BAB II

LANDASAN TEORI

2.1 Konsep Dasar Program

Konsep dasar program berisi tentang teori-teori yang berhubungan dengan pengertian program, pengertian aplikasi, pengertian kas, pengertian penerimaan kas, pengertian pengeluaran kas dan pengertian *database*.

2.1.1 Pengertian Program

Menurut Kadir (2015:1) mendefinisikan bahwa, "Program adalah kumpulan instruksi yang ditujukan untuk komputer supaya peralatan tersebut dapat melakukan tindakan-tindakan yang dikehendaki oleh pemakai program (*user*)".

Menurut Yuswanto (2008:8) mendiskripsikan bahwa, "Program merupakan kata, ekspresi, pernyataan atau kombinasi yang disusun dan dirangkai menjadi satu kesatuan prosedur, berupa urutan langkah untuk menyelesaikan masalah yang diimplementasikan dengan bahasa pemrograman sehingga dapat dieksekusi oleh komputer".

Berdasarkan pengertian diatas penulis menyimpulkan, Program adalah suatu produk atau hasil dari proses pemrograman.

2.1.2 Pengertian Aplikasi

Menurut Hendrayudi (2009:143) mendeskripsikan bahwa, "Aplikasi adalah kumpulan perintah program yang dibuat untuk melakukan pekerjaan-pekerjaan tertentu (khusus)".

Menurut Maryono dan Patmi Istiana (2008:8) mendefinisikan bahwa, "Aplikasi merupakan program yang dikembangkan untuk memenuhi kebutukan pengguna dalam menjalankan pekerjaan tertentu".

Dari pengertian diatas penulis menyimpulkan aplikasi adalah suatu perangkat lunak komputer yang memanfaatkan kemampuan komputer langsung untuk melakukan suatu tugas yang diinginkan pengguna.

2.1.3 Pengertian Kas

Menurut Rudianto (2010:131) mendefinisikan bahwa, "Kas merupakan alat pertukaran yang dimiliki koperasi dan siap digunakan dalam transaksi koperasi setiap saat diinginkan".

Menurut Supriyati (2016:7) mengemukakan bahwa, "Kas adalah uang yang disimpan diperusahaan/bank yang setiap saat bisa diuangkan tanpa mengurangi nilainya".

Berdasarkan pengertian diatas penulis menyimpulkan bahwa kas merupakan aktiva lancar yang meliputi uang kertas/logam dan benda-benda lain yang dapat digunakan sebagai media tukar/alat pembayaran yang sah dan dapat diambil setiap saat.

2.1.4 Pengertian Penerimaan Kas

Menurut Arifin (2007:95) mengemukakan bahwa, "Penerimaan kas atau kas masuk (*cash inflow*) adalah uang yang benar-benar diterima perusahaan". Sumber kas masuk menurut (Arifin 2007:95) diantaranya berasal dari:

1. Sumber penerimaan kas yang rutin dalam suatu perusahaan adalah penerimaan dari penjualan barang maupun jasa secara tunai.

2. Penerimaan dari hasil penagihan piutang, maupun pendapatan lain seperti bunga bank, deviden, jasa giro dan sebagainya.
3. Penjualan aktiva tetap yang sudah tidak ekonomis untuk digunakan dan bukan merupakan kegiatan utama perusahaan.
4. Restitusi Pajak Pertambahan Nilai (PPN).
5. Pengembalian kelebihan Pajak Penghasilan (PPh) yang telah dibayar.
6. Pinjaman dari pihak lain, seperti pinjaman dari bank.
7. Penambahan modal, diantaranya melalui penjualan saham.

Sedangkan menurut Hartoko (2011:110) mengemukakan bahwa, "Penerimaan kas merupakan penjualan tunai, penjualan kedit, utang bank, penjualan jasa, serta pendapatan lain misalnya mendapat hadiah".

Berdasarkan pengertian yang ada penulis menyimpulkan penerimaan kas adalah kas yang diterima oleh perusahaan baik berupa uang tunai maupun surat – surat berharga yang mempunyai sifat dapat segera digunakan, yang berasal dari transaksi perusahaan maupun penjualan tunai, pelunasan piutang atau transaksi lainnya yang dapat menambah kas perusahaan.

2.1.5 Pengertian Pengeluaran Kas

Menurut Hartoko (2011:120) mendefinisikan bahwa "Pengeluaran kas adalah seluruh pengeluaran yang dilakukan dari kas suatu usaha dalam kurun waktu tertentu, misalnya bulanan, semester, dan lain-lain".

Menurut Samryn (2015:176) mengemukakan bahwa, "Pengeluaran kas adalah pembayaran kepada pihak lain dengan penyerahan uang tunai, penyerahan cek, transfer kas antar bank termasuk transfer melalui fasilitas ATM".

Berdasarkan pengertian diatas penulis menyimpulkan pengeluaran kas adalah transaksi keuangan yang menyebabkan *asset* berupa kas yang dimiliki perusahaan berkurang.

2.1.6 Pengertian *Database*

Menurut Anhar (2010:45) mengemukakan bahwa, "*Database* adalah sekumpulan tabel-tabel yang berisi data dan merupakan kumpulan dari *field* atau kolom".

Menurut Yanto (2016:14) mendefinisikan bahwa, "*Database* adalah sekumpulan dari bermacam-macam tipe *record* yang memiliki hubungan antar *record*".

Berdasarkan pengertian yang ada penulis menyimpulkan *Database* merupakan sekumpulan informasi yang saling berkaitan pada tujuan tertentu.

2.2 Peralatan Pendukung (*Tools Program*)

Peralatan pendukung mampu mendeskripsikan sistem yang dirancang dalam Tugas Akhir ini, yaitu *Object Oriented Programming* (OOP) dan *Unified Modelling Language* (UML). *Unified Modelling Language* (UML) yang dibahas, meliputi *Use Case Diagram, Activity Diagram, Deployment Diagram dan Sequence Diagram*. Sedangkan *Object Oriented Programming* (OOP), antara lain Netbeans, Java, PhpMyAdmin, Xampp dan *Black Box Testing*. Penjelesan mengenai peralatan pendukung akan dijelaskan lebih lanjut.

2.2.1 Pengertian *Object Oriented Programing* (OOP)

Menurut Supardi (2009:128) mendefinisikan bahwa, "OOP merupakan cara baru berpikir, pandangan, atau paradigma baru untuk membuat program atau merancang sistem dengan memperhatikan objek, ciri objek, dan perilakunya."

Menurut Rosa A.S dan M. Shalahuddin (2014:100) mendeskripsikan bahwa, "*Object Oriented Programming* (OOP) adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya".

Berdasarkan pengertian yang ada penulis menyimpulkan bahwa OOP (*Object Oriented Programing*) merupakan suatu metode pemrograman yang berorientasi objek.

Pada saat ini, metode berorientasi objek banyak dipilih karena metodologi lama banyak menimbulkan masalah seperti adanya kesulitan pada saat mentransformasi hasil dari satu tahap pengembangan ke tahap berikutnya, misalnya pada metode pendekatan terstruktur, jenis aplikasi yang dikembangkan saat ini berbeda dengan masa lalu. Aplikasi yang dikembangkan pada saat ini sangat beragam (aplikasi bisnis, *real-time*, *utilty*, dan sebgaianya) dengan *platform* yang berbeda-beda, sehingga menimbulkan tuntutan kebutuhan metodologi pengembangan yang dapat mengakomodasi ke semua jenis aplikasi tersebut (Rosa A.S dan M. Shalahuddin, 2014:100).

Keuntungan menggunakan metodologi berorientasi objek menurut (Rosa A.S dan M. Shalahuddin, 2014:100) adalah sebagai berikut:

1. Meningkatkan produktivitas

   Karena kelas dan objek yang ditemukan dalam suatu masalah masih dapat dipakai ulang untuk masalah lainnya yang melibatkan objek tersebut *(reusable).*

2. Kecepatan pengembangan

   Karena sistem yang dibangun dengan baik dan benar pada saat analisis dan peracangan akan menyebabkan berkurangnya kesalahan pada pengkodean.

3. Kemudahan pemiliharaan

   Karena dengan model objek, pola-pola yang cenderung tetap dan stabil dapat dipisahkan dan pola-pola yang mungkin sering berubah-ubah.

4. Adanya konsistensi

   Karena sifat pewarisan dan penggunaan notasi yang sama pada saat analisis, perancangan maupun pengkodean.

5. Meningkatkan kualitas perangkat lunak

   Karena pendekatan pengembangan lebih dekat dengan dunia nyata dan adanya konsistensi pada saat pengembangannya, perangkat lunak yang dihasilkan akan maupun memenuhi kebutuhan pemakai serta mempunyai sedikit kesalahan.

2.2.2 Pengertian *Unified Modeling Language* (UML)

Menurut Nugroho (2010:6) mendiskripsikan bahwa, "UML (*Unified Modeling Language*) adalah 'bahasa' pemodelan untuk sistem atau perangkat lunak yang berparadigma 'berorientasi objek' ".

Menurut Rosa A.S dan M. Shalahuddin (2014:133) mendefinisikan bahwa, "UML *(Unified Modeling Language)* adalah salah standar bahasa yang banyak

digunakan di dunia industri untuk mendefinisikan *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek".

Dari pengertian yang ada penulis menyimpulkan bahwa UML (*Unified Modeling Language*) adalah bahasa standar untuk membangun sistem perangkat lunak (*software*).

Beberapa diagram UML (*Unified Modeling Language*) yang digunakan dalam Tugas Akhir ini, antara lain:

1. *Use Case Diagram*

Menurut Satzinger dalam Triandini dan I Gede Suartika (2012:17) mendeskripsikan bahwa, "*Use Case* adalah sebuah kegiatan yang dilakukan oleh sistem, biasanya dalam menaggapi permintaan dari pengguna sistem".

Menurut Rosa A.S dan M. Shalahuddin (2014:155) mendefinisikan bahwa, "*Use case* atau *diagram use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat".

Dari pengertian diatas penulis menyimpulkan *Use Case* adalah kegiatan atau urutan interaksi yang saling berkaitan antara sistem dan aktor.

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 (Rosa A.S dan M. Shalahuddin, 2014:155).

Syarat penamaaan 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* (Rosa A.S dan M. Shalahuddin, 2014:155).

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

2. *Activity Diagram*

Menurut Rosa A.S dan M. Shalahuddin (2014:161) mendeskripsikan bahwa, "*Activity Diagram* adalah menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis".

Menurut Nugroho (2010:62) mendeskripsikan bahwa, "Diagram aktivitas (*Activity Diagram*) sesungguhnya merupakan bentuk khusus dari state machine yang bertujuan memodelkan komputasi-komputasi dan aliran-aliran kerja yang terjadi dalam sistem/perangkat lunak yang sedang dikembangkan".

Berdasarkan pengertian diatas penulis menyimpulkan, *Activity Diagram* adalah menggambarkan aliran aktivitas dalam sistem yang sedang dirancang.

Menurut Rosa A.S dan M. Shalahuddin (2014:161) 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 (Rosa A.S dan M. Shalahuddin, 2014:161) :

a. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan
b. Urutan atau pengelompokan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya.
d. Rancangan menu yang ditampilkan pada perangkat lunak.

3. *Deployment Diagram*

Menurut Rosa A.S dan M. Shalahuddin (2014:154) mendeskripsikan bahwa, "Diagram *Deployment* atau *Deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi".

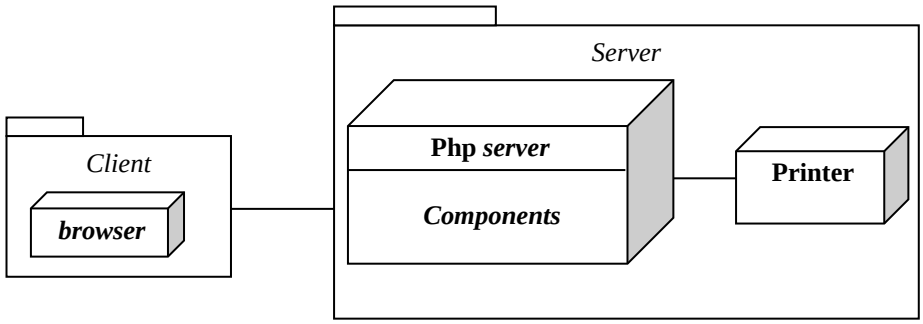
Menurut Sulistyorini (2009:24) mendefinisikan bahwa, "*Deployment Diagram* memperlihatkan konfigurasi saat aplikasi dijalankan (saat *run time*)".

Berdasarkan pengertian diatas penulis menyimpulkan, *Deployment Diagram* adalah gambaran proses pada suatu sistem yang berjalan dan bagaimana hubungan didalamnya.

Diagram *deployment* juga dapat digambarkan untuk memodelkan hal-hal berikut (Rosa A.S dan M. Shalahuddin, 2014:154) :

a. Sistem tambahan *(embedded system)* yang menggambarkan rancangan *device, node* dan *hardware.*

b. Sistem *client*/*server* misalnya seperti gambar berikut:



Sumber: (Rosa A.S dan M. Shalahuddin, 2014:154)

Gambar II.1 *Deployment Diagram* Sistem *Client/Server*

c. Sistem terdistribusi murni

d. Rekayasa ulang aplikasi

4. *Sequence Diagram*

Menurut Rosa A.S dan M. Shalahuddin (2014:165) mendeskripsikan bahwa, "*Sequence Diagram* adalah menggambarkan kelakuan objek *use case* dengan mendiskripsikan waktu hidup objek dan *massage* yang dikirmkan dan diterima antar objek".

Menurut Sulistyorini (2009:24) mengemukakan bahwa, "Diagram *sequence* merupakan diagram interaksi yang menekan pada pengiriman pesan (*message*) dalam suatu waktu tertentu".

Berdasarkan pengertian diatas penulis menyimpulkan *Sequence Diagram* adalah diagram yang menggambarkan kelakuan antar objek untuk mengirim dan menerima pesan pada waktu tertentu.

Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga

dibutuhkan untuk melihat skenario yang ada pada *use case* (Rosa A.S dan M. Shalahuddin, 2014:165).

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak (Rosa A.S dan M. Shalahuddin, 2014:165).

2.2.3 Pengertian Java

Menurut Supardi (2010:1) mendeskripsikan bahwa, "Java merupakan bahasa pemrograman yang dikembangkan dari bahasa pemrograman C++, sehingga bahasa pemrograman ini seperti bahasa C++".

Menurut Rosa A.S dan M. Shalahuddin (2014:101) mendeskripsikan bahwa, "Java merupakan bahasa pemrograman yang paling konsisten dalam mengimplementasikan paradigm pemrograman berorientasi objek".

Berdasarkan pengertian diatas penulis menyimpulkan, Java adalah bahasa pemrograman berorientasi objek yang dapat membuat seluruh bentuk aplikasi baik dekstop maupun web.

2.2.4 Pengertian Netbeans

Menurut Gata (2012:33) mengemukakan bahwa, "Netbeans IDE merupakan *editor* yang mengintegrasikan perangkat alat bantu GUI *(Graphic User Interface)* yang akan memudahkan *user* untuk membangun aplikasi dekstop".

Menurut Nofriadi (2015:4) mendefinisikan bahwa, "Netbeans merupakan sebuah aplikasi *Integrated Development Environment* (IDE) yang berbasis Java

dari *Sun Microsystem* yang berjalan diatas *swing* dan banyak digunakan sekarang sebagai editor untuk berbagai bahasa pemrograman".

Berdasarkan pengertian diatas penulis menyimpulkan Netbeans merupakan aplikasi *editor* berbasis java yang mengintegrasikan alat bantu *Graphic User Interface* (GUI) yang berjalan diatas *swing* dan banyak digunakan oleh programer untuk membuat aplikasi dekstop maupun *web*.

2.2.5 Pengertian PhpMyAdmin

Menurut Riyanto (2015:17) mendefinisikan bahwa, "PhpMyAdmin merupakan aplikasi *web* berbasis PHP yang telah banyak digunakan untuk administrasi *database* MySQL".

Menurut Nugroho (2010:88) mendeskripsikan bahwa, "PhpMyAdmin adalah suatu aplikasi *Open Source* yang berbasis web, aplikasi ini dibuat menggunakan prgram PHP, fungsi aplikasi ini adalah untuk mengakses *database* MySQL".

Berdasarkan pengertian yang ada penulis menyimpulkan, PhpMyAdmin merupakan aplikasi *web* berbasis PHP yang digunakan untuk mengakses *database* MySQL.

2.2.6 Pengertian Xampp

Menurut Riyanto (2015:1), "Xampp merupakan paket PHP dan MySQL berbasis *open source*, yang dapat digunakan sebagai tool pembantu pengembangan aplikasi berbasis PHP."

Menurut Pratama (2014:440), "Xampp adalah aplikasi *web server* bersifat instan (siap saji) yang dapat digunakan baik di sistem operasi Linux maupun dari sistem operasi Windows".

Dari pengertian diatas penulis dapat menyimpulkan bahwa, Xampp merupakan perangkat lunak bebas yang mendukung banyak sistem operasi dan gabungan dari beberapa program yang berfungsi sebagai *server*.

2.2.7 Pengertian *Black Box Testing*

Menurut Rosa A.S dan M. Shalahuddin (2015:275) mengatakan bahwa, "*Black box testing* (pengujian kotak hitam) yaitu menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dan kode program".

Menurut Maturidi (2014:68) "*Blackbox testing* adalah *text case* yang bertujuan untuk menunjukkan fungsi PL tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang diharapkan dan apakah informasi yang disimpan secara eksternal selalu dijaga kemutakhirannya".

Berdasarkan beberapa definisi *black box testing* diatas dapat disimpulkan bahwa *black box testing* adalah pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak.