BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

2.1.1. Konsep Dasar Sistem Informasi

Menurut Sutabri (2014:42) "Sistem Informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan".

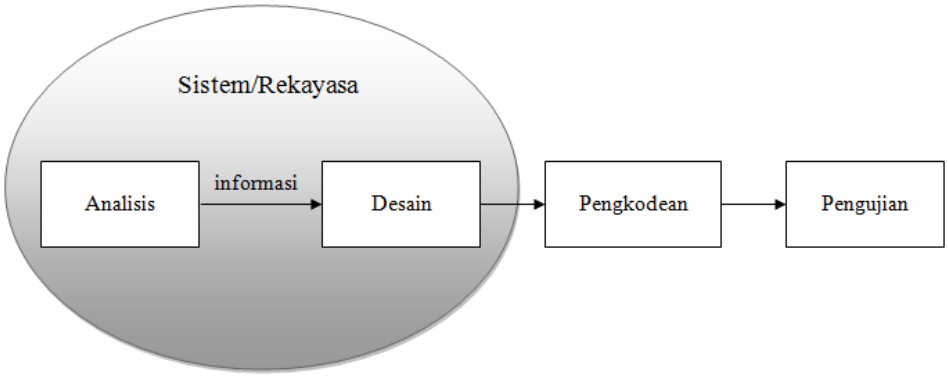
2.1.2. *Inventory*

Menurut Heryanto (2014:33) "*Inventory* atau yang sering disebut persediaan merupakan simpanan barang-barang mentah, material atau barang jadi yang disimpan untuk digunakan dalam masa mendatang atau dalam kurun waktu tertentu. Pengertian dalam definisi lainnya adalah suatu teknik untuk manajemen material yang berkaitan dengan persediaan".

2.1.3. Model Pengembangan Sistem

Menurut Sukamto dan Shalahuddin (2015:31)" Model *waterfall* adalah model SDLC yang paling sederhana. Model ini hanya cocok untuk pengembangan perangkat lunak dengan spesifikasi yang tidak berubah-ubah".

Model SDLC air terjun (*waterfall*) sering juga disebut model sekuensial linier atau (*sequential linier*) alur hidup klasik (*classic life cycle*). Model air terjun menyediakan pendekatan alur hidup perangkat lunak secara sekuensial atau terurut mulai dari analis, desain, pengkodean, pengujian dan tahap pendukung (*support*). Berikut adalah gambar model *waterfall* :



Sumber : Sukamto dan Shalahuddin (2015 : 29)

Gambar II.1.
Ilustrasi model *waterfall.*

Pembuatan model *waterfall* tentunya tidak terlepas dari tahapan-tahapan yang harus dikerjakan secara terstruktur. Penjelasan tahapan-tahapan pembuatan model *waterfall* adalah sebagai berikut:

1. Analisa Kebutuhan Perangkat Lunak

   Proses pengumpulan kebutuhan dilakukan secara intensif untuk mespesifikasikan kebutuhan perangkat lunak agar dapat dipahami perangkat lunak seperti apa yang dibutuhkan *user*. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu 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 represetasi 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 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 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*) atau pemeliharaan (*maintenance*)

   Tidak menutup kemungkinan sebuah perangkat lunak mengalami perubahan ketika sudah dikirimkan ke *user*.

Model *waterfall* sangat cocok digunakan untuk kebutuhan selama pengembang perangkat lunak kecil. Hal positif dari model *waterfall* adalah struktur tahap pengembangan sistem jelas, dokumentasi dihasilkan di setiap tahap pengembangan, dan sebuah tahap dijalankan setelah tahap sebelumnya selesai dijalankan (tidak ada tumpang tindih pelaksanaan tahap).

Dari kenyataan yang terjadi sangat jarang model *waterfall* dapat dilakukan sesuai alurnya karena sebab berikut :

1. Perubahan spesifikasi perangkat lunak terjadi ditengah pengembangan.
2. Sangat sulit bagi pelanggan untuk mendefinisikan semua spesifikasi di awal alur pengembangan. Pelanggan sering kali butuh contoh (*prototype*) untuk menjabarkan spesifikasi kebutuhan sistem lebih lanjut.
3. Pelanggan tidak mungkin bersabar mengakomodasi perubahan yang diperlukan di akhir alur pengembangan.

2.1.4.Program Terstruktur

Menurut Sukamto dan Shalahuddin (2015:67) "Pemograman terstruktur adalah konsep atau paradigma sudut pandang pemograman yang membagi-bagi program berdasarkan fungsi-fungsi atau prosedur-prosedur yang dibutuhkan program komputer".

2.1.5. PHP

Menurut Madcoms (2008:35) "PHP adalah salah satu bahasa program yang berjalan dalam sebuah *web server* dan berfungsi sebagai pengolah data pada sebuah *server*".

PHP merupakan bahasa *scripting* yang menyediakan cara yang mudah untuk melekatkan program pada halaman *web*. Suatu halaman akan diproses terlebih dahulu oleh PHP sebelum dikirim ke *client*, sehingga *script* dapat menghasilkan isi halaman yang dinamis, seperti minsalnya menampilkan hasil *query* dari MySQL pada halaman tersebut. PHP pada mulanya berarti *Personal Home Page*,

tetapi sekarang berubah menjadi PHP *Hypertext Preprocessor*. Dibawah ini merupakan salah satu contoh penulisan tag pembuka dan tag penutup.

```
HTML
<?php
Kode PHP disini;
?>
HTML
```

Dan

```
HTML
<script language=”php”>
Kode PHP disini;
</script>
HTML
```

2.1.6. *JavaScript*

Menurut Sianipar (2017:1) “*JavaScript* adalah sebuah bahasa *script* dinamis yang dapat dipakai untuk membuat halaman-halaman HTML statis lebih interaktif”.

Proses penulisan *javascript* dilakukan dengan menanamkan blok-blok kode di hampir semua tempat pada halaman *web* anda. Untuk melakukannya, blok kode *javascript* diawali dan diakhiri dengan tag *script* :

```
<script type=”text/javascript”>
Kode javascript ditempatkan disini;
</script>
```

Tag *script* memiliki satu atribut penting : *language*. Atribut ini menetapkan bahasa *script* yang sedang digunakan. Umumnya nilainya adalah *javascript* atau *javascript* 1.0, *javascript* 1.1, *javascript* 1.2 dan seterusnya. Dengan menetapkan nomor versi *javascript* spesifik, anda mengindikasikan kepada *browser* bahwa *script* ini hanya dapat dijalankan pada sebuah *browser* yang mendukung versi *javascript* sesuai dengan yang ditetapkan.

2.1.7. CSS

CSS adalah kependekan dari *Cascading Style Sheet*. CSS merupakan salah satu kode pemograman yang bertujuan untuk menghias dan mengatur gaya tampilan (*layout*) halaman *web*, supaya lebih elegan dan menarik. *Cascading Style Sheet* terdiri dari selektor, deklarasi, properti dan nilai. Seperti pada HTML, PHP dan bahasa pemrograman lainnya, CSS juga memiliki aturan sendiri. Contoh penulisan kode css :

```
Body {background-color: white;}
```

Body adalah Selektor, { } adalah deklarasi, *background-color* adalah properti dan *white* adalah nilai. Maksud dari kode diatas adalah mengatur warna latar belakang (*background color*) dari tag *Body* sebuah halaman web.

CSS digunakan oleh *web programmer* dan juga *blogger* untuk menentukan warna, tata letak *font*, dan semua aspek lain dari presentasi dokumen di situs mereka. Saat ini, hampir tidak ada situs *web* yang dibangun tanpa kode css.

2.1.8. MySQL

Menurut Raharjo (2015:16) "MySQL merupakan *software* RDBMS (atau *server database*) yang dapat mengolah *database* dengan sangat cepat, dapat menampung data dalam jumlah sangat besar, dapat diakses oleh banyak *user* (*multi-user*), dan dapat melakukan suatu proses secara sinkron atau berbarengan (*multi-threated*)".

MySQL adalah sebuah implementasi dari sistem manajemen basisdata relasional (RDBMS) yang didistribusikan secara gratis di bawah lisensi GPL (*General Public License*). Setiap pengguna dapat secara bebas menggunakan MySQL, namun dengan batasan perangkat lunak tersebut tidak boleh dijadikan produk turunan yang bersifat komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam basisdata yang telah ada sebelumnya; SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian basisdata, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

2.1.9. IIS Server

IIS (*Internet Information Service*) adalah sebuah HTTP *web server* seperti *apache* yang digunakan dalam sistem operasi *windows*. Layanan ini berfungsi sebagai pendukung protokol TCP/IP yang berjalan dalam lapisan aplikasi

(*aplication layer*). IIS juga menjadi fondasi *platform internet* dan *intranet microsoft*, yang mencakup *microsoft site server*, *microsoft commercial internet system* dan produk *microsoft back office* lainnya.

2.1.10. Black Box Testing

Black box testing adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak. Jadi dianalogikan seperti kita melihat suatu koatak hitam, kita hanya bisa melihat penampilan luarnya saja, tanpa tau ada apa dibalik bungkus hitam nya. Sama seperti pengujian *black box*, mengevaluasi hanya dari tampilan luarnya(*interface* nya) , fungsionalitasnya.tanpa mengetahui apa sesungguhnya yang terjadi dalam proses detilnya (hanya mengetahui *input* dan *output*).

2.1.11. Peralatan Pendukung

Merupakan suatu alat yang digunakan untuk menggambarkan bentuk logika model dari suatu sistem dengan menggunakan simbol-simbol, lambang-lambang, diagram-diagram yang menunjukkan secara tepat arti dan fungsinya. Dalam hal ini penulis menggunakan *tools system* menggunakan UML (*Unifield Modeling Language*) dan ERD (*Entity Relationship Diagram*).

1. UML (*Unifield Modeling Language*)

Menurut Sukamto dan Shalahuddin (2015:133) menerangkan bahwa “UML (*Unified Modeling Language*) adalah salah satu standar bahasa yang banyak digunakan di dunia industry untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemograman berorientasi objek”.

Pada UML (*Unifield Modeling Language*) terdiri dari beberapa macam diagram yaitu :

a. *Use Case Diagram*

Use case atau diagram *use case* merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan 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 di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1) Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2) *Use case* merupakan fungsionalitas yang di sediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

b. *Activity* Diagram

Diagram aktivitas atau *activity* diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1) Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
2) Urutan atau pengelompokan tampilan dari sistem/user interface di mana setiap aktivitas dianggap memiliki sebuah rancangan antar muka tampilan.
3) Rancangan pangujian di mana setiap aktivitas dianggap memerlukan sebuah pegujian yang perlu didefinisikan kasus ujinya.

c. *Component* diagram

Diagram komponen atau *component* diagram dibuat untuk menunjukkan organisasi dari ketergantungan di antara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem. Diagram komponen juga dapat digunakan untuk memodelkan hal-hal berikut :

1) *Source code* program perangkat lunak
2) Komponen *executable* yang dilepas ke user
3) Sistem yang harus beradaptasi dengan sistem lain
4) *Framework* sistem, *framework* pada perangkat lunak merupakan kerangka kerja yang dibuat untuk memudahkan pengembangan dan pemeliharaan aplikasi.

d. *Deployment* Diagram

Diagram *deployment* atau *deployment* diagram menunjukan konfigurasi komponen dalam proses eksekusi. Diagram *deployment* juga dapat digunakan untuk memodelkan hal-hal berikut :

1) Sistem tambahan (*embedded system*) yang menggambarkan rancangan *device*, *node*, dan *hardware*.
2) Sistem *client*/server
3) Sistem terdistribusi murni
4) Rekayasa ulang aplikasi

2. ERD (*Entity Relationship Diagram*)

Menurut Fathansyah (2012:75) sesuai namanya, ada 2 komponen utama pembentuk model *entity-relationship*, yaitu entitas (*entity*) dan relasi (*relation*). Kedua komponen ini dideskripsikan lebih jauh melalui sejumlah atribut/properti.

a. Entitas (*Entity*) dan Himpunan Entitas (*Entity Set*)

Entitas merupakan individu yang mewakili sesuatu yang nyata (eksistensinya) dan dapat dibedakan dari sesuatu yang lain. Sebuah kursi yang kita duduki, seseorang yang menjadi pegawai di sebuah perusahaan dan sebuah mobil yang melintas di depan kita adalah Entitas. Sekelompok entitas yang sejenis dan berada dalam lingkup yang sama membentuk sebuah himpunan entitas (*Entity Set*), Sederhananya, Entitas menunjuk pada individu suatu objek, sedangkan himpunan entitas menunjuk pada rumpun (famili) dari individu tersebut. Seseorang memang dapat menjadi sebuah entitas, tapi dapat berada pada himpunan entitas yang berbeda dengan seseorang yang lain.

b. Atribut (*Attributes Properties*)

Setiap entitas pasti memiliki Atribut yang mendeskripsikan karakteristik (properti) dari Entitas tersebut. Sebagaimana telah disebutkan sebelumnya, penentuan atribut-atribut yang relevan bagi sebuah entitas merupakan hal penting lainnya dalam pembentukan model data. Penetapan atribut bagi sebuah entitas umumnya memang didasarkan pada realita yang ada. Tetapi tidak selalu seperti itu. Kelak akan kita lihat, karena proses normalisasi atau pertimbangan-pertimbangan tertentu, ada sejumlah atribut yang tidak ada di „dunia nyata" tapi perlu kita tambahkan.

c. Relasi (*Relationship*) dan Himpunan Relasi (*Relationship Sets*)

Relasi menunjukkan adanya hubungan di antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Kumpulan semua relasi di antara entitas-entitas yang terdapat pada himpunan entitas-himpunan entitas tersebut membentuk himpunan relasi (*Relationship Sets*). Sebagaimana istilah Himpunan Entitas yang banyak sekali disingkat menjadi Entitas (walaupun sebenarnya memiliki perbedaan makna), istilah Himpunan Relasi jarang sekali digunakan dan lebih sering disingkat dengan istilah Relasi saja.

d. Kardinalitas/Derajat Relasi

Kardinalitas relasi menunnjukkan jumlah maksimum entitas yang dapat berelasi dengan entitas pada himpunan entitas yang lain. Kardinalitas Relasi yang terjadi di antara dua himpunan entitas (misalnya A dan B) dapat berupa:

1) Satu ke satu (*one to one*), yang berarti setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada

himpunan entitas B, dan begitu juga sebaliknya setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

2) Satu ke banyak (*one to many*), yang berarti setiap entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, di mana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

3) Banyak ke satu (*many to one*), yang berarti setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, tetapi tidak sebaliknya, di mana setiap entitas pada himpunan entitas A berhubungan dengan paling banyak satu entitas pada himpunan entitas B.

4) Banyak ke Banyak (*many to many*), yang berati setiap entitas pada himpunan entitas B, dan demmikian juga sebaliknya, di mana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.

3. LRS (*Logical Record Structure*)

> Menurut Iskandar dan Rangkuti (2018:126) "*Logical Record Structue* terdiri dari *link-link* diantara tipe *record*. *Link* ini menunjukan 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.

Logical Record Structure dibentuk dengan nomor dari tipe *record* dan beberapa tipe *record* yang digambarkan oleh kotak empat persegi panjang dan dengan nama yang unik. Perbedaan LRS dengan diagram *entity relation diagram*

nama tipe *record* berada diluar kotak *field* tipe *record* ditempatkan. Dua metode yang dapat digunakan, dimulai dengan hubungan kedua model yang dapat dikonversikan ke LRS. Metode yang lain dimulai dengan *entity relationship diagram* dan langsung dikonversikan ke LRS.

a. Konversi ERD ke LRS, Diagram *entity relationship diagram* harus diubah kebentuk LRS. Dari bentuk LRS inilah yang nantinya dapat di transformasikan ke bentuk relasi (tabel).

b. Konversi ERD ke LRS Sebuah model sistem yang digambarkan dengan sebuah ERD akan mengikuti pola pemodelan tertentu. Dalam kaitannya dengan konversi ke LRS, untuk perubahan yang terjadi adalah mengikuti aturan-aturan berikut:

   1) Setiap entitas diubah ke bentuk kotak dengan nama entitas, berada diluar kotak dan atribut berada didalam kotak.
   2) Sebuah *relationship* kadang disatukan dalam sebuah kotak bersama entitas, kadang sebuah kotak bersama-sama dengan entitas, kadang disatukan dalam sebuah kotak tersendiri.

b. Konversi LRS ke relasi atau table adalah bentuk pernyataan data secara grafis dua dimensi, yang terdiri dari kolom dan baris. Relasi adalah bentuk visual dari sebuah *file*, dan tipe tuple dalam sebuah *field*, atau yang dalam bentuk lingkaran diagram *entity relationship* dikenal denmgan sebutan atribut. konversi dari *logical record structure* dilakukan dengan cara:

   1) Nama *logical record structure* menjadi nama relasi.
   2) Tiap atribut menjadi sebuah kolom didalam relasi.

2.2 Penelitian Terkait

Penelitian terkait ini bertujuan untuk membahas konsep dasar yang ingin dikembangkan. Konsep dasar model pengembangan sistem serta jurnal yang menjadi acuan dari penulis yang disusun secara sistematis yang diharapkan dapat membantu penulis dalam menyelesaikan skripsi.

> Menurut Kustanto dan Yulis Wahyu Kritanto (2014:78) mengemukakan bahwa : "Harrisma Bengawan merupakan toko komputer yang memiliki beberapa cabang di kawasan wilayah Solo raya. Berkaitan dengan manajemen proses pemesanan atau mengetahui stok barang disetiap cabang membutuhkan waktu yang lumayan lama. Hal tersebut dikarenakan belum terbangunnya sistem *inventory* data secara *online*. Dalam pembuatan website ini menggunakan metode *waterfall* agar mempermudah soerang pragrammer dalam pembuatan sistem berbasis web tersebut. Maka dapat disimpulkan bahwa pengguanaan sistem yang terkomputerisasi akan lebih banyak menghemat waktu dan menghasilkan keakuratan penyajian data".

> Menurut Agusvianto (2017:40) mengemukakan bahwa : "Pengolahan data gudang pada PT. Alaisys sampai saat ini masih manual seperti pencatatan informasi pada penjualan dan pesediaan barang barang dengan menggunakan bon nota, buku pencatatan dan laporan yang semua masih ditulis tangan. Untuk mendapatkan data yang dibutuhkan harus mencari satu persatu nota yang telah disimpan. Dalam metode pengembangan sistem yang digunakan dalam pembuatan website ini adalah SDLC (*system development life cycle*). Tujuan menggunakan metode waterfall ini untuk mempermudah programmer dalam pembuatan sistem berbasis web tersebut. Maka dapat disimpulkan bahwa aplikaksi sistem *inventory* berbasis web dapat menyediakan informasi yang cepat, tepat dan akurat dalam menangani data maupun laporan serta mempermudah proses penyampaian laporan informasi ke kantor pusat".