BAB II

LANDASAN TEORI

2.1. Konsep Dasar Program

2.1.1. Pengertian Program

Menurut (abdul kadir, 2012) Program adalah kumpulan instruksi yang digunaka untuk mengatur komputer agar melakukan suatu tindakan tertentu, tanpa program komputer sesunguhnya tidak dapat berbuat apa apa

2.1.2 Bahasa Pemrograman

Menurut (Hidayat, Marlina, & Utami, 2017) Sebuah instruksi standar untuk memerintah komputer agar menjalankan fungsi tertentu. Bahasa pemrograman ini merupakan suatu himpunan dari aturan sintaks dan semantik yang dipakai untuk mendefinisikan program komputer. Bahasa ini memungkinkan seorang programmer dapat menentukan secara persis data mana yang akan diolah oleh komputer, bagaimana data ini akan disimpan/diteruskan, dan jenis langkah apa secara persis yang akan diambil dalam berbagai situasi.

Dalam pembuatan aplikasi ini penulis menggunakan beberapa bahasa pemrograman. Adapun bahasa pemrograman yang dipakai sebagai berikut :

A. Java

(Sukamto & Shalahuddin, 2016) Java merupakan salah satu dari bahasa pemrograman computer yang sudah bersifat orientasi objek, Java diciptakan atau dibuat oleh sebuah tim dari perusahaan Sun Microsytem, perusahaan *Workstation* UNIX (Sparc) yang sudah cukup terkenal.

Java merupakan bahasa pemrograman objek murni karena semua kode programnya dibungkus dalam kelas. Saat ini Sun Microsystem sudah diakuisasi Oracle C

orporation sehingga pengembangan Java diteruskan oleh Oracle Corporation.

B. NetBeans

Menurut (Enterprise, 2014) NetBeans merupakan IDE (*intergrated development environment)* untuk membuat aplikasi dengab java, PHP,C,C++ dan HTML5.

C. MySQL

Menurut (Hidayat et al., 2017) "MySQL adalah sistem manajemen database SQL yang bersifat Open Source dan paling populer saat ini. Sistem Database MySQL mendukung beberapa fitur seperti multithreaded, multi-user, dan SQL database management system (DBMS). Database ini dibuat untuk keperluan sistem database yang cepat, handal, dan mudah digunakan"

.

2.1.3. Basis Data

Sistem basis data adalah *system* 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. Menurut (Rosa A.S & M.Shalahuddin, 2016:43).

2.1.4. Model Pengembangan Perangkat Lunak (*Waterfall*)

Metode yang digunakan pada pengembangan perangkat lunak ini menggunakan model waterfall (Shalahuddin & Sukamto, 2018) yang terbagi menjadi lima tahapan yaitu :

A. Analisa Kebutuhan Perangkat Lunak

Mengidentifikasikan informasi-informasi dari *user* sehingga akan tercipta sebuah sistem komputer yang bisa melakukan tugas-tugas yang dibutuhkan oleh *user* tersebut seperti melakukan pemesanan dan konfirmasi pemesanan.

B. Desain

Dalam taham ini proses multi langkah yang focus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antar muka, dan prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke represenstasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya.

C. Pembuatan Kode Program

Dari tahap desain kemudia dilakukan dengan merancang program berbasis objek, dengan menggunakan bahas pemograman PHP, HTML dan dengan bantuan perangkat lunak NetBeans.

D. Pengujian

Pengujian dilakukan setelah kode program selesai dibuat dan program dapat berjalan. Pengujian fokus pada perangkat lunak secara dari segi lojik, fungsional, dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan meminilisir kesalahan (*error)* dan memastikan keluaran yang dihasilkan sesuai dengan apa yang diinginkan.

E. Pendukung atau Pemeliharaan

Tidak menuntut kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirim ke *user*. Perubahan ini bias terjadi karena adanya kesalahan yang muncul dan tidak tereteksi saat pengujian atau perangkat lunak harus beradaptasi dengan lingkungan baru. Tahap pendukung atau pemeliharaan dapat mengulangi proses pengembangan mulai dari analisia spesifikasi untuk prangkat lunak yang sudah ada, tapi tidak untuk membuat perangkat lunak baru.

2.2 Tools Program

2.2.1. Entity Relationship Diagram (ERD)

Menurut (Rosa A.S & M.Shalahuddin, 2016:50) Mengemukakan ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika.ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penimpanan basis data menggunakan OODBMS maka perancangan basis data tidak perlu menggunakan ERD.

A. Entitas (*Entity*)

Entitas merupakan data inti yang akan disimpan bakal table pada basis data benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer.

B. Atribut

Field atau kolom data yang butuh disimpan dalam suatu entitas.

C. Relasi

Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja.

D. Asosiasi (*Association*)

Penghubung antara relasi dan entitas di mana di kedua ujungnya memiiki *multiplicity* kemungkinan jumlah pemakaian, kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan entitas yang lain disebut dengan kardinalitas.

2.2.2. *Logical Record Structure* (LRS)

Menurut (Hidayat et al., 2017) "*Logical Record Structure* terdiri dari *link-link* diantara tipe record, Link ini menunjukkan arah dari satu tipe record lainnya. Banyak link dari LRS yang diberi tanda *field-field* yang kelihatan pada kedua *link* tipe *record*". Penggambaran LRS mulai dengan menggunakan model yang dimengerti. Dua metode yang dapat digunakan, dimulai dengan hubungan kedua model yang dapat dikonverensikan ke LRS, metode yang lain dimulai dengan ER-diagram dan langsung dikonversikan ke LRS.

Perbedaan LRS dan ERD adalah nama dan tipe record berada diluar field tipe record di tempatkan. LRS terdiri dari link-link diantara tipe record. Link ini menunjukkan arah dari satu tipe record lainnya. Banyak link dari LRS yang diberi tanda field-field yang kelihatan pada kedua *link* tipe *record*.

2.2.3. Pengkodean

Menurut (Jogiyanto Hartono, 2016:384) "Kode digunakan untuk tujuan mengklasifikasikan data, memasukkan data ke dalam komputer dan untuk mengambil bermacam-macam informasi yang berhubungan dengannya". Di dalam merancang suatu kode harus diperhatikan beberapa hal, yaitu sebagai berikut:

1). Harus mudah diingat.

Agar mudah diingat, maka dilakukan dengan cara menghubungkan kode tersebut dengan obyek yang diwakili dengan kodenya. Kode yang terlalu panjang sebaiknya dipecah menjadi bagian-bagian yang lebih pendek.

2). Harus unik.

Kode harus unik untuk masing-masing item yang diwakili. Unik berarti tidak Ada kode yang kembar.

3). Harus fleksibel.

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

4). Harus efisien.

Kode harus sependek mungkin, selain mudah diingat juga akan efisien bila direkam di simpanan luar komputer. Misalnya panjang dari kode cukup sepanjang 4 digit saja dan tidak akan efisien bila dipergunakan kode yang lebih dari 4 digit.

5). Harus konsisten.

Kode harus konsisten dengan kode yang telah dipergunakan. Misalnya perusahaan hanya membeli barang dagangan dari seorang pemasok (*supplier*) saja, maka dapat dipergunakan kode-kode barang yang sudah dipergunakan oleh pemasok.

6). Harus distandarisasi.

Kode harus distandarisasi oleh semua tingkatan dan departemen dalam organisasi. Kode yang tidak standar akan mengakibatka kebingungan,dalam pengertian dan dapat cenderung terjadi kesalahan pemakaian bagi yang menggunakan kode tersebut.

7). Spasi dihindari.

Spasi dalam kode sebaiknya dihindari, karena dapat menyebabkan kesalahan didalam menggunakan nya.

8). Hindari karakter yang mirip.

Karakter-karakter yang hampir serupa bentuk dan bunyi pengucapan sebaiknya tidak digunakan dalam kode.

9). Panjang kode harus sama.

Masing-masing kode yang sejenis harus mempunyai panjang yang sama. Misalnya panjang dari kode adalah 6 digit, maka kode 8210E sebaiknya ditulis 08210E.

Tipe dari kode ada beberapa macam tipe dari kode yang dapat digunakan dalam system informasi, diantaranya adalah kode mnemonic (*mnemonic code*), kode urut (*sequential code*), kode group (*group code*) dan kode decimal (*decimal code*). Masing-masing tipe dari kode tersebut mempunyai kebaikan dan kelemahannya tersendiri.

Dalam praktek, tipe-tipe kode yang ada dapat dikombinasikan. Menurut (Jogiyanto Hartono, 2016:386).

1. Kode Mnemonic

Kode mnemonic (*mnemonic code*) digunakan untuk tujuan supaya mudah diingat. Kode mnemonik dibuat dengan dasar singkatan atau mengambil sebagian karkater dari item yang akan diwakili dengan kode ini.

2. Kode Urut

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

3. Kode Blok

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

4. Kode Group

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

5. Kode Desimal

Kode decimal (*decimal 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.

2.2.4. HIPO (*Hierarky Input Proses Output*) (buku)

Menurut (Jogiyanto Hartono, 2016:787) "HIPO 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. HIPO berbasis pada fungsi, yaitu tiap-tiap modul di dalam sistem digambarkan oleh fungsi utamanya".

HIPO menggunakan tiga macam diagram untuk masing-masing tingkatannya menurut (Jogiyanto Hartono, 2016:788), yaitu sebagai berikut:

1. *Visual Table Of Contents* (VTOC)

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

2. *Overview Diagrams*

Overview diagrams menunjukan secara garis besar hubungan dari input, proses, dan output. Bagian input menunjukan item-item data yang akan

digunakan oleh bagian proses. Bagian proses berisi sejumlah langkah-langkah yang menggambarkan kerja dan fungsi. Bagian output berisi dengan item-item data yang dihasilkan atau dimodifikasi oleh langkah-langkah proses.

3. *Detail Diagrams*

Detail diagrams merupakan diagram tingkatan yang paling rendah di diagram HIPO. Diagram ini berisi dengan elemen-elemen dasar dari paket yang menggambarkan secara rinci kerja dari fungsi.

2.2.5. Diagram Alir Program (*Flowchart*)

1. Pengertian Bagan Alir (*flowchart*)

"Bagan alir (*flowchart*) adalah bagan (*chart*) yang menunjukan alir (*flow*) di dalam program atau prosedur sistem secara logika. Bagan alir digunakan terutama untuk alat bantu komunikasi dan untuk dokumentasi". Menurut (Jogiyanto Hartono, 2016:795). Pada waktu akan menggambar suatu bagan alir, analisis sistem atau pemrogram dapan mengikuti pedoman-pedoman sebaagai berikut ini.

1. Bagan alir sebaiknya digambar dari atas ke bawah dan dimulai dari bagia kiri dari suatu halaman.
2. Kegiatan di dalam bagan alir harus ditunjukan dengan jelas.
3. Harus ditunjukan dari mana kegiatan akan dimulai dan dimana akan berakhir.
4. Masing masing kegiatan di dalam bagan alir sebaiknya digunakan suatu kata yang mewakili suatu pekerjaan, misalnya:
    1. "Persiapkan" dokumen
    2. "Hitung" gaji
5. Masing-masing kegiatan di dalam bagan alir harus di dalam urutan yang semestinya.

6. Kegiatan yang terpotong dan akan disambung ditempt lain harus ditunjukkan dengan jelas menggunakan symbol penghubung.
7. Gunakanlah symbol symbol-simbol bagan alir yang standar.

2. Bentuk *Flowchart*

A. Program *Flowchart*

Menurut (Rizq, Trisna, Ahmadi, & Suardika, n.d.) "Program Flowchart Yaitu urutan instruksi yang digambarkan dengan simbol tertentu untuk memecahkan masalah dalam suatu program". Hal yang paling penting dalam pembuatan flowchart yaitu:

1). Jalannya proses digambarkan dari halaman atas ke bawah dan dari kiri ke kanan dan diberikan tanda panah untuk memperjelas.
2). Aktivitas yang digambarkan harus didefinisikan secara hati-hati dan definisi ini harus dapat dimengerti oleh pembacanya.
3). Hindari pengulangan proses yang tidak perlu dan logika yang berbelit sehingga jalannya proses menjadi singkat.
4). Sebuah flowchart diawali dari satu titik start dan diakhiri dengan *end*.
5). Setiap langkah dari aktivitas harus berada pada urutan yang benar.

B. Sistem *Flowchart*

Menurut (Jogiyanto Hartono, 2016:796) mengemukakan bahwa "Bagan alir sistem (*system flowchart*) merupakan bagan yang menunjukkan arus pekerjaan secara keseluruhan dari sistem. Bagan ini menjelaskan urut-urutan dari prosedur-prosedur yang ada di dalam sistem. Bagan alir sistem menunjukkan apa yang dikerjakan di sistem".

C. Teknik Pembuatan

Menurut (Rizq et al., n.d.) Adapun teknik pembuatan program flowchart ini dibagi menjadi dua bagian yaitu sebagai berikut:

1). General Way

Teknik pembuatan flowchart dengan cara ini biasanya dipakai didalam menyusun logika suatu program, yang menggunakan pengulangan proses secara tidak langsung (Non-Direct-Loop).

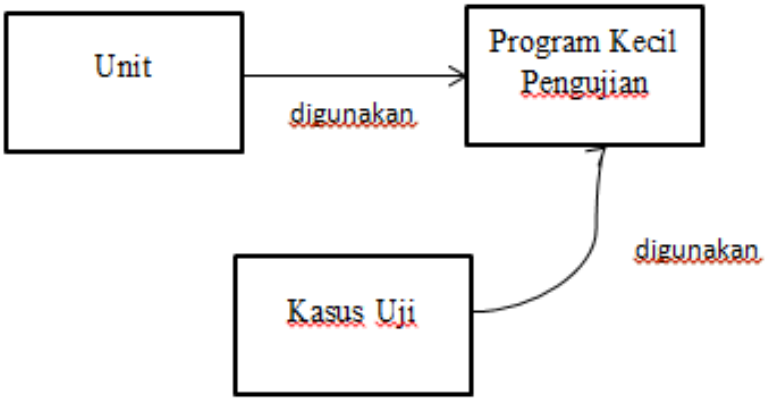
2). Iteration Way

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

Menurut (Sukamto & Shalahuddin, 2016) Pengujian unit fokus pada usaha verifikasi pada unit yang terkecil pada desain perangkat lunak (komponen atau modul perangkat lunak). Setiap unit perangkat lunak diuji agar dapat diperiksa apakah aliran masukan (*input*) dan keluaran (*output*) dari unit sudah sesuai dengan yang diinginkan.

Pengujian unit biasanya dilakukan saat kode program dibuat. Karena dalam sebuah perangkat lunak banyak memiliki unit-unit kecil maka untuk menguji unit-unit kecil ini biasanya dibuat program kecil (*main program*) untuk menguji unit-unit perangkat lunak. Unit disini secara fisik dapat berupa prosedur atau fungsi, sekumpulan prosedur atau fungsi yang ada dalam satu berkas (*file*) jika dalam pemrograman terstruktur, atau kelas, bias juga kumpulan kelas dalam satu *package* dalam pemrograman berorientasi objek. Ilustrasi pengujian unit adalah sebagai berikut :



Gambar 3.1 Pengujian Unit

Sumber (Sukamto & Shalahuddin, 2016)

Setiap unit diuji menggunakan sebuah program pengujian yang khusus dibuat untuk menguji sebuah unit menggunakan kumpulan kasus uji yang didefinisikan. Pengujian untuk validasi memiliki beberapa pendekatan sebagai berikut :

1. *Black-Box Testing* (pengujian kotak hitam)

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.

Pengujian kotak hitam dilakukan dengan membuat kasus uji yang bersifat mecoba semua fungsi dengan memakai perangkat lunak apakah sesuai dengan spesifikasi yang dibutuhkan. Kasus uji yang dibuat untuk melakukan pengujian kotak hitam harus dibuat dengan kasus benar dan kasus salah, misalkan untuk kasus proses *login* maka kasus uji yang dibuat adalah :

a. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang benar

b. Jika *user* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang salah, misalnya nama pemakai benar tapi kata sandi, atau sebaliknya, atau keduanya salah.

UNIVERSITAS