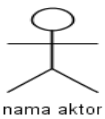
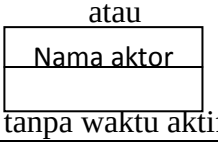
2.3.5 Pengertian *Sequence Diagram*

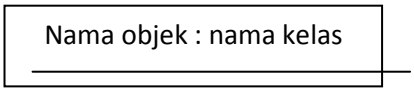
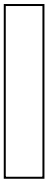
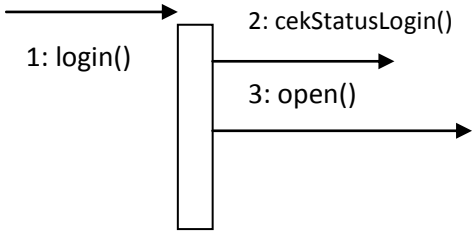
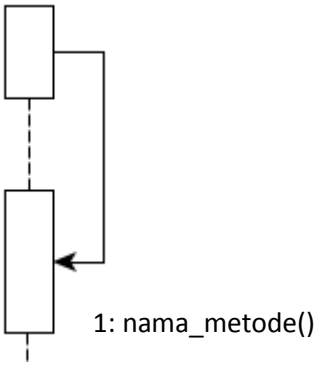
Sukamto dan Shalahuddin (2018:141), menjelaskan tentang *Sequence diagram* sebagai berikut :

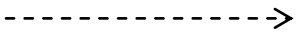
Diagram *sequence* menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah beserta metode-metode yang dimiliki kelas yang diinstansikan menjadi objek itu. Membuat diagram *sequence* juga dibutuhkan untuk melihat skenario yang ada pada use case.

Banyaknya diagram *sequence* yang harus digambarkan adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram *sequence* sehingga semakin banyak *use case* yang didefinisikan maka diagram *sequence* yang harus dibuat juga semakin banyak. Berikut simbol-simbol pada *Sequence Diagram* :

Tabel 2.4 Simbol-simbol pada *Sequence Diagram*

Simbol	Deskripsi
Actor  atau  tanpa waktu aktif	orang, proses, atau sistem lain yang berinteraksi dengan 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 actor

Garis hidup / *lifeline* 	menyatakan kehidupan suatu objek
Objek 	menyatakan objek yang berinteraksi pesan
Waktu aktif 	menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah sebuah tahapan yang dilakukan di dalamnya, misalnya  maka cekStatusLogin () dan open() dilakukan di dalam metode login() aktor tidak memiliki waktu aktif
Pesan tipe create <<create>> 	
Pesan tipe call <<create>> 	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
Pesan tipe send 1: masukan 	menyatakan bahwa suatu objek mengirimkan data/masukan/informasi ke objek lainnya, arah panah mengarah pada objek yang dikirimi]
Pesan tipe return 1: keluaran 	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

2.4 Teori Program

2.4.1 Pengertian *XAMPP*

Iqbal (2019:15), "Xampp merupakan sebuah software web server apache yang didalamnya sudah tersedia database server mysql dan support php programming."

Aryanto (2016:4), "*Xampp* merupakan sebuah aplikasi perangkat lunak pemrograman dan *database* yang di dalamnya terdapat berbagai macam aplikasi pemrograman seperti : *Apache, HTTP, MySQL, database,* bahasa pemrograman *PHP* dan *Perl*."

Dari beberapa definisi diatas penulis menyimpulkan bahwa *XAMPP* adalah sebuah aplikasi perangkat lunak pemrograman dan *database* yang di dalamnya terdapat berbagai macam aplikasi pemrograman yang terdiri dari Apache, *MySQL, PhpMyAdmin, Perl, Filezilla* dan lain-lain.

2.4.2 Pengertian *MySQL*

Sukamto dan Shalahuddin (2018:46), "SQL (*Structured Query Language*) adalah bahasa yang digunakan untuk mengelola data pada RDBMS. SQL awalnya dikembangkan berdasarkan teori aljabar relasional dan kalkulus."

Yosef Murya (2014:46), "MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (database management system) atau DBMS yang *multitheard, multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia."

Kesimpulannya, MySQL adalah bahasa yang digunakan untuk mengelola data SQL (database management system) atau DBMS yang *multitheard, multi-user.*

2.4.3 Pengertian *PHP*

Solichin dan Brotosaputro (2016:23), "PHP merupakan bahasa pemrograman berbasis *web* yang dibuat secara khusus untuk membangun aplikasi berbasis *web*."

Madcoms (2016:2), "*PHP* (*Hypertext preprocessor)* adalah bahasa script yang dapat ditanamkan atau disisipkan ke dalam HTML."

Berdasarkan kedua pendapat diatas, maka penulis dapat simpulkan, PHP adalah bahasa pemograman berbasis *server-side* yang bisa kita gunakan untuk membuat aplikasi web yang disisipkan pada HTML.

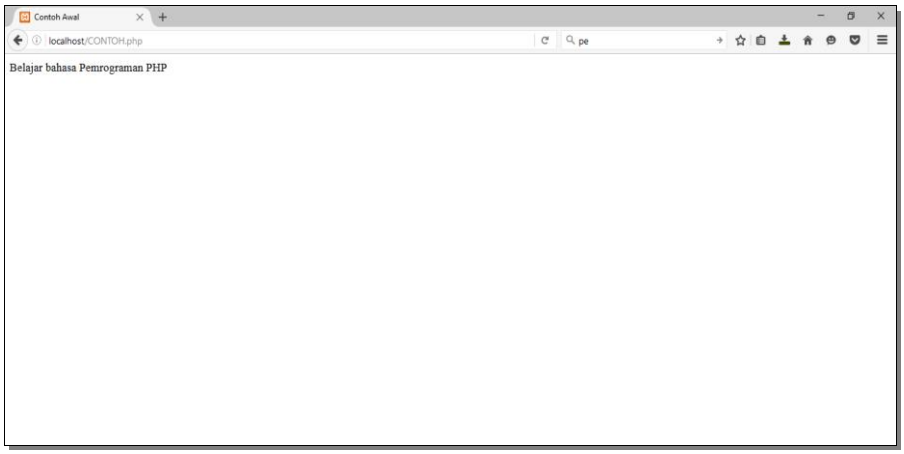
Gambar 2.2 Tampilan Logo *PHP*

2.4.3.1 Sintaks Dasar *PHP*

Kode (Script) *PHP* yang sering disebut dengan istilah embedded script yaitu script PHP yang disisipkan di antara sricpt HTML. Jadi dapat dikatakan script PHP hanya ditulis atau disisipkan ketika dibutuhkan saja, seperti

menampilkan data dari database meng-upload file, delete data, edit data dan lain sebagainya. Contoh script :

```
<HTML>
    <HEAD>
        <TITLE>Contoh Awal</TITLE>
    </HEAD>
<BODY>
    <?php
        echo “Belajar bahasa Pemrograman PHP”;
    ?>
    </BODY>
</HTML>
```



Gambar 2.3 Contoh *Script PHP*

2.4.3.2 Tipe Data *PHP*

Tipe data merupakan jenis dari suatu data yang akan diproses oleh bahasa pemrograman. Yosef Murya (2014:26), menjelaskan beberapa tipe data dalam PHP, sebagai berikut :

1. **Integer** merupakan tipe data yang berguna untuk menyimpan bilangan bulat. Range bilangan integer adalah antara -2.147.4833.647 sampai dengan 2.147.483.647

2. **Double Floating** adalah tipe data yang berguna untuk menyimpan bilangan desimal. Range bilangan floating point antara 1e308 sampai dengan 1e308.
3. **Boolean** adalah tipe data yang paling sederhana, hanya berupa **TRUE** dan **FALSE**.
4. **String** adalah tipe data yang terdiri dari kata, bias berupa kata tunggal maupun kalimat. Penulisan string harus diapit dengan tanda petik, baik berupa petik tunggal (‘…’) maupun petik ganda (“...”).
5. **Objek** adalah tipe data dibuat dengan tujuan agar para programmer terbiasa dengan OOP. Tipe data ini bias berupa bilangan.
6. **Array** merupakan **Tipe Compound Primitif,** terdapat pada bahasa pemrograman lain.
7. **Null** adalah tipe data yang tidak memuat apapun. Setiap variable yang diset menjadi tipe data Null, ini akan menjadikan variabel tersebut kosong.
8. **Resources** tipe data spesial yang satu ini dikhususkan untuk menyimpan *resources*, sumber atau alamat.

2.4.4 Pengertian *JavaScript*

Abdulloh (2018:193), “*Javascript* merupakan Bahasa pemrograman web yang pemrosesannya dilakukan di sisi *client.* Karena berjalan di sisi *client. Javascript* dapat dijalankan hanya dengan menggunakan browser.”

2.4.4.1 Dasar JavaScript

Cara menggunakan JavaScript adalah dengan dimasukkan di antara kode HTML menggunakan tag <script> dan </script>. Javascript bisa diletakkan di tag <body> ataupun tag <head> dari kode HTML. Untuk memasukkan javascript wanda harus menggunakan tag <script>, tag <script> dan </script> menentukan dimana javascript harus dimulai dan diakhiri.

Baris diantara tag <script> dan </script> ini berisi data Javascript contohnya seperti berikut :

```
<!DOCTYPE html>
<html>
<head>
   <title>Hello World Javascript</title>
</head>
<body>
   <script>
      console.log("Saya belajar Javascript");
      document.write("Hello World!");
   </script>
</body>
</html>
```