BAB II

LANDASAN TEORI

2.1. Konsep Dasar Program

Menurut Indriyawan (2007:1) menjelaskan bahwa "program adalah suatu proses untuk membuat pekerjaan menjadi lebih sederhana dan disusun teratur. Dengan demikian, proses bisa memudahkan manusia dalam menyelesaikan pekerjaannya".

Sukamto dan Shalahuddin (2013:67) mengatakan "Pemrograman terstruktur adalah konsep atau paradigma atau sudut pandang pemrograman yang membagi-bagi program berdasarkan fungsi-fungsi atau prosedur-prosedur yang dibutuhkan program komputer.

Berdasarkan uraian diatas maka dapat disimpulkan bahwa program adalah rangkaian instruksi-instruksi dalam bahasa komputer yang disusun secara logika dan sistematis.

2.1.1. Pengertian Aplikasi

Menurut Hendrayudi (2008:143) mendeskripsikan bahwa, "Aplikasi adalah kumpulan perintah program yang dibuat untuk melakukan pekerjaan-pekerjaan tertentu (khusus)".

Menurut Arifin (2009:13) "Aplikasi merupakan perangkat lunak pendukung yang memiliki fungsi tertentu untuk membantu pengguna dalam menyelesaikan tugas tertentu untuk membantu pengguna dalam menyelesaikan tugas dan pekerjaan-pekerjaan tertentu".

Berdasarkan pengertian diatas, penulis menyimpulkan aplikasi adalah perangkat lunak yang memiliki kumpulan perintah program yang memiliki fungsi tertentu untuk melakukan pekerjaan-pekerjaan tertentu (khusus) guna membantu pengguna dalam menyelesaikan pekerjaan tersebut.

2.1.2. Pengertian *Database*

Menurut Winarno dan Utomo (2010:142) "*Database* atau biasa disebut basis data merupakan kumpulan data yang saling berhubungan. Data tersebut biasanya terdapat dalam tabel-tabel yang saling berhubungan satu sama lain, dengan menggunakan *field*/kolom pada tiap tabel yang ada".

Menurut Pratama (2014:17), "Umumnya sebuah basis data memiliki satu atau beberapa buah table. Setiap table memiliki *field* masing-masing. Kedalam table dan *field* inilah data disimpan oleh pengguna melalui tatap muka aplikasi yang disediakan atau langsung melalui perintah diterminal (*command line*)".

Dari pengertian diatas penulis menyimpulkan *Database* adalah sekumpulan file yang saling berhubungan yang menyimpan data dan tersimpan dalam sebuah media penyimpanan.

2.1.3. Visual Basic.Net

Menurut Wardana (2008:11), "Visual Basic.NET adalah salah satu program beroreintasi objek, selain itu ada pula program java dan C++ yang juga berbasis objek".

Menurut Hidayatullah (2012:8), "Visual Basic.NET adalah salah satu dari kumpulan *tools* pemograman yang terdapat pada paket Visual Studio.NET. Pada

Visual Studio.NET terdapat beberapa *tools* pemograman lain, seperti: VisualC++, Visual C#.NET, dan Visual J#.NET".

Dari pengertian diatas penulis menyimpulkan Visual Basic.Net adalah sebuah alat untuk mengembangkan dan membangun aplikasi yang bergerak di atas sistem.

2.1.4. MySQL

MySQL adalah salah satu jenis database server yang terkenal. MySQL termasuk jenis RDBMS (*Relational Database Management System*). MySQL ini mendukung bahasa pemprograman PHP. MySQL juga mempunyai query atau bahasa SQL (*Structure Query Languange*) yang simple dan menggunakan escape character yang sama dengan PHP (Kurniawan, 2010:16).

Menurut Anhar (2010:12) mengatakan bahwa "MySQL (*My Structure Query Language*) adalah sebuah perangkat lunak sistem manajemen basis data SQL (*Database Management System*) atau DBMS dari sekian banyak DBMS, seperti Oracle, MS SQL, Portagre SQL, dan lain-lain.

Dari pengertian diatas penulis menyimpulkan MYSQL adalah perangkat lunak sistem manajemen basis data yang mempunyai query atau bahasa SQL (Structure Query Languange) yang simple yang sama dengan PHP.

2.2. Peralatan Pendukung (*Tools Program*)

Peralatan pendukung yang digunakan oleh penulis dala perancangan meliputi konsep pengertian komponen ERD, LRS, *Use case diagram, Activity diagram, User interface, XAMPP* dan Pengujian *blackbox*.

2.2.1. *Entity Relationship Diagram* (ERD)

Menurut Rosa dan Shalahuddin (2015:50) dalam bukunya yang berjudul Rekayasa Perangkat Lunak ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan menggunakan OODBMS maka perancangan basis data tidak perlu menggunakan ERD. ERD memiliki beberapa aliran notasi seperti notasi Chen, Baker, notasi Crow's Foot, dan beberapa notasi lain. Namun yang banyak digunakan adalah notasi dari Chen.

Entity Relationship Diagram adalah diagram yang menggambarkan keterkaitan antartabel beserta dengan *field-field* di dalamnya pada suatu database sistem. Sebuah database memuat minimal sebuah tabel dengan sebuah atau beberapa buah *field* (kolom) di dalamnya. Namun pada kenyataannya, database lebih sering memiliki lebih dari satu buah tabel (dengan beberapa *field* di dalamnya). Setiap tabel umumnya memiliki keterkaitan hubungan. Keterkaitan antartabel ini bisa disesebut dengan Relasi. Terdapat tiga buah jenis relasi antar tabel didalam bagan ERD (Pratama, 2014:49).

1. *One to One* (Satu ke Satu)

Relasi ini menggambarkan hubungan satu *field* pada tabel pertama ke satu *field* pada tabel kedua. Relasi ini paling sederhana. Sebagai contoh, pada sistem informasi perpustakaan terdapat tabel Buku (dengan *field* Kode_Buku, Kode_Katagori, Kode_Penulis, Nama_Penulis, Judul, Penerbit) dan tabel Katagori (Kode_Katagori, Nama_Katagori, Alamat). *Field* Kode_Katagori memilki keterkaitan (relasi) satu ke satu pada table Buku dan table Katagori.

2. *One to Many* (Satu ke Banyak)

Relasi ini menggambarkan hubungan satu *field* pada tabel pertama ke dua atau beberapa buah *field* di table kedua.

3. *Many to Many* (Banyak ke Banyak)

Sebagai contoh, sebuah sistem informasi sekolah memiliki penggunaan guru dan siswa di dalamnya. Sistem informasi ini memiliki sebuah database bernama sisfosekolah dengan tiga buah tabel didalamnya. Ketiga tabel tersebut adalah tabel Guru(memuat field NIP, Nama_Guru, Jabatan, Pangkat_Golongan, Alamat), tabel Mata Pelajaran (memuat field Kode_Mata_Pelajaran, Nama_Mata_Pelajaran), dan tabel Mengajar.

Berdasarkan pengertian di atas penulis menyimpulkan ERD (Entity Relationship Diagram) merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi.

2.2.2. *Logical Relationship Structure* (LRS)

Menurut Frieyadie (2007:13) "LRS merupakan hasil dari pemodelan *Entity Relational Ship* (ER) beserta atributnya sehingga bisa terlihat hubungan-hubungan antar entitas". Dalam pembuatan LRS terdapat 3 hal yang dapat mempengaruhi menurut Frieyadie (2007:13), yaitu:

1. Jika tingkat hubungan (*cardinality*) satu pada satu (*one-to-one*), maka di gabungkan dengan entitas yang lebih kuat (*strong entity*), atau digabungkan dengan entitas yang memiliki atribut yang lebih sedikit.

2. Jika tingkat hubungan (*cardinality*) satu pada banyak (*one-to-many*), maka hubungan relasi atau digabungkan dengan entitas yang tingkat hubungannya banyak.
3. Jika tingkat hubungan (*cardinality*) banyak pada banyak (*many-to-many*), maka hubungan relasi tidak akan digabungkan dengan entitas manapun, melainkan menjadi sebuah LRS.

Menurut Hasugian dan Shidiq (2012:608) memberikan batasan bahwa LRS adalah "sebuah model sistem yang digambarkan dengan sebuah *diagram*-ER akan mengikuti pola atau aturan permodelan tertentu dalam kaitanya dengan konvensi ke LRS". Perubahan yang terjadi yaitu mengikuti aturan-aturan sebagai berikut:

1. Setiap entitas akan diubah kebentuk kotak.
2. Sebuah atribut relasi disatukan dalam sebuah kotak bersama entitas jika hubungan yang terjadi pada *diagram*-ER 1:M (relasi bersatu dengan *cardinality* M) atau tingkat hubungan 1:1 (relasi bersatu dengan *cardinality* yang paling membutuhkan referensi).
3. Sebuah relasi dipisah dalam sebuah kotak tersendiri (menjadi entitas baru) jika tingkat hubunganya M:M (*many to many*) dan memiliki *foreign key* sebagai *primary key* yang diambil dari kedua entitas yang sebelumnya saling berhubungan.

Berdasarkan uraian di atas penulis menguraikan LRS merupakan hasil proses transformasi dari ERD dimana *primary key* yang terdapat pada masing-masing relasi akan masuk pada entitas yang lebih kuat.

2.2.3. *Use Case Diagram*

Diagram *use case* merupakan pemodelan untuk kelakuan (*behavior)* sistem informasi yang akan dibuat. *Use case* mendiskripsikan 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 dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu (Rosa & Shalahuddin, 2015:155).

Use Case merupakan metode berbasis teks untuk menggambarkan dan mendokumentasikan proses yang komplek. Use Case menambahkan detail untuk kebutuhan yang telah dituliskan pada definisi sistem kebutuhan. Use Case dikembangkan oleh analisis sistem bersama-sama dengan pengguna. Pada tahap selanjutnya, berdasarkan use case ini, analisis menyusun model data dan model proses (Fatta 2007:91).

Dari pengertian di atas penulis menyimpulkan *Use Case Diagram* adalah gambaran graphical dari beberapa atau semua *actor, use case*, dan interaksi diantaranya yang memperkenalkan suatu sistem.

2.2.4. *Activity Diagram*

Menurut Rosa dan Shalahuddin (2015 : 161), "Diagram aktivitas atau *activity diagram* mengambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak".

Menurut Triandi dan Suardika (2012:43) mengatakan bahwa "Activity Diagram adalah penggambaran *Flow of Activites* yang terdapat di dalam *Use case Description* kedalam notasi UML".

Berdasarkan pengertian di atas penulis menyimpulkan *Activity Diagram* menggambarkan tentang aktifitas yang terjadi pada sistem. Dari pertama sampai akhir, diagram ini menunjukkan langkah – langkah dalam proses kerja sistem yang kita buat.

2.2.5. User Interface

User interface merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi (Arhami, 2007:14).

Menurut Satzinger (2010:530) *User Interface* adalah bagian dari sebuah sistem informasi yang membutuhkn interaksi pengguna untuk membuat *input* dan *ouput*.

Berdasarkan uraian di atas penulis menyimpulkan *User interface* merupakan bentuk tampilan grafis yang berhubungan langsung dengan pengguna. Antarmuka pengguna berfungsi untuk menghubungkan antara pengguna dengan sistem operasi, sehingga komputer tersebut bisa digunakan.

2.2.6. *XAMPP*

Menurut Imansyah (2010:4) mengatakan bahwa "XAMPP adalah installer yang membundel Aphace, PHP, dan MySQL untuk windows dalam satu paket, dengan menginstal XAMPP anda bisa menjadikan komputer menjadi server".

Menurut Pratama (2014:440), "*Xampp* adalah aplikasi *web server* bersifat instan (siap saji) yang dapat digunakan baik di sistem operasi Linux maupun dari sistem operasi *Windows*".

Dari pengertian diatas penulis dapat menyimpulkan bahwa, *Xampp* merupakan paket PHP dan *MySQL* berbasis *open source* yang bersifat instan, yang

dapat digunakan baik di sistem operasi Linux maupun dari sistem operasi *Windows*.

2.2.7. Pengujian *Black Box*

Menurut Sukamto dan Shalahuddin (2013:275) *black box testing* adalah "menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan."

Menurut Simarmata (2010:316) *Black box* testing meliputi beberapa pengujian, yaitu:

1. Pengujian fungsional (*functional testing*)

   Pada jenis pengujian ini, perangkat lunak diuji untuk persyaratan fungsional. Pengujian dilakukan dalam bentuk tertulis untuk memeriksa apakah aplikasi berjalan seperti yang diharapkan

2. Pengujian Tegangan (*stress testing*)

   Pengujian tegangan berkaitan dengan kualitas aplikasi di dalam lingkungan. Idealnya adalah untuk menciptakan sebuah lingkungan yang lebih menuntut aplikasi, tidak seperti saat aplikasi dijalankan pada beban kerja normal.

3. Pengujian Beban (*load testing*)

   Pada pengujian beban, aplikasi akan diuji dengan beban berat atau masukan, seperti yang terjadi pada pengujian situs web, untuk mengetahui apakah aplikasi gagal atau kinerjanya menurun.

Berdasarkan uraian di atas penulis menyimpulkan *Black box testing* adalah metode dimana penguji atau tester hanya mengetahui apa yang harus

dilakukan suatu *softwar*e. Penguji tidak mengetahui bagaimana s*oftw*are tersebut beroperasi. Jadi penguji hanya menerima hasil dari apa yang dimasukkan (*input*) tanpa mengetahui bagaimana atau mengapa bisa demikian.