**BAB III**

**LANDASAN TEORI**

* 1. **Aplikasi**

Aplikasi merupakan program yang dikembangkan untuk memenuhi kebutuhan pengguna dalam melakukan pekerjaan tertentu. Jadi aplikasi merupakan sebuah program yang dibuat dalam sebuah perangkat lunak dengan komputer untuk memudahkan pekerjaan-pekerjaan atau tugas-tugas seperti penerapan, penggunaan data yang dibutuhkan oleh pengguna (Yuhefizar, 2012).

Pengertian Aplikasi adalah program siap pakai yang dapat digunakan untuk menjalankan perintah-perintah dari pengguna aplikasi tersebut dengan tujuan mendapatkan hasil yang lebih akurat sesuai dengan tujuan pembuatan aplikasi tersebut, aplikasi mempunyai arti yaitu pemecahan masalah yang menggunakan salah satu pemrosesan data, aplikasi yang biasanya berpacu pada kompetensi yang diinginkan atau diharapkan maupun pemrosesan data yang diharapkan untuk memenuhi kebutuhan pengguna (Santoso, 2010).

* 1. **Pembayaran**

Pembayaran adalah berpindahnya hak kepemilikan atas sejumlah uang atau dari pembayar kepada penerimanya baik langsung maupun melalui media jasa-jasa perbankan (Hasibuan, 2010).

Pembayaran adalah pelunasan hutang oleh debitur kepada kreditur biasa dilakukan dalam bentuk uang atau barang, pembayaran juga biasa dalam bentuk jasa (Yunirman, 2009).

* 1. **Pengolahan Data**

Pengolahan data adalah manipulasi data kedalam bentuk yang lebih berarti berupa informasi, sedangkan informasi adalah hasil dari kegiatan-kegiatan pengolahan data yang memberikan bentuk yang lebih berarti dari suatu kegiatan atau peristiwa (Indrajanti, 2014).

Pengolahan data adalah bentuk pengolahan terhadap data untuk membuat data itu berguna sesuai dengan hasil yang diinginkan agar dapat digunakan (Fathansyah, 2012).

* 1. ***Point of Sales***

*Point of Sales* (POS) atau yang biasa disebut sebagai sistem kasir merupakan sebuah program khusus yang digunakan untuk mempermudah transaksi penjualan yang dibutuhkan pengusaha toko retail maupun restoran. Pada awalnya, *Point of Sales* digunakan sebagai sebutan untuk mesin kasir (*Cash Register*) namun, dalam perkembangannya *Point of Sales* atau POS dinilai kurang tepat jika diasosiasikan dengan *Cash Register* karena *Cash Register* sendiri sebenarnya adalah kalkulator yang dilengkapi dengan laci otomatis. Sedangkan *Point of Sales* (POS) lebih tepat dikatakan sebagai sebuah *software* pengganti mesin kasir yang lebih canggih. *Point of Sales* (POS) sendiri kini terbagi menjadi dua, yaitu system *Point of Sales* (POS) konvensional yang berupa *software* yang di *install*-kan ke sebuah komputer, hingga ada lagi yang disebut sebagai *Point of Sales* (POS) berbasis *cloud* yang mengandalkan *internet* dan memungkinkan akses secara *mobile* (Andi, 2011).

Dengan menggunakan aplikasi *Point of Sales* tentu ada beberapa keuntungan yang memberikan nilai pada perusahaan:

1. Meningkatkan Citra Usaha

Dengan menggunakan *Point of Sales* (POS) pengelolaan transaksi atau pemesanan menjadi lebih mudah, cepat, *modern* dan professional. Sehingga dapat membuat citra konsumen terhadap perusahaan menjadi lebih baik.

1. Keuntungan Berkompetisi

*Point of Sales* dirancang untuk mempermudah pengelolaan transaksi, begitu pula perusahaan akan menjadi lebih berbeda dari pada perusahaan yang belum menggunakan *Point of Sales*.

1. Keuntungan Kualitas Pelayanan

Dapat meningkatkan kualitas pelayanan pada perusahaan, karena transaksi terbilang lebih cepat, sistematis dan efisien sehingga mendukung kualitas pelayanan terhadap konsumen.

1. Memudahkan *Process Controlling*

Dengan menggunakan *Point of Sales* dapat dengan mudah melakukan *process controlling* terhadap semua transaksi yang terjadi. Semua laporan transaksi disediakan dengan cepat, sehingga mempermudah untuk melihat penjualan dan pengambilan keputusan jika sewaktu-waktu dibutuhkan.

* 1. **Analisa**

Analisa sistem dapat didefinisikan sebagai penguraian dari suatu sistem informasi yang utuh kedalam bagian-bagian komponennya dengan maksud untuk mengidentifikasi dan mengevaluasi permasalahan, kesempatan, hambatan yang terjadi dan kebutuhan yang diharapkan sehingga dapat di usulkan perbaikannya (Al Bahra, 2013).

* 1. ***Unified Modelling Language* (UML)**

*Unified Modelling Language* (UML) adalah salah satu standar bahasa pemrograman yang banyak digunakan di dunia industri untuk mengidentifikasi, *requirement*, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek (Rosa dkk, 2013).

Jenis diagram yang dapat digunakan dalam UML :

1. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas ini memiliki apa yang disebut atribut dan metode atau operasi (Rosa dkk, 2013).

Table 3.1 Komponen *Class Diagram* (Rosa dkk, 2013)

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
|  | *Generalization*, Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk. |
|  | *Navy Association*, upaya untuk menghindari asosiasi dengan lebih dari 2 objek. |
| lk.png | *Class*, himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
|  | *Collaboration*, deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi aktor. |
|  | *Association,* apa yang menghubungkan antara objek satu dengan objek lainnya. |
|  | *Realization,* operasi yang benar-benar dilakukan oleh suatu objek. |
|  | *Dependency*, hubungan dimana perubahan yang terjadi pada suatu elemen mandiri (*independent*) akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri. |

1. *Use Case*

*Use Case* merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Use Case* mendeskripsikan sebuah interaksi antara suatu atau lebih aktor dengan sistem informasi yang akan dibuat. *Use Case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu (Rosa dkk, 2013).

*Use Case* diagram mempunyai beberapa bagian penting seperti :

1. *Actor* merupakan sistem lain yang berinteraksi dengan sistem infromasi yang akan dibuat itu sendiri, jadi walaupun *symbol* dari *actor* adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use Case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.
3. Relasi, menggambarkan hubungan antara *actor*  dan *use case*.

relasi-relasi tersebut dapat dibagi menjadi :

Table 3.2 Komponen Diagram *Use Case* (Rosa dkk, 2013)

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
|  | *Actor,* fungsionalitas yang disediakan sistem sebagai unit-unit yang saling betukar pesan antar unit atau aktor. |
|  | *Use Case,* aktivititas / sarana yang disiapkan oleh bisnis / sistem |
|  | *Generalization,* hubungan generalisasi dan spesialisasi antara dua buah *use case* dimana fungsi yang satu adalah lebih umum dari lainnya. |
|  | *Include,* relasi *use case* di tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. |
|  | *Extend,* relasi *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu. Mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek. |
|  | *Assocciation*, komunikasi antara *actor* dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan *actor.* |

1. *Sequence Diagram*

*Sequence Diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup atau *life cycle* objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang di instansiasi menjadi objek-objek itu sendiri (Rosa dkk, 2013).

Table 3.3 Komponen *Sequnce Diagram* (Rosa dkk, 2013)

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
| D:\TA\TA\Laporan\Komponen Sequnce Diagra\2. boundary.jpg | *Boundary* *Class*, kelas yang memodelkan interaksi antara satu atau lebih *actor* dengan sistem. |
| D:\TA\TA\Laporan\Komponen Sequnce Diagra\3. control.jpg | *Control Class,* digunakan untuk memodelkan “perilaku” mengatur. |
| D:\TA\TA\Laporan\Komponen Sequnce Diagra\4. entity.jpg | *Entity Class*, memodelkan informasi yang harus disimpan oleh sistem, memperlihatkan struktur data dari suatu sistem. |
|  | *Object Message,* spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktivitas yang terjadi. |
| D:\TA\TA\Laporan\Komponen Sequnce Diagra\1. actor.jpg | *Actor,* menggambarkan pihak-pihak yang berperan dalam sistem. |

1. *Activity Diagram*

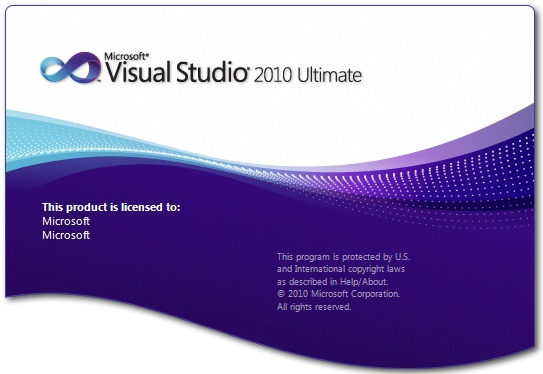
*Activity Diagram* menggambarkan berbagai alir aktivitas atau tindakan dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana alir berakhir. *Activity Diagram* juga dapat menggambarkan proses parallel yang mungkin terjadi pada beberapa eksekusi (Rosa dkk, 2013).

Table 3.4 Komponen *Activity Diagram* (Rosa dkk, 2013)

|  |  |
| --- | --- |
| **Simbol** | **Keterangan** |
|  | Aktivitas, aktivitas yang dilakukan sistem, aktivitas biasnya diawali dengan kata kerja. |
|  | Status awal, status awal aktivitas sistem |
|  | Percabangan, aosiasi percabangan jika ada pilihan aktivitas lebih dari satu. |
|  | Penggabungan, asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu. |
|  | Status akhir. Status akhir yang dilakukan sistem,  diagram aktivitas memiliki sebuah status akhir. |

* 1. ***Visual Studio* 2010**

*Visual Basic* 2010 merupakan salah satu bahasa pemrograman yang merupakan bagian dari produk pemrograman yang dikeluarkan oleh *Microsoft*, yaitu *Microsoft* *Visual Studio* 2010. *Visual Basic* 2010 sudah menggunakan *Framework*  berbasis .*NET*, sehingga anda dapat membuat *Console Application, Windows Application,* serta *Web Application* secara mudah dan terstruktur (Andi, 2011).



Gambar 3.1 Tampilan *Visual Studio* 2010

Terdapat berbagai versi *Visual Studio .NET* yaitu versi *Profesional, Premium* dan yang paling lengkap adalah versi *Ultimate*. Semua versi ini adalah versi berbayar dengan harga yang sangat mahal (Hidayatullah, 2015).

Berikut adalah kelebihan-kelebihan dari aplikasi *Visual Studio* 2010 antara lain :

1. Sederhana dan mudah dipahami

Seperti pada *Visual Studio* 2010, bahasa yang digunakan sangat sederhana sehingga lebih mudah untuk dipahami bagi yang masih baru terhadap dunia pemrograman.

1. Mendukung GUI

*Visual Studio* 2010 bisa membuat *software* dengan antarmuka grafis yang lebih *user friendly*.

1. Menyederhanakan *deployment*

*Visual Studio* 2010 mengatasi masalah *deployment* dari aplikasi berbasis *Windows* yaitu *DLL Hell* dan registrasi *COM* (*Compponent Object Model*). Selain itu tersedia *wizard* yang mempermudah dalam pembuatan *file setup.*

1. Menyederhanakan pengembangan perangkat lunak

Ketika terjadi kesalahan penulisan kode dari sisi *sintaks* (bahasa) *Visual Studio* 2010 langsung menuliskan kesalahannya pada bagian *Message Windows* sehingga *programmer* dapat memperbaiki kode dengan cepat.

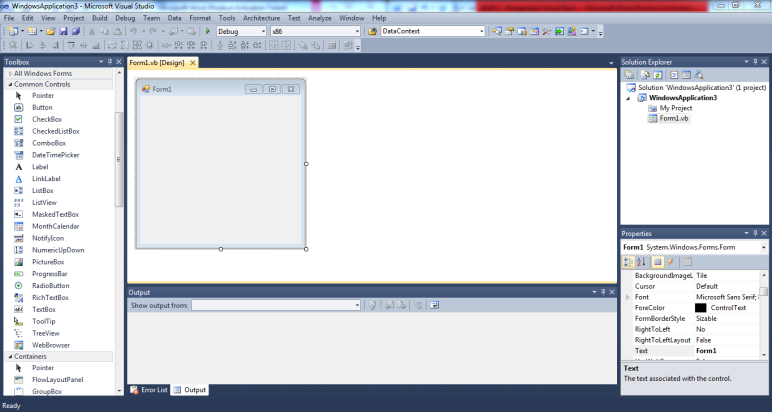
1. Mendukung penuh *Objecy Oriented Progamming* (OOP)

Memliki fitur bahasa pemrograman berorientasi objek seperti *inheritance* (pewarisan), *encapsulationi* (pembungkusan), dan *pholymorphism* (banyak bentuk).

1. Mempermudah pengembangan aplikasi berbasis *WEB*

Disediakan desainer *form Web.* Selain itu disediakan layanan *Web XML* sehingga memungkinkan suatu aplikasi “berkomunikasi” dengan aplikasi lainnya dari berbagai *platform* menggunakan *protocol* *internet* terbuka.

1. Migrasi ke *Visual Studio* 2010 dapat dilakukan dengan mudah, jika sudah mengembangkan aplikasi di *Visual Basic,* maka konversi ke *Visual Studio* 2010 dapat dijalankan dengan mudah.
2. Banyak digunakan oleh *programmer-programmer* di seluruh dunia. Salah satu keuntungannya adalah jika memiliki masalah/peretanyaan, maka bias ditanyakan kepada *programmer-programmer* lain di seluruh dunia melalui forum-forum di *internet*.



Gambar 3.2 Jendela Kerja *Microsoft Visual Basic* 2010

Pada intinya *Visual Studio* 2010 adalah salah satu dari kumpulan *tools* pemrograman yang terdapat pada paket *Visual Studio .NET*. Lingkungan pengembangan dari *Visual Studio* 2010 disebut dengan .*NET Framework*. *Framework* ini menangani bagaimana .*NET programming* membangun tipe *intrinsic, class* dan antarmuka.

* 1. **MYSQL**

MYSQL (*My Structur Query Language*) adalah suatu perangkat lunak *database* relasi (R*elational Database Management System* atau RDBMS). MYSQL itu bekerja menggunakan bahasa basis data DBMS (*Database Manangement System*). Pada MYSQL, sebuah *database* mengandung satu atau sejumlha tabel. Tabel terdiri dari sejumlah baris dan setiap baris mengandung satu atau beberapa kolom. Fungsi yang dapat dilakukan termasuk objek basis data, memanipulasi objek, mempopulasikan tabel basis data dengan data, memperbarui data yang sudah ada dalam tabel, menghapus data, melakukan *query* basis data, mengontrol akses basis data dan melakukan administrasi basis data secara keseluruhan (Arifudzaki dkk, 2010).

Data *Language* ini terbagi menjadi dua macam, yaitu :

1. DDL (*Data Definition Language*), yaiu perintah yang digunakan untuk pendefinisian suatu struktur data. Misalnya menciptakan *database*, *field*, dan sebagainya.
2. DML (*Database Manipulation Language*), yaitu perintah untuk proses manipulasi data, misanya *create, read, update,* dan *delete* (CRUD).
   1. **Diagram Alir (*Flowchart*)**

Bagan alir (*Flowchart*) adalah bagan (*chart*) yang menunjukan alir (*flow*) di dalam program atau prsedur sistem secara logika, bagan alir program (*program flowchart*) merupakan bagan alir yang mirip dengan bagan alir sistem, yaitu untuk menggambarkan prosedur di dalam sistem (Santoso, 2010).

Tabel 3.5 Simbol *Flowchart* (Santoso, 2010)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| no | Simbol | Arti | no | simbol | arti |
| 1 | Terminal Points | Awal / Akhir Flowchart | 10 | Prdefined Process | Rincian operasi berada ditempat lain |
| 2 | Input/Output | Mempresentasikan input data atau output data yang diproses atau informasi | 11 | Preparation | Pemberian harga awal |
| 3 | Penghubung | Keluar ke atau masuk dari bagian lain khususnya halaman yang sama | 12 | Punched card | Input/Output yang menggunakan kartu berlubang |
| 4 | Anak Panah | Mempresentasikan alur kerja | 13 | Dokumen | I/O dalam format yang dicetak |
| 5 | Penjelasan | Digunakan untuk komentar tambahan | 14 | Magnetic Tipe | I/O menggunakan pita magnetik |
| 6 | Proses | Mempresentasikan operasi | 15 | Magnetic Disk | I/O yang menggunakan pita magnetik |
| 7 | Keputusan | Keputusan dalam program | 16 | Magnetic Drum | I/O menggunakan drum magnetik |
| 8 | On-Line Storage | I/O yang menggunakan penyimpanan akses langsung | 17 | Manual Operation | Operasi manual |
| 9 | Punched Tape | I/O yang menggunakan pita kertas berlubang | 18 | Communic action link | Transmisi data melalui channel komunikasi |

* 1. **Pengujian**

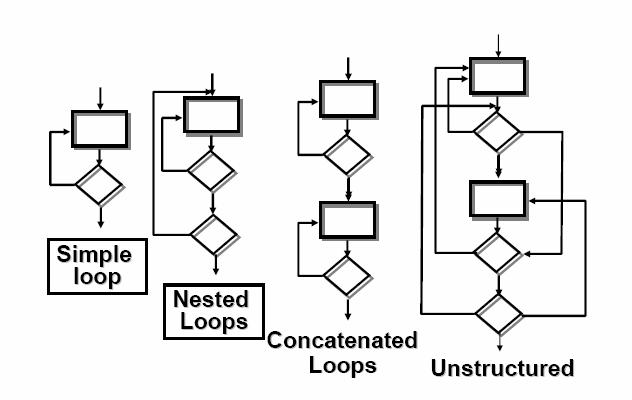
Pengujian perankat lunak adalah sebuah elemen topic yang memiliki cakupan luas dan sering dikaitkan dengan verifikasi (*verification*) dan validasi (*validation*) (V&V). verifikasi mengacu pada sekumpulan aktifitas yang menjamin bahwa perangkat lunak mengimplementasikan dengan benar sebuah fungsi yang spesifik. Validasi mengacu pada sekumpulan aktifitas yang berbeda dan mennjamin bahwa perangkat lunak yang dibangun dapat ditelusuri sesui dengan kebutuhan pelanggan (*customer*). Pengujian ini dilakukan dengan dua cara, yaitu pengujian *black box* dan *white box* (Rosa dkk, 20113).

1. Pengujian *Black-Box*

*Black box testing* adalah menguji perangkat lunak dari segi spesifikasi fungsional tanpa menguji desain dank ode program. Pengujian dimaksudkan untuk mengetahui apakah fungsi-fungsi, masukan, dan keluaran dari perangkat lunak sesuai dengan spesifikasi yang dibutuhkan. *Black box testing* dilakuka dengan membuat kasus uji yang bersifat mencoba 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 (Rosa dkk, 2013).

1. *User* memasukkan nama pemakai (*username*) dan kata sandi (*password*) yang benar.
2. *User* memasukan nama pemakai (*username*) dan kata sandi (*password*) yang salah.
3. Pengujian *White-Box*

*White box testing* yaitu menguji perangkat lunak dari segi desain dan kode program apakah mampu menghasilkan fungsi-fungsi, masukan, dan keluaran yang sesuai dengan spesifikasi kebutuhan. Pengujian ini dilakukan dengan memeriksa logika dari kode program. Pembuatan kasus uji bias mengukuti standar pengujian dari dari standar pemrograman yang seharusnya. Contoh dari pengujian kotak putih misalkan alur (dengan menelusuri) pengulangan (*looping*) pada logika pemrograman (Rosa dkk, 2013).



Gambar 3.4 Contoh Pengujian *White Box*