BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

2.1.1. Konsep Dasar Sistem Informasi

> Menurut Kadir (2014 : 8), "menyatakan bahwa sistem informasi mencakup sejumlah komponen (manusia, komputer, taknologi informasi, dan prosedur kerja), ada sesuatu yang di proses (data menjadi informasi), dan dimaksudkan untuk mencapai suatu sasaran atau tujuan."

1. Karakteristik Sistem

Menurut (Dudung, 2015) Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu, yang memiliki komponen (*component*), batas sistem (*boundary*), lingkungan luar sistem (environment), penghubung sistem (*interface*), masukan sistem (*input*), keluaran sistem (*output*), pengolah sistem (*proses*), dan sasaran sistem (*objectives*), karakteristik yang dimasuk ada sebagai berikut :

a. Komponen (*Component*)

   Sebuah sistem yang terdiri dari sejumlah komponen yang saling berinteraksi, yang berarti bahwa setiap bakerja bersama untuk membentuk serikat pekerja. Komponen sistem biasanya dikenal dengan subsistem. Subsistem memiliki hal sistem itu sendiri dalam fungsinya dan memiliki sistem keseluruhan.

b. Batas Sistem (*Boundary*)

   Pembatasan yang membatasi sistem merupakan daerah antara sistem dengan sistem lainnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan. Menunjukkan sistem membatasi ruang lingkup sistem.

c. Lingkungan Luar Sistem (*Environment*)

Lingkungan luar sistem di luar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar dapat 6 bermanfaat serta merugikan sistem. Lingkungan eksternal yang menguntungkan merupakan energi dari sistem dan lingkungan luar yang merugikan harus ditahan dan dikendalikan, kalau tidak akan mengganggu kehidupan kelansungan sistem.

d. Penghubung Sistem (*Interface*)

Sistem link adalah media penghubung antara subsistem lainnya. Melalui *interface* ini memungkinkan sumber daya mengalir dari satu subsistem ke subsistem lainnya.

e. Masukan Sistem (*Input*)

Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat pengobatan masukan (input pemeliharaan) dan sinyal *input* (sinyal *input*). Masukan energi pemeliharaan dimasukkan sehingga sistem tersebut dapat beroperasi. Sinyal *input* diproses untuk mendapatkan keluaran energi.

f. Keluaran Sistem (*Output*)

Keluaran sistem adalah hasil dari energi dalam meskipun dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan. *Output* dapat menjadi masukan bagi subsistem lain atau suprasistem.

g. Pengolah Sistem (Proses)

Suatu sistem dapat memiliki bagian pengolahan yang akan mengubah input menjadi *output*.

h. Sasaran Sistem (*Objectives*)

Sebuah sasaran yang ingin dicapai untuk menentukan masukan yang diperlukan dari *output* sistem menjadi sistem yang dihasilkan.

2.1.2. Pengertian Persediaan

Menurut Sari dan Dahria (2010 : 1) "Persediaan (*inventory*) adalah meliputi semua barang yang dimiliki perusahaan pada saat tertentu, dengan tujuan untuk dijual atau dikonsumsi dalam siklus operasi normal perusahaan. Aktiva lain yang dimiliki perusahaan, tetapi tidak untuk dijual atau dikonsumsi tidak termasuk dalam klasifikasi persediaan."

Persediaan (*inventori*) untuk setiap perusahaan akan berbeda, tergantung kepada jenis perusahaan yang bersangkutan :

1. Pada perusahaan dagang, berupa persediaan barang dagangan (*merchandise inventory*)
2. Sementara pada perusahaan pabrik (manufaktur) persediaan terdiri atas :
   a. Persediaan bahan baku (*Direct materials inventory*)
   b. Persediaan barang dalam proses (*Work in proses inventory*)
   c. Persediaan barang jadi (Finished goods inventory)

2.1.3. Metode Pengembangan Sistem

Pada prinsipnya permodelan sistem *waterfall* perkembangannya dilakukan secara sistematis dan terarah secara berurutan melalui dari tahap sistem, tahap analisa, tahap desain sistem, *coding*, *testing*, *support*, dan dapat kembali ke tahap awal apabila semua tahapan pengembangan sistem telah dilalui. Model *waterfall*

adalah model SDLC yang paling sederhana. Model ini hanya cocok untuk pengembangan perangkat lunak dengan spesifikasi yang tidak berubah-ubah.

Menurut Rosa dan Shalahudin (2015 : 25) "*SDLC atau Software Development Life Cycle* atau yang sering disebut juga *System Development Life Cycle* adalah proses mengembangkan atau mengubah suatu sistem perangkat lunak dengan menggunakan model-model dan metodologi yang digunakan orang untuk mengembangkan sistem-sistem perangkat lunak sebelumnya (berdasarkan best practice atau cara-cara yang sudah teruji baik)".

Adapun pengembangan sistem yang penulis lakukan adalah dengan menggunakan model *cllasic life cycle /* model *water fall* dengan tahapan – tahapan sebagai berikut (Rosa & shalahudin, 2015:28) :

1. Rekayasa Perangkat Lunak (*Software Engineering*)

   Merupakan kegiatan untuk menentukan kebutuhan perangkat lunak apa yang dibutuhkan oleh pengguna.

2. Desain

   Merupakan perancangan perangkat lunak yang dilakukan berdasarkan data-data yang telah dikumpulkan pada tahap sebelumnya. Desain perangkat lunak adalah proses multilangkah yang focus pada desain pembuatan program perangkat lunak termasuk struktur data, arsitektur perangkat lunak, representasi antarmuka, dan prosedur pengkodean. Tahap ini mentranslasi kebutuhan perangkat lunak dari tahap analisis kebutuhan ke representasi desain agar dapat diimplementasikan menjadi program pada tahap selanjutnya.

3. Penulisan Kode Program

   Kegiatan yang mengimplementasikan hasil dari perancangan perangkat lunak kedalam kode program yang dimengerti oleh bahasa mesin.

4. Pengujian (*Testing*)

   Memfocuskan pada logika internal dari perangkat lunak, fungsi eksternal, mencari dari segi kemungkinan, dan memeriksa apakah hasil output sudah sesuai dengan hasil yang diharapkan setelah proses.

5. *Support*

   Bertujuan untuk menjaga agar sistem tetap berjalan dengan produktif dan sisitem dapat memiliki daya tahan bertahun – tahun, dimana sistem digunakan, dipelihara, dan dikembangkan terus menerus untuk mencapai keuntungan yang diinginkan.

Dalam pengembangan sistem perangkat lunak yang penulis pakai adalah metode *water fall* yang tentunya juga mempunyai beberapa keunggulan dan kelemahan, yaitu:

Keunggulan Metode *Water fall* :

1. *software* yang dikembangkan dengan metode ini biasanya menghasilkan kualitas yang baik.
2. Dokumen pengembangan sistem sangat terorganisir, karena setiap fase harus terselesaikan dengan lengkap sebelum melangkah ke fase berikutnya.

Kekurangan Metode *Water fall* :

1. Membutuhkan keahlian yang baik atau yang telah berpengalaman dalam mengembangkan perangkat lunak, dalam arti metode ini kurang cocok bagi pemula.
2. Diperlukan manajemen yang baik, karena proses pengembangan tidak dapat berulang sebelum menghasilkan suatu produk, yaitu aplikasi. Apabila dalam

suatu proses seperti perancangan tidak selesai tepat waktu, maka akan mempengaruhi keseluruhan proses pengembangan perangkat lunak.

3. Iterasi sering terjadi menyebabkan masalah baru.
4. *Client* kesulitan untuk menyatakan semua keinginannya secara eksplisit diawal tahap pengembangan.
5. Hasil software yang dikembangkan baru akan diketahui lama setelah proyek pengembangan dimulai.

2.1.4. Konsep Dasar Program

Menurut (Huda, 2017), "dalam dunia pemrograman, biasanya dihadapkan pada dua jenis metode pemrograman, yaitu pemrograman procedural (*procedural*) dan pemrograman berorientasi object (*object oriented*)."

Pemrograman prosedural merupakan suatu metode menulis program yang didasarkan pada "serangkaian tugas yang diselesaikan dalam bentuk fungsi atau prosedur". Cara pandang pemrograman prosedural yaitu sebuah *program* adalah suatu urutan instruksi. Programmer harus mem-*break down* suatu *problem/*masalah menjadi sub *problem* yang lebih sederhana. Fokus utama metode procedural ini adalah fungsi dan prosedur, dimana keduanya digunakan untuk memanipulasi data. Dalam hal ini data bersifat pasif.

Lain halnya dengan pemrograman berorientasi objek (OOP), fungsi dan data bukan menjadi hal yang terpisah. Fungsi dan data menjadi satu kesatuan yang disebut sebagai objek aktif. Cara pandang OOP ini yaitu sebuah program merupakan serangkaian objek yang bekerjasama untuk menyelesaikan suatu *problem.*

Dengan kata lain, metode prosedural berfokus pada cara komputer menangani tugas, sedangkan metode OOP berfokus pada tuhas yang kita kembangkan untuk dieksekusi komputer. Kedua jenis metode pemrograman tersebut dapat digunakan untuk menangani masalah yang sama, asalkan Bahasa pemrograman yang digunakan mendukung metode-metode tersebut. Contoh Bahasa pemgrograman yang mendukung OOP diantaranya : Java, C++, Pascal, Visual Basic .NET, Ruby, Python, PHP, C#, Delphi, Perl, dsb. Selain itu masih banyak Bahasa lain yang termasuk ke dalam Bahasa procedural, atau bahkan bisa mendukung kedua metode pemrograman tersebut.

A. Istilah-istilah OOP

1. Objek

   Untuk mempermudah pemahaman, maka disini akan dijelaskan melalui analogi. Pada dasarnya semua benda yang adadi dunia nyata dapat dianggap sebagai objek. Misalnya rumah, mobil, sepeda, motor, gelas, komputer, meja, sepatu, dll. Setiap objek memiliki atribut sebagai status (state) dan tingkah laku sebagai behavior.

   Contoh objek : Motor maka atribute (*state*) nya adalah pedal, roda, jeruji, *speedometer*, warna, jumlah roda. Sedangkan tingkah laku (*behavior*) nya adalah kecepatan menaik, kecepatan menurun, dan perpindahan gigi motor.

   Analogi pemrograman berorientasi objek sama dengan penggambara pada dunia nyata seperti contoh di atas. Dalam OOP, state disimpan pada variabel dan tingkah laku disimpan pada *method*.

Dalam bahasa teoretis OOP, Objek berfungsi untuk membungkus data dan fungsi bersama menjadi satu unit dalam sebuah program komputer. Objek merupakan dasar dari modularitas dan struktur dalam sebuah program komputer berorientasi objek.

2. Class

   Definisi *class* yaitu *template* untuk membuat objek. *Class* merupakan prototipe atau *blue prints* yang mendefinisikan variabel-variabel dan *method-method* secara umum. Objek merupakan hasil instansiasi dari suatu class. Proses pembentukan objek dari suatu kelas disebut sebagai *instantiation*. Objek disebut juga sebagai *instances*.

   Dalam bahasa teoretis OOP, class merupakan kumpulan atas definisi data dan fungsi-fungsi dalam suatu unit untuk suatu tujuan tertentu. Sebagai contoh '*class of dog*' adalah suatu unit yang terdiri atas definisi-definisi data dan fungsi-fungsi yang menunjuk pada berbagai macam perilaku/turunan dari anjing. Sebuah *class* adalah dasar dari modularitas dan struktur dalam pemrograman berorientasi objek.

   Sebuah *class* secara tipikal sebaiknya dapat dikenali oleh seorang non-programmer sekalipun terkait dengan domain permasalahan yang ada, dan kode yang terdapat dalam sebuah class sebaiknya (relatif) bersifat mandiri dan independen (sebagaimana kode tersebut digunakan jika tidak menggunakan OOP). Dengan modularitas, struktur dari sebuah program akan terkait dengan aspek-aspek dalam masalah yang akan diselesaikan melalui program tersebut. Cara seperti ini akan menyederhanakan pemetaan dari masalah ke sebuah program ataupun sebaliknya.

3. *Attributes*

   Atribut adalah data yang membedakan antara objek satu dengan yang lainnya. Dalam class, atribut sering disebut sebagai variabel. Atribut dibedakan menjadi dua jenis yaitu *Instance Variable* dan *Class Variable*.

   a. *Instance variable* adalah atribut untuk tiap objek dari kelas yang sama. Tiap objek mempunyai dan menyimpan nilai atributnya sendiri. Jadi, tiap objek dari *class* yang sama boleh mempunyai nilai yang sama atau berbeda.
   b. *Class Variable* adalah atribut untuk semua objek yang dibuat dari class yang sama. Semua objek mempunyai nilai atribut yang sama. Jadi semua objek dari *class* yang sama mempunyai hanya satu nilai yang value nya sama.

4. *Behavior*

   Behavior/tingkah laku adalah hal-hal yang bisa dilakukan oleh objek dari suatu class. *Behavior* dapat digunakan untuk mengubah nilai atribut suatu objek, menerima informasi dari objek lain, dan mengirim informasi ke objek lain untuk melakukan suatu tugas (*task*).

   Dalam *class*, *behavior* disebut juga sebagai methods. *Methods* sendiri adalah serangkaian *statements* dalam suatu *class* yang menghandle suatu task tertentu. Cara objek berkomunikasi dengan objek yang lain adalah dengan menggunakan *method*.

5. Abstraksi

   Abstraksi adalah kemampuan sebuah program untuk melewati aspek informasi yang diproses olehnya, yaitu kemampuan untuk memfokus pada inti. Setiap objek dalam sistem melayani sebagai model dari “pelaku” abstrak yang dapat melakukan kerja, laporan dan perubahan keadaannya, dan berkomunikasi

dengan objek lainnya dalam sistem, tanpa mengungkapkan bagaimana kelebihan ini diterapkan. Proses, fungsi atau metode dapat juga dibuat abstrak, dan beberapa teknik digunakan untuk mengembangkan sebuah pengabstrakan.

B. Konsep-Konsep OOP

1. Enkapsulasi (*Encapsulation*)

   Definisi enkapsulasi: Pembungkusan variabel dan *method* dalam sebuah obyek yang terlindungi serta menyediakan interface untuk mengakses variabel tersebut. Variabel dan *method* yang dimiliki oleh suatu objek, bisa ditentukan hak aksesnya. Dalam OOP, konsep enkapsulasi sebenarnya merupakan perluasan dari struktur dalam bahasa C.

   Contoh: jam tangan. Dalam hal ini, penting sekali untuk mengetahui waktu, sedangkan cara jam mencatat waktu dengan baik antara jam bertenaga baterai atau bertenaga gerak tidaklah penting kita ketahui.

   Dengan kata lain enkapsulasi berfungsi untuk memastikan pengguna sebuah objek tidak dapat mengganti keadaan dalam/dari sebuah objek dengan cara yang tidak layak; hanya metode dalam objek tersebut yang diberi izin untuk mengakses keadaannya. Setiap objek mengakses interface yang menyebutkan bagaimana objek lainnya dapat berinteraksi dengannya. Objek lainnya tidak akan mengetahui dan tergantung kepada representasi dalam objek tersebut.

2. Pewarisan (*Inheritance*)

   Pewarisan merupakan pewarisan atribut dan *method* dari sebuah *class* ke *class* lainnya. *Class* yang mewarisi disebut *superclass* dan *Class* yang diwarisi

disebut *subclass*. *Subclass* bisa berlaku sebagai superclass bagi *class* lainya, disebut sebagai *multilevel inheritance*.

Contoh : terdapat *class* sepeda dan sepeda gunung. Sepeda termasuk *superclass*. Sepeda gunung termasuk subclass. Hal ini dikarenakan sepeda gunung memiliki variabel dan *method* yang dimiliki oleh sepeda.

Prinsip dasar *inheritance* yaitu persamaan-persamaan yang dimiliki oleh beberapa kelas dapat digabungkan dalam sebuah class induk sehingga setiap kelas yang diturunkannya memuat hal-hal yang spesifik untuk kelas yang bersangkutan.

Keuntungan Pewarisan :

a. *Subclass* menyediakan *state/behaviour* yang spesifik yang membedakan dengan *superclass*, sehingga memungkinkan *programmer* untuk menggunakan ulang *source code* dari *superclass* yang telah ada.
b. *Programmer* dapat mendefinisikan superclass khusus yang bersifat generik, yang disebut *abstract class* (abstraksi), untuk mendefinisikan *class* dengan tingkah laku dan *state* secara umum.

2.1.5. *Unified Modelling Language* (UML)

Menurut (Ritonga, 2015), "*Unified Modeling Language* (UML) adalah tujuan umum, perkembangan, bahasa pemodelan di bidang rekayasa perangkat lunak, yang dimaksudkan untuk menyediakan cara standar untuk memvisualisasikan desain sistem."

UML awalnya termotivasi oleh keinginan untuk membakukan sistem notasi yang berbeda dan pendekatan untuk desain perangkat lunak yang dikembangkan

oleh Grady Booch, Ivar Jacobson dan James Rumbaugh di *Rational Software* di 1994-1995, dengan pengembangan lebih lanjut yang dipimpin oleh mereka melalui tahun 1996.

Pada tahun 1997 UML diadopsi sebagai standar oleh *Object Management Group* (OMG) dan telah dikelola oleh organisasi ini. Pada tahun 2005 UML juga diterbitkan oleh *International Organization for Standardization* (ISO) sebagai standar ISO disetujui. Sejak itu telah periodic direvisi untuk menutupi revisi terbaru dari UML.

> Menurut Widodo (2011:10), "Beberapa *literature* menyebutkan bahwa UML menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa diagram yang digabung, misalnya diagram komunikasi, diagram urutan dan diagram pewaktuan digabung menjadi diagram iteraksi".

Namun demikian model-model itu dapat dikelompokkan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain :

1. Diagram Kelas (*Class Diagram*)

   Bersifat statis, diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas aktif.

2. Diagram Paket (*Package Diagram*)

   Bersifat statis, diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen.

3. Diagram *Use-Case* (*Use-Case Diagram*)

   Bersifat statis, diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk

mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.

4. Diagram Interaksi dan *Sequence* (*Sequence Diagram*)

   Bersifat dinamis, diagram urutan adalah iterasiksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.

5. Diagram Komunikasi (*Communication Diagram*)

   Bersifat dinamis, diagram sebagai pengganti diagram kolaborasi UML yang menekankan organisasi struktural dari objek-objek yang menerima serta mengirim pesan.

6. Diagram *Statechart* (*Statechart Diagram*)

   Bersifat dinamis, diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*state*), transisi, kejadian serta aktivitas.

7. Diagram Aktivitas (*Activity Diagram*)

   Bersifat dinamis, diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas ke aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan memberi tekanan pada aliran kendali antar objek.

8. Diagram Komponen (*Component Diagram*)

   Bersifat statis, diagram komponen ini memperlihatkan organisasi serta kebergantungan sistem atau perangkat lunak pada komponen-komponen yang telah ada sebelumnya.

9. Diagram Deployment (*Deployment Diagram*)

   Bersifat statis, diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*), memuat simpul-simpul beserta komponen-komponen yang

didalamnya. Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai kebutuhan. Pada UML dimungkinkan kita menggunakan diagram-diagram lainnya misalnya data *flow diagram, entity relationship diagram*, dan sebagainya.

2.1.6. *Entity Relationship Diagram* (ERD)

> Menurut salah satu para ahli, Brady dan Loonam (2010), "*Entity Relationship diagram* (ERD) merupakan teknik yang digunakan untuk memodelkan kebutuhan data dari suatu organisasi, biasanya oleh *System Analyst* dalam tahap analisis persyaratan proyek pengembangan system. Sementara seolah-olah teknik diagram atau alat peraga memberikan dasar untuk desain database relasional yang mendasari sistem informasi yang dikembangkan."

ERD bersama-sama dengan detail pendukung merupakan model data yang pada gilirannya digunakan sebagai spesifikasi untuk database. Entitas adalah objek dalam dunia nyata yang dapat dibedakan dengan objek lain, sebagai contoh mahasiswa, dosen, departemen. Entitias terdiri atas beberapa Atribut sebagai contoh Atribut dari entitas mahasiswa adalah nim, nama, alamat, email, dll.

Atribut nim merupakan unik untuk mengidentifikasikan / membedakan mahasiswa yg satu dengan yg lainnya. Pada setiap entitas harus memiliki 1 Atribut unik atau yang disebut dengan *primary Key*. Atribut adalah Setiap entitas pasti mempunyai elemen yang disebut Atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari Atribut mempunyai sesuatu yang dapat mengidentifikasikan isi elemen satu dengan yang lain. Gambar Atribut diwakili oleh simbol elips.

A. Jenis – jenis attribute

1. Identifier (*Key*) digunakan untuk menentukan suatu *entity* secara unik (*primary key*)

2. Descriptor (*nonKey attribute*) digunakan untuk menspesifikasikan kerakteristik dari suati *entity* yang tidak unik.

Relasi adalah hubungan antara beberapa entitas. Sebagai contoh relasi antar mahasiswa dengan mata kuliah dimana setiap mahasiswa bisa mengambil beberapa mata kuliah dan setiap mata kuliah bisa diambil oleh lebih dari 1 mahasiswa. Relasi tersebut memiliki hubungan banyak ke banyak.

B. Contoh – contoh ERD (*Entity Relationship Diagram*)

Kardinalitas menyatakan jumlah himpunan relasi antar entitas. Pemetaan kardinalitas terdiri dari :

1. *One-to-one* : sebuah entitas pada A berhubungan dengan entitas B paling banyak contoh diatas relasi pegawai dan departemen dimana setiap pegawai hanya bekerja pada 1 departemen.
2. *One-to-many* : sebuah entitas pada A berhubungan dengan entitas B lebih dari satu contoh diatas adalah 1 departemen memiliki banyak pegawai.
3. *Many-to-many* : sebuah entitas pada A berhubungan dengan entitas B lebih dari satu dan B berhubungan dengan A lebih dari satu juga contoh diatas adalah relasi mahasiswa dengan mata kuliah.

2.1.7. Pemrograman Web

A. PHP

Informasi kini semakin mudah untuk kita dapatkan, dengan tidak mengenal waktu dan ruang, ketika kita butuh sebuah informasi pada saat itu lah kita bisa langsung mendapatkannya. Ada sebuah teknologi yang mendukung pada

mudahnya kita mendapatkan informasi tersebut, yaitu berkat adanya teknologi internet. (Kamus Populer, 2015)

Seperti yang telah kita ketahuai bahwasannya sebuah informasi yang didapat dari jaringan internet sebetulnya terbentuk dari beberapa teknologi yang begitu kompleks. Sebut saja, jaringan *internet*, aplikasi *browser*, *web server*, *website*, bahasa pemrograman untuk membuat website, dan masih banyak lagi teknologi lainnya yang jika dikupas tidak akan selesai dalam satu artikel. (Kamus Populer, 2015)

Dan untuk kali ini kami akan membahas tentang salah satu teknologi atau lebih tepatnya disebut program yang mendukung terbentuknya sebuah *website*, dia adalah PHP. Mungkin ada sebagian dari para pembaca yang sudah lebih dahulu mengenal apa itu PHP. Untuk lebih menyempurnakan lagi pengetahuan kita tentang PHP, kami akan membahas seluk-beluk PHP pada artikel ini, yaitu mencakup *pengertian PHP* dan Sejarah PHP. (Kamus Populer, 2015)

1. Pengertian PHP

   PHP adalah kependekan dari PHP: *Hypertext Preprocessor*. Sedangkan pengertian PHP adalah bahasa pemrograman *web server-side* yang bersifat *open source*. PHP merupakan script yang terintegrasi dengan HTML dan berada pada *server* (*server side HTML embedded scripting*). PHP adalah *script* yang digunakan untuk membuat halaman website yang dinamis.

   Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru atau *up to date*. Semua *script* PHP dieksekusi pada *server* di mana *script* tersebut dijalankan.

2. Sejarah PHP

PHP pertama kali dibuat oleh Rasmus Lerdorf, yang diberi nama FI (*Form Interpreted*) dan digunakan untuk mengelola *form* dari Web, pada tahun 1995. Pada perkembangannya, kode tersebut dirilis ke umum sehingga mulai banyak dikembangkan oleh *programmer* di seluruh dunia.

Pada tahun 1997 PHP 2.0 dirilis. Pada versi ini sudah terintegrasi dengan bahasa pemrograman C dan dilengkapi dengan modulnya sehingga kualitas kerja PHP meningkat secara signifikan. Pada tahun ini pula sebuah perusahaan yang bernama Zend merilis ulang PHP dengan lebih bersih, baik, dan lebih cepat. Lalu pada tahun 1998 PHP 3.0 diluncurkan.

Kemudian pada tahun 1999 PHP versi 4.0 dirilis. PHP versi ini paling banyak digunakan pada awal abad ke 21 karena suah mampu membangun web komplek dengan stabilitas kecepaan yang tinggi.

Lalu pada tahun 2004 Zend merilis PHP 5.0. Dalam versi ini, inti dari *interpreter* PHP mengalami perubahan besar. Versi ini juga memasukkan model pemrograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemrograman ke arah paradigma berorientasi objek. Kini PHP sudah ada pada versi 6, pada versi ini PHP sudah *support* untuk *Unicode*. Juga banyak fitur penting lainnya yang telah ditambahkan ke dalam PHP 6, antara lain:

a. *Support Unicode*, dukungan terhadap penggunaan *Unicode* telah ditambahkan, sehingga lebih mudah untuk membangun dan memelihara aplikasi.

b. Perbaikan keamanan.

c. Fitur dan konstruksi baru, sejumlah fitur sintaks baru ditambahkan, seperti 64-bit *integer type*, memabngun perulangan untuk *array* multidimensi, serta dukungan untuk *labeled breaks*.

3. Kelebihan PHP

a. PHP lebih mudah untuk dikembangkan karena banyak sekalis forum atau milis yang secara khusus membahas tentang bahasa pemrograman ini.

b. PHP lebih mudah untuk difahami, pada sebagian sintak-sintak programnya menggunakan bahasa inggris.

c. PHP adalah basa pemrograman yang bersifat *open source* dan multi platform. Ini berari PHP bisa dijalankan di berbagai sistem operasi seperti Windows, Unix, Linux, Macintosh dll.

B. Mysql

Menurut Arief (2011d:152), "MySQL adalah salah satu jenis database server yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan database sebagai sumber dan pengolahan datanya."

Kepopuleran MySQL antara lain karena MySQL menggunakan SQL sebagai bahasa dasar untuk mengakses databasenya sehingga mudah untuk digunakan, kinerja *query* cepat, dan mencukupi untuk kebutuhan database perusahaan-perusahaan yang berskala kecil sampai menengah, MySQL juga bersifat *open source* (tidak berbayar) .

MySQL merupakan database yang pertama kali didukung oleh bahasa pemrograman *script* untuk internet (PHP dan *Perl*). MySQL dan PHP dianggap sebagai pasangan *software* pembangun aplikasi web yang ideal. MySQL lebih sering digunakan untuk membangun aplikasi berbasis web, umumnya pengembangan aplikasinya menggunakan bahasa pemrograman script PHP.

MySQL didistribusikan dengan licensi *open source* GPL (*General Public License*) mulai versi 3.23 pada bulan juni 2000. Software MySQL bisa di unduh melalui website resminya dihttp://www.MySQL.org atau dihttp://www.mysql.com.

2.1.8. Pengertian Blackbox Testing

Menurut (Iansyah, 2016), "pengujian dengan menggunakan metode *black box* adalah suatu pendekatan untuk dapat menguji dalam setiap fungsi pada suatu *program* agar dapat berjalan dengan benar."

Beberapa proses yang dilakukan dalam pengujian ini diantaranya yaitu :

1. Fungsi-fungsi yang tidak benar baik *input* atau pun *output* dalam hal ini hanya melihat apakah proses input dan output sudah sesuai, contohnya jika ada *software* yang menampilkan *form input* data identitas, jika *user* melengkapi *form* maka *program* akan melakukan proses simpan, namun jika *user* tidak melengkapi *form program* tidak boleh melakukan proses simpan, jika perangkat lunak tidak sesuai misalnya tidak melengkapi *form* namun dapat tersimpan, hal ini perlu untuk diperbaiki.
2. Kesalahan *interface* dalam hal kesalahan *interface* sering terjadi pada *software* yang tidak diuji coba dengan baik, misalnya tampilan web dengan

menggunakan *framework*, ada beberapa *framework* yang tidak mendukung dengan beberapa *browser*, hingga tampilan *interface* kurang maksimal saat *user* memakai *browser* yang tidak mendukung *framework* yang digunakan.

3. Kesalahan dalam struktur data atau akses database, yang sering menjadi kendala karena hal ini dapat berdampak pada akses web menjadi lamban.
4. Prilaku atau kinerja kesalahan yang ada pada perangkat lunak.
5. Inisialisasi dan penghentian kesalahan pada perangkat lunak.

   Dengan menerapkannya pengujian kotak hitam (blackbox testing) mendapatkan suatu set kasus uji yang sudah memenuhi kriteria pada MYE79 berikut yaitu :

   a. Dimana pengujian kasus yang sudah mengurangi dengan suatu perhitungan yang lebih besar dari satu, pada jumlah suatu kasus pengujian tambahan yang wajib dirancang untuk mencapai pengujian yang relevan.
   b. Pengujian kasus yang memberitahu kamu tentang hal yang ada bahkan tidak ada kesalahan, dibandingkan suatu kesalahan yang sudah terkait dengan tes yang khusus.

2.2. Penelitian Terkait

Dalam penyusunan skripsi, penulis mendapat acuan dari jurnal yang disusun secara sistematis yang diharapkan dapat membantu penulis dalam pembuatan dan pengembangan sistem.

> Permasalahan yang sering dihadapi adalah persediaan barang digudang yang tidak akurat. Persediaan barang sering kosong justru ketika pelanggan membutuhkan barang tersebut. Hal ini tentu sangat mengecewakan pelanggan dan mempengaruhi keuntungan toko. Kekosongan persediaan barang menyebabkan toko harus melakukan pemesanan barang secara mendadak

Setelah dilakukan proses audit terhadap ketersediaan bahan baku, ditemukan ketidaksamaan terhadap apa yang telah dilaporkan terhadap apa yang terjadi di dalam proses produksi yang terjadi. Ketidaksamaan tersebut berupa perbedaan nominal dari jumlah penggunaan dan kondisi dari bahan baku yang ada pada gudang. Ketidaksamaan tersebut tentu saja menjadi ancaman bagi perusahaan, akibat yang dapat terjadi adalah membengkaknya biaya penyediaan bahan baku yang tentu saja mengurangi laba dari perusahaan dan resiko perusahaan mengalami bangkrut menjadi lebih tinggi. Dari uraian diatas, diartikan perusahaan PT. Gemilang Sinergitama Mandiri mandi memerlukan pengendalian persediaan dan hasil produksinya. Oleh karena itu penulis memilih untuk mengadakan penelitian yang bertemakan sistem informasi manajemen. (Kumaladewi, Utami dan Arrosyid, 2015)

Pengelolan bahan baku adalah mengontrol arus aktifitas pada bahan baku. Dengan pengelolaan bahan baku yang baik, maka persediaan dapat dimonitor sesuai dengan kebutuhan. Untuk memudahkan akses pengelolaan bahan baku diperlukannya aplikasi berbasis *web*. Metode yang dapat digunakan untuk mengatasi pengelolaan bahan baku yang berlebihan yaitu metode *Economic Order Quantity*. Metode ini dapat menentukan jumlah barang yang harus dipesan untuk memenuhi permintaan bahan baku dengan biaya persediaan yang diminimalkan. Dalam pengembangan aplikasi ini menggunakan bahasa pemrograman *PHP* dan basis data *MySQL*. Berdasarkan pengujian, aplikasi ini dapat mengelola master data, pembelian bahan baku, pengelolaan bahan baku produksi, perhitungan biaya pemesanan bahan baku, menghasilkan laporan pembelian bahan baku, laporan produksi, jurnal dan buku besar. Perusahaan dapat menggunakan aplikasi ini untuk mengelola bahan baku produksi dengan perhitungan *Economic Order Quantity*. (Yanuar, 2016)

Sebagai perusahaan Teknologi distributor, PT.Alaisys selalu melakukan pengawasan dan Percatatan terhadap barang persedian. Pelaporan dari gudang ke kantor pusat di lakukan dengan cara menyalin data dari kartu ke dalam *microsof office excel*. Laporan dalam *format excel* tersebut harus di kirim via *email* atau *Gadget*. Sistem tersebut menjadikan pihak kantor pusat tidak dapat mengetahui data dari gudang dengan efisien. Dari permasalahan tersebut di atas, PT.Alaisys memerlukan adanya aplikasi aplikasi sistem inventory gudang berbasis web. Aplikasi dapat di gunakan dalam mengiventariskan produk yang ada pada stok stok di gudang yang meliputi pencatatan, pengolahan, dan pelaporan data data pada persediaan gudang. Dengan adanya Aplikasi berbasis web, kantor pusat dapat melihat laporan dari gudang dengan tepat sasaran, akurat, dan efisien. (Agusvianto, 2017)