**BAB III**

**LANDASAN TEORI**

1. **Konsep Dasar Sistem**

Dalam berbagai sudut pandang yang dikemukakan oleh beberapa pakar, sistem memiliki berbagai perbedaan namun mempunyai tujuan yang sama dimana sistem berkembang sesuai konteks dan sistem didefinisikan sebagai sekumpulan prosedur yang saling berkaitan dan saling berhubungan untuk melakukan tugas bersama-sama. (Sutabri, 2012).

1. **Komponen Sistem**
2. Perangkat lunak *(Software)*

Software mencakup semua perangkat lunak yang dibangun dengan bahasa pemprograman tertentu, pustaka, untuk kemudian menjadi sistem operasi, aplikasi, dan *driver.* dan keseluruhannya bekerjasama agar sistem berjalan dengan baik.

1. Perangkat Keras *(Hardware)*

*Hardware* mencakup semua perangkat keras yang disatukan dan dapat digunakan oleh *software.*

1. Perangkat Otak *(Brainware)*

Brainware mencakup kemampuan otak manusia, yang mencakup ide, pemikiran, analisis di dalam menciptakan dan menggabungkan *hardware* serta *software* melalui sejumlah prosedur.

1. **Karakteristik Sistem**

Dalam pengembangan sebuah sistem perlu diketahui unsur - unsur yang membedakan antara sistem yang satu dengan sistem yang lain. Berikut karakter sistem yang membedakan suatu sistem :

1. **Mempunyai Komponen *(Components)***

Komponen sistem adalah segala sesuatu yang menjadi bagian penyusun sistem. Komponen sistem dapat berupa benda nyata ataupun abstrak. Komponen sistem disebut sebagai sub sistem, dapat berupa orang, benda, hal atau kejadian yang terlibat didalam sistem.

1. **Mempunyai Batas *(Boundry)***

Batas sistem diperlukan untuk membedakan satu sistem dengan sistem yang lain. Tanpa adanya batas sistem, maka sangat sulit untuk menjelaskan suatu sistem. Batas sistem akan memberikan batasan scope tinjauan terhadap sistem.

1. **Mempunyai Lingkungan (Environments)**

Lingkungan sistem adalah segala sesuatu yang berada diluar sistem. Lingkungan sistem dapat menguntungkan ataupun merugikan. Umumnya, lingkungan yang menguntungkan akan selalu dipertahankan untuk menjaga keberlangsungan sistem. Sedangkan lingkungan sistem yang merugikan akan diupayakan agar mempunyai pengaruh seminimal mungkin, bahkan jika mungkin ditiadakan.

1. **Mempunyai Penghubung atau Antarmuka (Interface)**

Penghubung atau antar muka merupakan komponen sistem, yaitu segala sesuatu yang bertugas menjebatani hubungan antar komponen dalam sistem. Penghubung/antar muka merupakan sarana yang memungkinkan setiap komponen saling berinteraksi dan berkomunikasi dalam rangka menjalankan fungsi masing-masing komponen. Dalam dunia komputer, penghubung/antar muka dapat berupa berbagai macam tampilan dialog layar monitor yang memungkinkan seseorang dapat dengan mudah mengoperasikan sistem aplikasi komputer yang digunakan.

1. **Mempunyai Masukan (Input)**

Masukan merupakan komponen sistem, yaitu segala sesuatu yang perlu dimasukan kedalam sistem sebagai bahan yang akan diolah lebih lanjut untuk menghasilkan keluaran yang akan diolah lebih lanjut untuk menghasilkan keluaran yang berguna. Dalam Sistem Informasi Manajemen, masukan disebut sebagai data.

1. **Mempunyai Pengolahan (Proses)**

Pengolahan merupakan komponen sistem yang mempunyai peran utama mengolah masukan agar menghasilkan keluaran yang berguna bagi para pemakainya. Dalam Sistem Informasi Manajemen, pengolahan adalah berupa program aplikasi komputer yang dikembangkan untuk keperluan khusus. Program aplikasi tersebut mampu menerima masukan, mengolah masukan, dan menampilkan hasil olahan sesuai dengan kebutuhan para pemakai.

1. **Mempunyai Keluaran (Output)**

Keluaran merupakan komponen sistem yang berupa berbagai macam bentuk keluaran yang dihasilkan oleh komponen pengolahan. Dalam Sistem Informasi Manajemen, keluaran adalah informasi yang dihasilkan oleh program aplikasi yang akan digunakan oleh para pemakai sebagai bahan pengambilan keputusan.

1. **Mempunyai Sasaran (*Objectives*) dan tujun (*Goal*)**

Setiap komponen dalam sistem perlu dijaga agar saling bekerja sama dengan harapan agar mampu mencapai sasaran dan tujuan sistem. Sasaran berbeda dengan tujuan. Sasaran sistem adalah apa yang ingin dicapai oleh sistem untuk jangka waktu yang relatif pendek.

Sedangkan tujuan merupakan kondisi atau hasil akhir yang ingin dicapai oleh sistem untuk jangka waktu yang panjang. Dalam hal ini, sasaran merupakan hasil pada setiap tahapan yang mendukung upaya pencapaian tujuan.

1. **Pengertian Aplikasi**

Menurut Jogiyanto (2009) aplikasi merupakan program yang berisikan perintah-perintah untuk melakukan pengolahan data. Jogiyanto juga menjelaskan bahwa pengertian secara umum adalah proses dari cara manual yang ditranformasikan ke computer dengan membuat sistem atau program agar data diolah lebih berdaya guna secara optimal.

1. **Pengertian Web**

Website adalah keseluruhan halaman-halaman web yang terdapat dalam sebuah domain yang mengandung informasi. Sebuah website biasanya dibangun atas banyak halaman web yang saling berhubungan. Jadi dapat dikatakan bahwa, pengertian website adalah kumpulan halaman-halaman. yang digunakan untuk menampilkani informasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing-masing dihubungkan dengan jaringan-jaringan halaman. Hubungan antara satu halaman website dengan halaman website lainnya disebut dengan hyperlink, sedangkan teks yang dijadikan media penghubung disebut hypertext (Rahmat Hidayat, 2010).

1. **Pengertian Web Aplikasi**

Pengertian web aplikasi menurut para ahli :

Menurut Remick (2011), aplikasi web adalah sebuah aplikasi yang menggunakan teknologi browser untuk menjalankan aplikasi dan akses melalui jaringan komputer.

Menurut Rouse (2011), aplikasi web adalah sebuah program yang di simpan di server dan dikirim melalui internet akses melalui antarmuka browser.

Dari pengertian di atas dapat disimpulkan aplikasi web merupakan aplikasi yang diakses menggunakan web browser melalui jaringan internet atau intranet.

Web aplikasi adalah sebuah aplikasi yang penggunaan dalam suatu komputer, yang disusun sedemikian rupa sehingga komputer dapat memproses input dan output untuk memudahkan tugas khusus pengguna, yang dapat di gunakan secara online atau bisa di gunakan kapan pun dan di manapun.

1. **PHP**

PHP (Hypertext Prepocessor) merupakan bahasa pemrograman skrip yang diletakkan dalarn server yang biasa digunakan untuk membuat aplikasi web yang bersifat dinamis. Maksud web dinamis adalah dapat membentuk suatu tampilan web berdasarkan perrnintaan terkini, dapat dilakukan dengan menampilkan isi database ke halaman web. PHP juga di digunakan secara command line, yaitu tanpa melibatkan web server maupun browser (Kadir, 2013)

* + 1. **Sejarah PHP**

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 script yang digunakan untuk mengolah formulir dari web.

Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan dinamakan PHP/FI. Dengan perilisan kode sumber ini menjadi sumber terbuka, maka banyak programmer yang tertarik untuk ikut mengembangkan PHP (Priyanto Hidayatullah,2004).

1. **Framework**

Menurut Budi Raharjo (2015) Web Aplikasi Framework (WAF), atau yang sering disingkat Web Framework adalah satu kumpulan kode berupa pustaka (library) dan alat (tool) yang dipadukan sedemikian rupa menjadi satu kerangka kerja (framework) guna memudahkan dan mempercepat proses pengembangan aplikasi web.

Framework secara sederhana dapat diartikan kumpulan dari fungsi – fungsi atau prosedur-prosedur dan class-class untuk tujuan tertentu yang sudah siap digunakan sehingga bias lebih mempermudah dan mempercepat pekerjaan seorang programmer, tanpa harus membuat fungsi atau class dari awal.

1. **CI (CodeIgniter)**

Menurut Budi Raharjo dalam bukunya belajar otodidak framework Codeigniter (2015) CodeIgniter adalah framework web untuk Bahasa pemrograman PHP, yang dibuat oleh Rick Ellis pada tahun 2006, penemu dan pendiri Ellislab ([www.ellislab.com](http://www.ellislab.com)). Ellislab adalah suatu tim kerja yang berdiri pada tahun 2002 dan bergerak di bidang pembuatan softwere dan tool untuk para pengembang web. Sejak tahun 2014 sampai sekarang, Ellipslab telah menyerahkan hak kepemilikan Codeigniter ke British Columbia Institute of Technologi (BCIT) untuk proses perkembangan lebih lanjut. Saat ini situs resmi dari codeigniter telah berubah dari [www.ellislab.com](http://www.ellislab.com) ke [www.codeigniter.com](http://www.codeigniter.com).

Codeigniter merupakan aplikasi sumber terbuka yang berupa framework PHP dengan model MVC (model, View, Controler) untuk membangun website dinamis menggunakan PHP.

Arsitektur atau sering di sebut pola Model-View-Controler (MVC) muncul sejak era 70-an atas pemikiran Prof.Trygve Reenskaug, seorang berkebangsaan Norwegia. Dasar arsitektur ini adalah memisahkan antara logika aplikasi dengan tampilan. Menggunakan pola ini diharapkan dapat meminimalisasi penulisan perintah, sehingga resiko terjadinya bug juga minimal, serta meningkatkan efisiensi aplikasi (Antonius Bugraha widhi Pramata 2010).

Codeigniter pertama kali dirilis pada 28 Februari 2006. Versi stabil terakhir adalah versi 3.1.6.

Terdapat 3 jenis komponen yang membangun suatu MVC :

1. Model, biasanya berhubungan langsung dengan database untuk memanipulasi data (insert, update, delete), mrnangani validasi dari bagian controller, namun tidak dapat berhubungan langsung dengan bagian view.
2. View, merupakan bagian yang menangani presentation logic. Pada suatu aplikasi web bagian ini biasanya berupa file template HTML, yang diatur oleh controller. View berfungsi untuk menerima dan mempresentasikan data kepada user. Bagian ini tidak memiliki akses langsung terhadap bagian model.
3. Controller, merupakan bagian yang mengatur hubungan antara bagian model dan view, controller berfungsi untuk menerima request data dari user kemudian menentukan apa yang akan diproses oleh aplikasi.

Kelebihan Codeigniter (CI) di bandingkan dengan framework PHP lain diantaranya:

1. Performa sangat cepat.
2. Konfigurasi yang sangat minim (nearly zero configuration).
3. Banyak komunitas.
4. Dokumentasai yang sangat lengkap

**3.8 XAMPP**

Xampp merupakan paket PHP dan MySQLberbasis open source, yang dapat digunakan sebagai tool pembantu pengembangan aplikasi berbasis PHP. XAMPP mengombinasikan beberapa paket perangkat lunak dalam satu paket.(Riyanto, 2014).

**3.9 MySQL**

MySQL *merupakan* salah satu jenis *database* server yang sangat popule, hal ini disebabkan karena MySQL menggunakan bahasa SQL sebagai bahasa dasar untuk mengakses databasenya. MySQL bersifat *Open Source*, software ini dilengkapi dengan *source code* (kode yang dipakai untuk membuat MySQL), bentuk executablenya atau kode yang dapat dijalankan secara langsung dalam sistem operasi. (Kadir, 2010).

* 1. **Alat Bantu Perancangan Sistem**

Alat bantu yang di gunakan dalam pembuatan sistem ini adalah *Unifield Modeling Language (*UML). UML itu sendiri adalah bahasa permodelan yang berorientasi objek yang digunakan untuk menganalisa dan merancang sistem yang dimulai dari kelas-kelas dan komponen-komponen sistem.

UML merupakan bahasa visual untuk permodelan dan komunikasi mengenai sebuah sistem dengan diagram dan teks-teks pendukung. UML muncul karena adanya kebutuhan permodelan visual untuk menspesifikasikan , menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. (Rosa A.S dan M.Shalahuddin, 2014).

* + 1. **Use Case Diagram**

*Use case* atau diagram *use case* merupakan permodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat.

*Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. (Rosa A.S dan M.Shalahuddin, 2013).

Tabel 3.1 Simbol-simbol dalam *Use Case Diagram*

Sumber: Rosa, 2013.

| **Simbol** | **Nama** | **Keterangan** |
| --- | --- | --- |
| Description: use_case.png | *Use Case* | Fungsionalitas yang disediakan oleh sistem sebagai unit atau *actor* yang biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*. |
| Description: use_case - Copy.png | *Actor* | Orang atau proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan di buat di luar sistem informasi yang akan di buat itu sendiri |
| Description: use_case - Copy (2).png | *Assosiation* | Komunikasi antara *actor* dan *use case* yang berpartisipasi pada *use case* atau *use case* yang memiliki interaksi dengan aktor.*.* |
| Description: use_case - Copy (3) - Copy.png | *Relasi extend* | Relasi *use case* tambahan ke sebuah  *use case* dimana *use case* yang di tambahkan dapat berdiri sendiri. |
| Description: use_case - Copy (4).png | *Relasi include* | Memungkinkan suatu use case secara optional menggunakan fungsionalitas yang disediakan oleh usecase lainnya. |

**3.10.2 *Activity Diagram***

*Activity Diagram* menggambarkan *workflow* 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 *actor*, jadi aktivitas yang dapat dilakukan oleh sistem (Rosa, 2014). Berikut simbol yang digunakan dalam *Activity diagram:*

Tabel 3.2 Simbol-simbol dalam *Activity Diagram*

Sumber: Rosa, 2013.

|  |  |  |
| --- | --- | --- |
| SIMBOL | NAMA | KETERANGAN |
|  | *Start state* | Titik awal atau permulaan |
|  | *End state* | Titik akhir atau akhir dari aktivitas |
|  | *Activity* | *Activity* atau aktivitas yang dilakukan oleh *actor* |
|  | *Decision* | Pilihan untuk mengambil keputusan |
|  | *Transition* | Sebuah kejadian yang memicu sebuah state objek dengan cara memperbaharui satu atau lebih nilai atributnya |
|  | *State* | State dari system yang mencerminkan eksekusi ari sistem |
|  | *Join* | Untuk menggabungkan beberapa kegiatan secara pararel menjadi satu |

|  |  |  |
| --- | --- | --- |
| SIMBOL | NAMA | KETERANGAN |
|  | *Fork* | Menunjukan kegiatan yang dilakukan secara pararel |

**3.2.3 *Sequence Diagram***

*Sequence Diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan *Sequence* Diagrammaka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *Sequence Diagram* juga dibutuhkan untuk melihat skenario yang ada pada *use case* (Rosa A.S, 2014).

Tabel 3.3 Simbol-simbol dalam *Sequence Diagram*

Sumber: Rosa, 2013.

| **Simbol** | **Nama** | **Keterangan** |
| --- | --- | --- |
|  | *Actor* | Orang ataupun pihak yang akan mengelola system. |
|  | *Garis Hidup /*  *Lifeline* | Menggambarkan sebuah objek dalam sebuah sistem atau salah satu komponennya. |
|  | *Boundary* | Pemodelan bagian dari sistem yang bergantung pada pihak lain disekitarnya dan merupakan pembatas sistem dengan dunia luar. |

**.**

Su

|  |  |  |
| --- | --- | --- |
| **Simbol** | **Nama** | **Keterangan** |
|  | *Control* | Permodelan“perilaku mengatur”  khusus untuk satu atau beberapa use case saja. |
|  | *Entity* | Permodelan informasi yang harus disimpan oleh sistem yang memperlihatkan struktur data dari suatu sistem. |
|  | *Message* | Mengindifikasikan urutan komunikasi yang terjadi antar objek. |

**3.2.4 *Class Diagram***

*Class Diagram* atau diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. (Rosa A.S dan M.Shalahuddin, 2013).

Tabel 3.4 Simbol-simbol dalam *Class Diagram*

Sumber: Rosa, 2013.

|  |  |  |
| --- | --- | --- |
| **Simbol** | **Nama** | **Keterangan** |
|  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama |
|  | *Generalization* | Garis yang melambangkan konsep pewarisan dari sautu kelas ke satu atau lebih sub kelas |
| **........................** | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya |

aktivitas

Tabel 3.4 Simbol-simbol dalam *Class Diagram*

Sumber: Rosa, 2013.

|  |  |  |
| --- | --- | --- |
| **Simbol** | **Nama** | **Keterangan** |
| Description: Description: class diagram simbol | *Interface* | Sama dengan konsep interfase dalam pemprograman berorientasi objek. |
| Description: Description: class diagram simbol | *Directed Association* | Relasi antar kelas dengan makna kelas yang satu digunakan dengan kelas yang lain. Asosiasi biasanya juga disertai dengan *Multiplycity.* |
| Description: Description: class diagram simbol | *Aggregation* | Relasi antar kelas dengan makna kelas yang satu digunakan dengan kelas yang lain. Asosiasi biasanya juga disertai dengan *Multiplycity.* |