BAB II

LANDASAN TEORI

2.1. Konsep Dasar Program

2.1.1. Program

"Program adalah perangkat lunak (*software*) yang sebenarnya merupakan tuntunan instruksi yang ditulis dalam bentuk kode-kode menggunakan bahasa pemrograman tertentu dan telah dikompilasi dengan menggunakan compiler yang sesuai" (Tabrani & Aghniya, 2019).

2.1.2. Bahasa Pemrograman

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 (Hidayat, Marlina, & Utami, 2017).

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

1. Java

   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 Corporation sehingga pengembangan Java diteruskan oleh Oracle Corporation (Sukamto & Shalahudin, 2016).

2. NetBeans

   NetBeans merupakan salah satu IDE yang dikembangkan dengan bahasa pemrograman java. NetBeans mempunyai lingkup pemrograman java terintergrasi dalam suatu perangkat lunak yang di dalamnya menyediankan pembangunan pemrograman GUI, text editor, complier, dan interpreter. NetBeans adalah sebuah perangkat lunak open source sehingga dapat digunakan secara gratis untuk keperluan komersial maupun nonkomersial yang didukung oleh SunMicrosystem (Rusmayanti, 2014).

3. MySQL

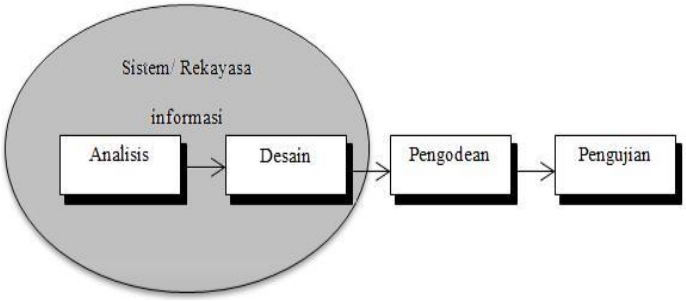
   MySQL adalah database yang reliable dan dapat digunakan sebagai database server. MySQL bersifat multiplatform. MySQL adalah sebuah aplikasi under shell yang artinya untuk menkonfigurasi mysql di perlukan perintah-perintah tertentu. PhpMyadmin adalah sebuah aplikasi yang ditulis dalam PHP yang memungkinkan pengguna mengadministrasikan database MySQL. Dengan PhpMyadmin konfigurasi MySQL dapat dilakukan dengan mudah dan cepat. Adapun beberapa kelebihan MySQL yaitu mudah dalam instalasi, mampu menampung record ratusan giga, dan merupakan *software* yang *free* (Rizq, Trisna, Ahmadi, & Suardika, 2015).

2.1.3. Basis Data

Basis data atau *database* adalah kumpulan informasi yang disimpan di dalam computer secara sistematik sehingga dapat diperiksa menggunakan suatu program computer untuk memperoleh informasi basis data tersebut. Dalam konsep *database*, urutan atau hierarki *database* sangatlah penting (Supono, 2018).

2.1.4. Model Pengembangan Perangkat Lunak (*Waterfall*)

Metode yang digunakan pada pengembangan perangkat lunak ini menggunakan model *waterfall* (Sukamto & Shalahudin, 2016) yang terbagi menjadi lima tahapan yaitu :



Sumber: (Sukamto & Shalahuddin, 2016)

Gambar II.1

Ilustrasi 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 data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan kerepresentasi desain agar dapat diimplementasikan mennjadi program pada tahap selanjutnya. Desain perangkat lunak yang dihasilkan pada tahap ini juga perlu didokumentasikan.

3. Pembuatan kode program

   Desain harus ditranslasikan ke dalam 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 secar dari segi logik dan fungsional dan memastikan bahwa semua bagian sudah diuji. Hal ini dilakukan untuk meminimalisir kesalahan dan memastikan keluaran yang dihasilakan sesuai dengan yang diinginkan.

5. Pendukung (*support*) atau pemeliharaan (*maintenance*)

   Tidak menutup kemungkinan sebuah perangkat lunak menggalami perubahan ketika sudah dikirimkan ke *user*. Perubahan bisa saja 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 spesifik untu perubahan, tapi tidak membuat perangkat lunak baru.

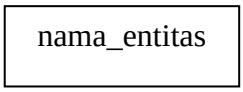
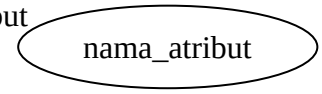
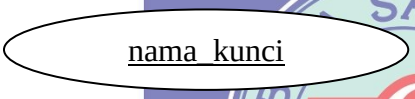
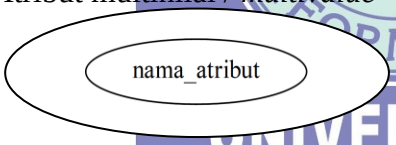
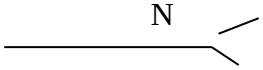
2.2. *Tools Program*

2.2.1. *Entity Relationship Diagram* (ERD)

ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan basis data menggunakan OODBMS maka perancangan basis data tidak perlu menggunakan ERD. ERD memiliki beberapa aliran notasi seperti notasi chen (Sukamto & Shalahudin, 2018). Berikut adalah simbol-simbol yang digunakan pada ERD dengan notasi chen :

Tabel II. 1

ERD dengan notasi chen (Sukamto & Shalahudin, 2018)

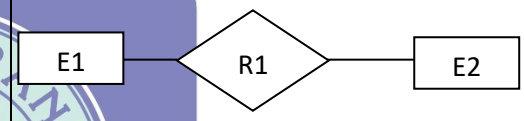
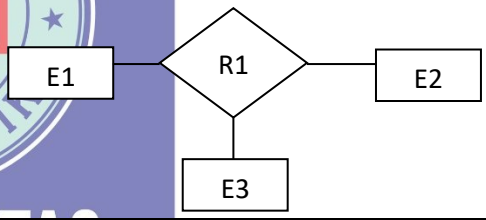
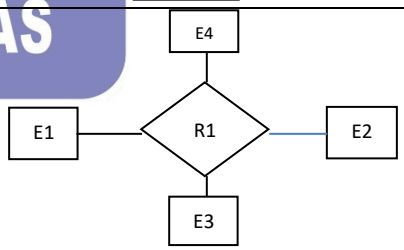
Simbol	Deskripsi
Entitas / *entity* 	Entitas merupakan data inti yang akan disimpan; bakal tabel pada data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer; penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel.
Atribut 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas.
Atribut kunci primer 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses *record* yang diinginkan; biasanya berupa id; kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut bersifat unik.
Atribut multinilai / *multivalue* 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.
Relasi 	Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja.
Asosiasi / *association* N 	Penghubung antara relasi dan entitas di mana di kedua ujunganya memiliki *multiplicity* kemungkinan jumlah pemakaian. Kemungkinan jumlah maksimum keterhubunga antar entitas yan lain disebut dnegan kardinalitas. Misalkan ada kardinalitas 1 ke N atau yang sering di sebut dengan *one to many* menghubungkan entitas A dan entitas B maka ERD biasanya memiliki hubuangan *binary* (satu relasi menghubungkan dua buah entitas). Beberapa metode perancangan ERD menoleransi

	hubungan relasi *ternary*(satu relasi menghubungkan tiga buah relasi) atau N-*ary* (satu relasi menghubungkan banyak entitas), tapi banyak metode perancangan ERD yang tidak mengizinkan hubungan *ternary* atau N-*ary*

Berikut adalah contoh bentuk hubungan relasi dalam ERD :

Tabel II. 2

Contoh ERD (Sukamto & Shalahudin, 2018)

Nama	Gambar
Binary	
Ternary	
N-*ary*	

2.2.2. *Logical Record Structure* (LRS)

"*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 (Hidayat, Marlina, & Utami, 2017).

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

"Kode digunakan untuk tujuan mengklasifikasikan data, memasukkan data ke dalam komputer dan untuk mengambil bermacam-macam informasi yang berhubungan dengannya" (Mustakini, 2016).

Di dalam merancang suatu kode harus diperhatikan beberapa hal, yaitu sebagai berikut:

a. 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.

b. Harus unik.

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

c. Harus fleksibel.

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

d. 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.

e. 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.

f. 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.

g. Spasi dihindari.

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

h. Hindari karakter yang mirip.

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

i. 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 (Mustakini, 2016).

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 (*Hierarcky Input Proses Output*)

"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" (Mustakini, 2016).

HIPO menggunakan tiga macam diagram untuk masing-masing tingkatannya, 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" (Mustakini, 2016). Pada waktu akan menggambar suatu bagan alir, analisis sistem atau pemrogram dapan mengikuti pedoman-pedoman sebaagai berikut ini :

a. Bagan alir sebaiknya digambar dari atas ke bawah dan dimulai dari bagia kiri dari suatu halaman.
b. Kegiatan di dalam bagan alir harus ditunjukan dengan jelas.
c. Harus ditunjukan dari mana kegiatan akan dimulai dan dimana akan berakhir.
d. Masing masing kegiatan di dalam bagan alir sebaiknya digunakan suatu kata yang mewakili suatu pekerjaan, misalnya:
   1). "Persiapkan" dokumen
   2). "Hitung" gaji
e. Masing-masing kegiatan di dalam bagan alir harus di dalam urutan yang semestinya.
f. Kegiatan yang terpotong dan akan disambung ditempt lain harus ditunjukkan dengan jelas menggunakan simbol penghubung.
g. Gunakanlah simbol-simbol bagan alir yang standar.

2. Bentuk *Flowchart*

a. Program *Flowchart*

"Bagan alir program (*program flowchart*) merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program. Bagan alir program dibuat dari derivikasi bagan alir sistem" (Mustakini, 2016).

b. Sistem *Flowchart*

"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" (Mustakini, 2016).

c. Teknik Pembuatan

Adapun teknik pembuatan program flowchart ini dibagi menjadi dua bagian (Yulia, 2017) 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-DirectLoop*).

2). *Iteration Way*

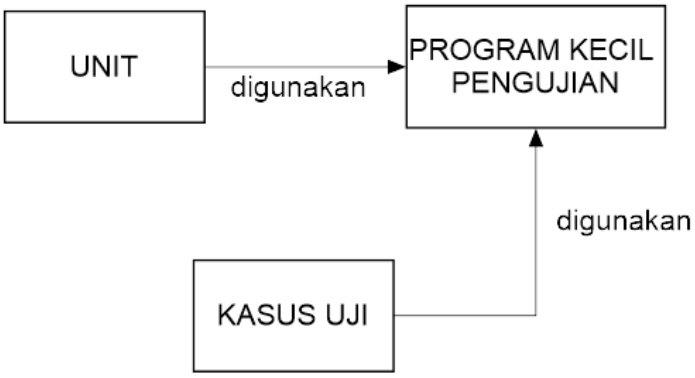
Teknik pembuatan flowchart dengan cara ini biasanya dipakai untuk logika program yang cepat serta bentuk permasalahan yang kompleks. Dimana pengulangan proses yang terjadibersifat langsung (*Direct-Loop*)

2.2.6. Implementasi dan Pengujian Unit

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 (Sukamto & Shalahudin, 2018).

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 :



Sumber : (Sukamto & Shalahudin, 2018)

Gambar II.2

Pengujian Unit

Setiap unit diuji menggunakan sebuah program pengujian yang khusus dibuat untuk menguji sebuah unit menggunakan kumpulan kasus uji yang didefinisikan. Pengujian untuk validasi mengunakan *black-box testing*. *Black-Box Testing* (pengujian kotak hitam) yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program.

Pengujian di maksudkan 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 :

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

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