BAB II

LANDASAN TEORI

2.1. Konsep Dasar Program

Agar komputer dapat melakukan sesuai kebutuhan, maka komputer perlu diberikan instruksi. Instruksi tersebut dibentuk menjadi sebuah program. Pemrograman komputer bukan hanya pengetikan instruksi, akan tetapi instruksi tersebut haruslah memiliki fungsi didalam program tersebut sehingga tercapailah suatu program yang dapat memecahkan suatu masalah dan memberikan kemudahan bagi penggunanya.

2.1.1. Program

Bentuk dari berbagai macam jenis aplikasi yang dipergunakan dalam bidang bisnis ataupun dalam bidang ilmiah yang berguna untuk menghasilkan suatu laporan, informasi atau tujuan yang diinginkan berupa rangkaian instruksi dalam bahasa komputer yang disusun secara logis dan sistematis biasa disebut dengan program.

Menurut Ropianto dkk (2018:01) Mengemukakan “Program adalah kumpulan atau runtunan instruksi untuk penyelesaian suatu masalah tersebut.”

2.1.2. Bahasa Pemrograman

Bahasa pemrograman sangat membantu seorang *programmer* dalam menentukan data mana yang akan diolah oleh komputer, bagaimana data ini akan disimpan atau diteruskan, dan jenis langkah apa yang akan diambil dalam berbagai situasi.

Menurut Ropianto dkk (2018:1) "Bahasa Pemrograman adalah agar program yang kita berikan dimengerti komputer maka kita harus memberikan dengan bahasa yang dimengerti oleh komputer. Bahasa yang digunakan untuk menulis program yang dapat dimengerti komputer, disebut dengan bahasa pemrograman."

Secara umum bahasa pemrograman dibagi menjadi 3 tingkatan yaitu :

a. Bahasa Tingkat Rendah (*low level language*)

   Bahasa pemrograman yang berorientasi pada mesin. Pemrograman yang menggunakan bahasa ini harus dapat berpikir berdasarkan logika mesin komputer, sehingga bahasa ini dinilai kurang fleksibel dan sulit dipahami oleh pemula.

b. Bahasa Tingkat Menengah (*middle level language*)

   Bahasa pemrograman yang menggunakan aturan-aturan grammatical dalam penulisan pernyataan, mudah dipahami dan memiliki instruksi-instruksi tertentu yang dapat langsung diakses komputer.

c. Bahasa Tingkat Tinggi (*high level language*)

   Bahasa pemrograman yang dalam penulisan pernyataan mudah dipahami secara langsung. Bahasa program ini terbagi menjadi 2 yaitu *procedur oriented languge* dan *problem oriented language*.

Salah satu tahap dari pengembangan suatu program adalah menerjemahkan atau mengkodekan rancangan terperinci yang telah dibuat menjadi suatu program komputer yang siap pakai. Dengan menerjemahkan berarti kita melakukan penulisan program dengan bahasa komputer yang kita kuasai. Sedangkan untuk pembuatan Tugas Akhir ini dibuat dengan menggunakan bahasa program *Java* dan menggunakan aplikasi *NetBeans* IDE 8.2.

1. XAMPP

Menurut Iqbal (2019:15) “XAMPP merupakan sebuat *software* web *server apache* yang di dalamnya sudah tersedia *database server mysql* dan *support php programming*. XAMPP merupakan *software* yang mudah digunakan, gratis, dan mendukung instalasi di *Linux* dan Windows.”

Menurut Bay Haqi & Setiawan (2019:8) “XAMPP adalah perangkat lunak (*free software*) yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program.” Fungsi XAMPP sendiri sebagai server yang berdiri sendiri (*localhost*), yang terdiri dari beberapa program, antara lain: *Apache*, HTTP *Server*, MySQL, *database*, dan penerjemah bahasa yang ditulis dengan bahasa pemrograman PHP dan *Perl*.

2. PHP (*Perl Hypertext Preprocessor*)

Menurut Ropianto dkk (2018:2) “PHP merupakan singkatan dari PHP: *Perl Hypertext Preprocessor*. Skrip PHP dieksekusi di server dan hasilnya dikirimkan ke client (browser),” PHP mendukung berbagai jenis *database*. (contoh : MYSQL, Informix, Oracle, Sybase, Solid, PostgreSQL, Generic ODBC, dll).

Menurut Bay Haqi & Setiawan (2019:9) “PHP adalah bahasa skrip pemrograman yang dapat ditanamkan atau disisipkan ke dalam HTML. PHP banyak dipakai untuk memrogram situs web dinamis. PHP dapat digunakan untuk membangun sebuah CMS.”

3. HTML (*Hyper Text Markup Language*)

> Menurut Suryana & Koesheryatin (2014:29) Mengemukakan bahwa :
> *Hyper Text Markup Language* (HTML) adalah bahasa yang digunakan untuk menulis halaman web. HTML merupakan pengembangan dari standar pemformatan dokumen teks, yaitu *Standard Generalized Markup Language* (SGML), HTML pada dasarnya merupakan dokumen ASCII atau teks biasa, yang dirancang untuk tidak tergantung pada suatu sistem operasi tertentu.

4. MySQL

Menurut MF (2018:21) “MySQL adalah sistem manajemen *database* SQL yang sifatnya *open source* (terbuka) dan paling banyak digunakan saat ini. Sistem *database* MySQL mampu mendukung beberapa fitur seperti *multithreaded, multiuser,* dan SQL *database management systems* (DBMS). “

5. Javascript

Menurut Winarno dkk (2014:129), “*Javascript* merupakan bahasa *scripting* client side yang sangat populer. Hampir semua *programmer* web menggunakan *javascript* untuk memberi efek pemrograman di halaman. *Javascript* tidak hanya berdiri sendiri, tapi *javascript* juga menjadi dasar yang bisa digunakan untuk teknologi lainnya, seperti *Ajax*, *jQuery* dan *jQuery Mobile*.”

Menurut Suryana & Koesheryatin (2014:181) “*JavaScript* adalah bahasa script berdasar pada objek yang memperbolehkan pemakai untuk mengendalikan banyak aspek interaksi pemakai pada suatu dokumen HTML. Di mana objek tersebut dapat berupa suatu *window*, *frame*, URL, dokumen, *form*, *button*, atau *item* yang lain.”

2.1.3. Basis Data

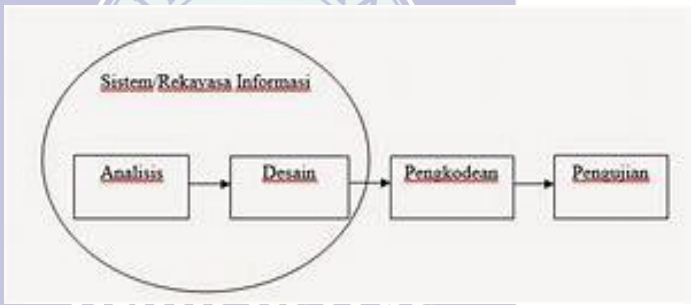
Menurut Lubis (2016:3) “Basis data adalah tempat berkumpulnya data yang saling beruhubungan dalam suatu wadah (organisasi/perusahaan) bertujuan agar dapat mempermudah dan mempercepat untuk pemanggilan atau pemanfaatan kembali data tersebut.”

Menurut Sukamto dan Shalahuddin (2016:43) “Sistem basis data adalah sistem terkomputerisasi yang tujuan utamanya adalah memelihara data yang sudah diolah atau informasi dan membuat informasi tersedia saat dibutuhkan. Pada

intinya basis data adalah media untuk menyimpan data agar dapat diakses dengan mudah dan cepat."

2.1.4. Model Pengembangan Perangkat Lunak

Menurut Sukamto dan Shalahuddin (2016:28) "Model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial linier (*sequential linear*) atau alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut dimulai dari analisis, desain, pengodean, pengujian, dan tahap pendukung (*support*)." Berikut adalah model air terjun:



Sumber: Sukamto dan Shalahuddin (2016:29)

Gambar II.1
Model *Waterfall*

1. Analisis Kebutuhan Perangkat Lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk menspesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan oleh *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu untuk didokumentasikan.

2. Desain

   Desain perangkat lunak adalah proses multi langkah yang fokus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Pembuatan Kode Program

   Desain harus ditranslasikan kedalam program perangkat lunak. Hasil dari tahap ini adalah program komputer sesuai dengan desain yang telah dibuat pada tahap desain.

4. Pengujian

   Pengujian fokus pada perangkat lunak secara dari segi lojik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan (*error*), dan memastikan keluaran yang dihasilkan sesuai dengan yang diinginkan.

5. Pendukung (*support*) dan Pemeliharaan (*maintenance*)

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke user. Perubahan bisa terjadi karena adanya kesalahan yang muncul dan tidak terdeteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisis spesifikasi untuk perubahan perangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.2. *Tools Program*

2.2.1. ERD (*Entity Relationship Diagram*)

Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan Entity Relationship Diagram (ERD). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika.

> Menurut Trisyanto (2017:89), "ERD adalah gambar atau diagram yang menujukan informasi dibuat, disimpan, dan digunakan dalam sistem bisnis, entitas biasanya menggambarkan jenis informasi yang sama, dalam entitas digunakan untuk menghubungkan antar entitas yang sekaligus menujukan hubungan antar data. Pada akhirnya ERD bisa juga digunakan untuk menunjukkan aturan-aturan bisnis yang ada pada sistem informasi yang akan dibangun."

Menurut Lubis (2016:38) "ERD adalah suatu pemodelan berbasis pada persepsi dunia nyata yang mana terdiri dari kumpulan objek dasar yang disebut dengan entitas (*entity*) dan hubungan diantara objek-objek tersebut dengan menggunakan perangkat konseptuan dalam bentuk diagram."

Menurut Sukamto dan Shalahuddin (2016:50) "*Entity Relationship Diagram* (ERD) adalah pemodelan awal basis data yang dikembangkan berdasarkan teori himpunan dalam bidang matematika untuk pemodelan basis data relational."

Dalam sistem *Entity Relationship Diagram* (ERD), terdapat beberapa istilah penting diantaranya:

1. Entitas (*Entity*)

   Suatu entitas yang dapat berupa orang, tempat, obyek, atau kejadian yang dianggap penting bagi perusahaan, sehingga segala atributnya harus dicatat dan disimpan dalam basis data.

2. Atribut (*Atribute*)

   Setiap entitas mempunyai karakterisik tertentu yang dinamakan dengan atribut.

3. Relasi (*relationship*)

   Hubungan antara dua atau lebih entitas yang saling berkaitan.

   Ada tiga tipe relasi (*relationship*), yaitu:

   a. *One-to-one relationship* (1:1) Dimana maximum *cardinality* setiap *entity* adalah 1. Contoh: Satu nasabah bank hanya memiliki satu *account*.

   b. *One-to-many relationship* (1:N). Dimana maximum cardinality dari suatu *entity* adalah 1 dan maximum *cardinality* dari entity lain adalah N.

      Contoh: Satu nasabah bank dapat memiliki lebih dari satu *account*.

   c. *Many-to-many relationship* (M:N). Dimana maximum *cardinality* kedua *entity* yang berhubungan adalah N. Contoh : Satu nasabah dapat memilliki beberapa *account* dan satu *account* dapat dimiliki oleh beberapa nasabah (rekening bersama).

4. *Identifier*

   Merupakan nama *atribute* yang digunakan untuk mengidentifikasi suatu entitas, Ada tiga jenis identifier diantaranya *Primary Key*, *Secondary Key*, dan *Foreign Key*. Dalam penulisan tugas akhir ini hanya menggunakan *Primary Key* dan *Foreign Key*, berikut ini penjelasan dari *Primary Key* dan *Foreign Key*:

   a. *Primary Key* merupakan suatu kode identifikasi yang bersifat unik yang ditunjukkan oleh masing-masing *record* dalam sistem. Tujuan dari *Primary key* adalah untuk menunjukkan lokasi tiap catatan di dalam suatu *file* mengenai catatan-catatan serupa.

   b. *Foreign Key* merupakan *atribute* yang merupakan *Primary key* dari relasi lain yang ditarik/dihubungkan ke suatu relasi.

5. Kardinalitas (*Cardinality*)

   Merupakan kendala-kendala yang timbul dalam hubungan antar entitas.

2.2.2. LRS (*Logical Relational Structure*)

LRS (*Logical Record Structure*) dibentuk dengan nomor dari tipe *record*. Beberapa tipe *record* digambarkan oleh kota persegi panjang dan dengan nama yang unik. Perbedaan LRS dan E-R diagram adalah nama tipe *record* berada diluar kotak *field* tipe *record* ditempatkan.

LRS merupakan hasil dari pemodelan *Entity Relationship* (ER) beserta atributnya sehingga bisa terlihat hubungan-hubungan antar entitas.

Berikut adalah cara membentuk skema database atau LRS (*Logical Record Structure*) berdasarkan *Entity Relationship Diagram:*

1. Jika relasinya satu-ke-satu, maka *foreign key* diletakan pada satu dari dua entitas yang ada atau menyatukan kedua entitas tersebut.
2. Jika relasinya satu-ke-banyak, maka *foreign key* diletakan pada entitas *Many.*
3. jika relasinya banyak-ke-banyak, maka dibuat “file konektor” yang berisi dua *foreign key* yang berasal daru kedua entitas.

2.2.3. Pengkodean

Pengkodean digunakan untuk mengklasifikasikan data yang dimasukan kedalam komputer ataupun untuk mengambil macam-macam informasi, kode dapat terbentuk dari kumpulan angka, huruf, atau simbol lainnya.

Menurut Mustakini (2016:384) “Kode digunakan untuk tujuan mengklasifikasikan data, memasukan data ke dalam komputer dan mengambil bermacam-macam informasi yang berhubungan dengannya. Kode dapat dibentuk dari kumpulan angka, huruf, dan karakter-karakter khusus (misalnya %, /, &, $, dan lain sebagainya)”. Angka merupakan simbol yang banyak digunakan pada sistem kode akan tetapi kode yang berbentuk angka lebih dari 6 digit akan sangat sulit untuk

di ingat kode numerik (*numeric code*) menggunakan 10 macam kombinasi angka di dalam kode. Kode alphabetik (*alphabetic code*) menggunakan 26 kombinasi huruf untuk kodenya. Kode alphanumerik (*alphanumeric code*) merupakan kode yang menggunakan golongan angka, huruf, dan karakter-karakter khusus meskipun kode numerik, alphabetik, dan alphanumerik merupakan kode yang paling banyak digunakan di dalam sistem informasi, tetapi kode yang lain juga mulai banyak digunakan, seperti misalnya kode batang (*barcode*). Dalam merancang kode yang baik ada beberapa hal yang harus diperhatikan, yaitu sebagai berikut :

a. Harus mudah diingat

   Agar kode mudah diingat, maka dapat dilakukan dengan cara menghubungkan kode tersebut dengan obyek yang diwakili dengan kodenya.

b. Harus unik

   Kode harus unik untuk masing-masing item yang diwakili. Unik berarti todak ada kode yang kembar.

c. Harus fleksibel

   Kode harus fleksibel sehingga memungkinkan perubahan-perubahan atau penambahan item baru dapat tetap diwakilo oleh kode.

d. Harus efisien

   Kode harus sependek mungkin, selain mudah diingat juga akan efisien bila direkam di simpanan luar komputer.

e. Harus konsisten

   Kode harus konsisten dengan kode yang telah dipergunakan.

f. Harus distandarisasi

   Kode harus distandarisasi untuk seluruh tingkatan dan departemen dalam organisasi. Kode yang tidak standart akan mengakibatkan kebingungan, salah

pengertian dan cenderung dapat terjadi kesalahan pemakai begitu juga dengan yang menggunakan kode tersebut.

g. Hindari spasi

Spasi dalam kode sebaiknya dihindari, karena dapat menyebabkan kesalahan dalam menggunakannya.

h. Hindari karakter yang mirip

Karakter-karakter yang hampit serupa bentuk dan bunyi pengucapannya sebaiknya tidak digunakan dalam kode.

Ada beberapa macam tipe dari kode yang dapat digunakan di dalam sistem informasi, antara lain :

1. Kode Mnemonik (*mnemonik code*)

Digunakan untuk tujuan supaya mudah diingat. Kode mnemonik dibuat dengan dasar singkatan atau mengambil sebagian karakter dari item yang akan di wakili dengan kode ini. Sebagai contoh kode “P” untuk mewakili pria dan kode “W” untuk wanita akrab untuk mudah diingat. Umumnya kode mnemonik menggunakan huruf, akan tetapi dapat juga menggunakan gabungan huruf dan angka misalnya barang dagangan komputer IBM pc dengan ukuran memori 640 kb, *colour monitor*, dapat dikodekan menjadi K-IBM- PC-640- CO supaya lebih mudah diingat. Kebaikan dari kode ini adalah mudah diingat dan kelemahannya adalah kode cepat terlalu panjang.

2. Kode Urut (*sequential code*)

Kode yang ini disebut juga kode seri (serial code) merupakan kode yang nilainya urut antara satu kode dengan kode berikutnya.

3. Kode Blok (*block code*)

   Kode blok (*block code*) mengklasifikasikan item ke dalam kelompok blok tertentu yang mencerminkan suatu klasifikasi tertentu atas dasar pemakaian maksimum yang diharapkan.

4. Kode Desimal (*decimal code*)

   Kode desimal (*desimal code*) mengklasifikasikan kode atas dasar 10 unit angka desimal dimulai dari angka 0 sampai dengan angka 9 atau dari 00 sampai dengan 99 tergantung dari banyaknya kelompok.

5. Kode Grup (*group code*)

   Kode grup (*group code*) merupakan kode yang berdasarkan *field-field*, dan tiap *field-field* nya mempunyai arti.

6. Kode Batang (*barcode*)

   Sebagai kumpulan kode yang berbentuk garis, dimana masing-masing ketebalan setiap garis berbeda sesuai dengan isi kodenya.

2.2.4. HIPO (*Hierarky Input Proses Output)*

1. Pengertian HIPO

Menurut Trisyanto (2017:82) “HIPO (Hierarky Input Proses Output) merupakan metodologi yang dikembangkan dan didukung oleh IBM. HIPO sebenarnya adalah alat dokumentasi program, akan tetapi sekarang, HIPO juga banyak digunakan sebagai alat desain dan teknik dokumentasi dalam siklus pengembangan sistem.”

Menurut Mustakini (2016:787) Mengungkapkan “HIPO (*Hierarchy Input Proses Output*) merupakan metodelogi yang dikembangkan dan didukung oleh IBM. HIPO sebenernya adalah alat dokumentasi program, akan tetapi sekarang HIPO juga

banyak digunakan sebagai alat desain dan teknik dokumentasi dalam siklus pengembangan sistem HIPO berbasis pada fungsi, yaitu tiap-tiap modul didalam sistem digambarkan oleh fungsi utamanya."

2. Tingkatan Diagram HIPO

Diagram HIPO didesain untuk menyediakan dokumentasi dari setiap tingkatan, ada tiga tingkatan diagram yang tercakup yaitu. berikut pengertiannya:

a. Daftar Isi Visual/ *Visual Table Of Content* (VTOC)

Diagram ini menggambarkan hubungan dari fungsi-fungsi di sistem secara berjenjang.

b. Diagram Ringkasan/ *Overview Diagram*

Diagram ini menunjukan secara garis besar hubungan dari *input*, *pocess*, dan *output*. Bagian *input* menunjukkan *item-item* data yang akan digunakan oleh bagian proses. Bagian proses berisi sejumlah langkah-langkah yang menggambarkan kerja dari fungsi. Bagian *output* berisi dengan *item-item* data yang dihasilkan atau dimodifikasi oleh langkah-langkah proses.

c. Diagram Rinci/ *Detail Diagram*

Merupakan diagram tingkatan yang paling rendah pada diagram HIPO. Diagram ini berisi dengan elemen-elemen dasar dari paket yang menggambarkan secara rinci kerja dari fungsi.

2.2.5. Diagram Alir Data (*Flowchart*)

Menurut Trisyanto (2017:80) "*Flowchart* merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program, bagan alir program dibuat dari *derivikasi* bagan alir sistem."

Menurut Mustakini (2016:795) "Bagan alir (*flowchart*) merupakan kumpulan dari notasi diagram simbolik yang menunjukkan aliran data dan urutan operasi dalam sistem."

1. Bentuk dari *flowchart* yaitu sebagai berikut:
   a. Program *Flowchart*

      Simbol-simbol yang menggambarkan proses secara terperinci dan detail antara instruksi yang satu dengan instruksi yang lainnya didalam komputer yang bersifat *logic* dan sistem.

   b. Sistem *Flowchart*

      Diagram alir berupa simbol-simbol yang menggambarkan ururtan prosedur secara detail didalam suatu sistem komputer yang digunakan untuk proses pengolahan data serta menggambarkan hubungan antara peralatan tersebut yang bersifat fisik.

2. Tehnik pembuatan *flowchart* yaitu sebagai berikut:
   a. General Way

      Teknik pembuatan flowchart dengan cara ini, digunakan dalam menyusun suatu pprogram yang menggunakan pengulangan proses secara tidak langsung. (*Non Direct Loop*).

b. Iteration Way

Teknik pembuatan flowchart dengan cara ini biasanya dipakai untuk logika program yang cepat serta bentuk permasalahan yang kompleks, dimana pengulangan proses yang terjadi bersifat langsung (*Direct Loop*).

2.2.6. Implementasi dan Pengujian Unit

Pengujian dimaksudkan untuk mengetahui fungsi-fungsi, masukan dan keluaran dan perangkat lunak sesuai spesifikasi yang dibutuhkan dan pengujian dengan metode *black box testing* memungkinkan *software* untuk membuat himpunan kondisi *input* yang akan melatih seluruh syarat-syarat fungsional suatu program.

Menurut Sukamto dan Shalahuddin (2016:275) Mengatakan "*Black Box Testing* yaitu 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".