**BAB 2**

**LANDASAN TEORI**

**2.1 Teori Umum**

Teori umum adalah teori yang menjadi landasan pada saat melakukan sebuah penelitian.

**2.1.1 Internet**

*Internet* merupakan *Interconnection Networking* yang mempunyai arti hubungan antar komputer yang terjadi di seluruh dunai dengan sistem sebuah sistem jaringan melalui jalur telekomunikasi. *Internet* menggunakan standar [sistem](http://id.wikipedia.org/wiki/Sistem) [global](http://id.wikipedia.org/wiki/Dunia) [*Transmission Control Protocol*](http://id.wikipedia.org/wiki/Transmission_Control_Protocol)*/*[*Internet Protocol*](http://id.wikipedia.org/wiki/Internet_Protocol) *Suite* (TCP/IP) sebagai protokol pertukaran paket (*packet switching communication protocol*) untuk melayani miliaran pengguna di seluruh dunia. Rangkaian internet yang terbesar dinamakan Internet.

**2.1.2 *Entity Relationship Diagram***

*Entity relationship diagram* adalah pendekatan top-down untuk desain database yang diawali dengan mengidentifikasi data penting yang disebut "*Entity*" dan hubungan antara data yang harus direpresentasikan dalam model. Diagram ini digunakan untuk mengkonstruksikan model data konseptual, memodelkan struktur data dan hubungan antar data dan mengimplementasikan [basis data](http://id.wikipedia.org/wiki/Basis_data) secara logika maupun secara fisik dengan [DBMS](http://id.wikipedia.org/wiki/DBMS) (*Database Management system*).

Elemen pada *Entity Relationship Diagram* :

1. *Entity* (Entitas)

*Entity* (entitas) adalah sesuatu apa saja yang ada di dalam sistem, nyata maupun abstrak dimana data tersimpan atau di mana terdapat data.

2. *Relationship*

Relationship adalah hubungan alamiah yang terjadi antara entitas.

3. *Atribute*

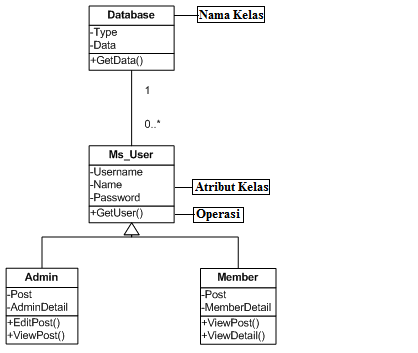
Deskripsi kelompok data yang mempunyai karakteristik yang sama (data yang mendeksripsikanentity dan relationship), merupakan *field* yang akan disimpan.(*Connoly & Begg. 2005,342*).

### 2.1.3*Unified Modeling Language* (UML)

Menurut (*Whitten dan Bentley, 2007,399*), UML (*Unified Modeling Language*) merupakan kumpulan konvensi pemodelan untuk menggambarkan fungsi-fungsi yang disediakan oleh sistem tersebut.

#### 2.1.3.1 *Class* *Diagram*

*Class diagram* adalah sebuah diagram yang menggambarkan tentang obyek-obyek bernama kelas yang menyusun suatu sistem dan juga hubungan antar kelas-kelas tersebut (*Whitten & Bentley., 2007*).



(*Whitten & Bentley., 2007*)

Gambar 2.1 Contoh *Class Diagram*

Terdapat tiga bagian pada suatu *class,* yakni :

1. Nama kelas

Nama kelas terdapat di bagian pertama *class* yang merepresentasikan nama dari *class* tersebut.

1. Atribut kelas

Atribut kelas terdapat di bagian tengah *class* yang merepresentasikan tipe data yang dimiliki.

1. Operasi

Operasi terdapat di bagian bawah *class* yang merepresentasikan kegiatan-kegiatan yang akan dilakukan oleh *class* tersebut.

Adapun notasi-notasi yang terdapat pada *class* diagram*,* seperti :

1. *Visibility*

*Visibility* digunakan untuk menentukan apakah atribut atau operasi dari sebuah kelas dapat digunakan oleh kelas yang lain.

Tabel 2.1 *Visibility*

|  |  |  |
| --- | --- | --- |
| ***Visibility*** | **Simbol** | **Deskripsi** |
| *Private* | - | Hanya dapat hanya dapat digunakan oleh kelas yang mendefinisikan |
| Protected | # | Dapat digunakan oleh kelas yang mendefinisikannya dan turunan dari kelas tersebut |
| Public | + | Dapat digunakan oleh semua kelas yang berhubungan |

1. *Multiplicity*

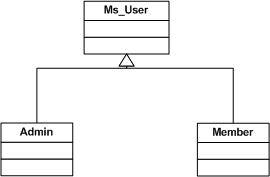
*Multiplicity* digunakan untuk menentukan banyaknya kelas yang berhubungan dengan kelas yang dimaksud.

Tabel 2.2 *Multiplicity*

|  |  |
| --- | --- |
| ***Multiplicity*** | **Deskripsi** |
| 0…1 | Nol atau satu |
| 1 | Satu |
| 0…\* | Nol atau lebih |
| 1…\* | Satu atau lebih |

1. Generalisasi

Generalisasi digunakan untuk merepresentasikan *inheritance* dalam *class* diagram dengan menggambarkan hubungannya melalui *superclass* dan *subclass*. *Superclass* adalah bentuk umum dari sebuah *subclass*, dan *subclass* adalah bentuk spesifik dari sebuah *superclass*. *Subclass* memiliki semua properti yang dimiliki oleh *superclass*.



(*Whitten & Bentley., 2007*)

Gambar 2.2 Generalisasi

1. Asosiasi

Asosiasi digunakan untuk menggambarkan adanya hubungan antar kelas dengan memberikan keterangan relasi dan *multiplicity.* Hubungan asosiasi dibedakan menjadi dua jenis, yaitu:

1. *Uni-directional*

Hubungan *Uni-directional* merupakan hubungan dimana dua kelas saling terhubung tetapi hanya salah satu yang memiliki peran dalam hubungan tesebut.

1. *Bi-directional*

Hubungan *Bi-directional* merupakan hubungan dimana dua kelas saling terhubung dan keduanya memiliki peran dalam hubungan tesebut.

1. Agregasi

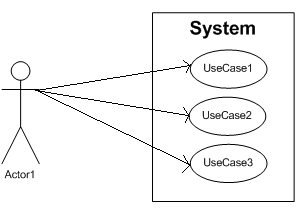
Agregasi digunakan untuk menggambarkan hubungan dimana sebuah kelas merupakan bagian dari kelas lain, tetapi bukan merepresentasikan *inheritance.*

1. Komposisi

Komposisi merupakan bentuk lain dari agregasi tetapi hubungan antar kelas komposisi lebih erat dibandingkan agregasi karena kelas A yang merupakan bagian dari kelas B tidak akan ada jika kelas B tidak ada.

#### 2.1.3.2 *Use Case* *Diagram*

*Use case* diagramadalah sebuah diagram yang menggambarkan interaksi antara sistem dengan pengguna. Jadi, *use case* diagrammenggambarkan siapa yang akan berinteraksi atau menggunakan sistem tersebut (*Whitten & Bentley., 2007,401*).



(*Whitten & Bentley., 2007*)

Gambar 2.3 *Use Case Diagram*

*Use case* diagram dideskripsikan oleh *use case narrative* yang menggambarkan bagaimana *user*  berinteraksi dengan untuk menyelesaikan pekerjaannya.

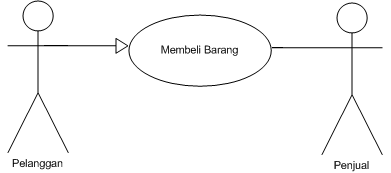
Tabel 2.3 Keterangan *User Case Diagram*

|  |  |
| --- | --- |
| **Elemen** | **Keterangan** |
| Nama *use case* | Nama *use case* harus merepresentasikan tujuan yang ingin dicapai oleh *use case* tersebut dan diawali dengan kata kerja. |
| Aktor | Aktor yang menjalankan *use case.* |
| Event yang dijalankan | Langkah-langkah yang harus dilakukan untuk menyelesaikan |
| *Precondition* | Kondisi yang harus dipenuhi sebelum menjalankan *use case.* |
| *Postcondition* | Hasil yang dicapai setelah *use case* berhasil dijalankan. |

Hubungan-hubungan pada *use case diagram*:

1. Asosiasi

Asosiasi menggambarkan adanya interaksi antara *use case* dengan aktor. Asosiasi digambarkan dengan simbol garis yang bisa mempunyai anak panah atau tanpa anak panah. Asosiasi dengan anak panah yang berasal dari aktor menunjuk ke *use case* menandakan bahwa aktor inisiator *use case* tersebut. Asosiasi tanpa anak panah menandakan interaksi antara *use case* dengan aktor penerima.

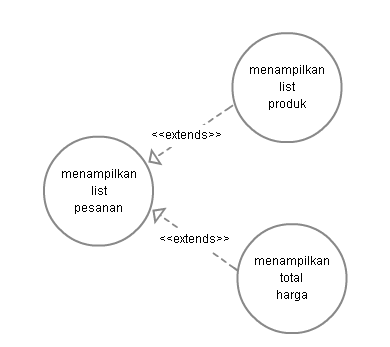


(*Whitten & Bentley., 2007*)

Gambar 2.4 Asosiasi

1. *Extends*

*Extends* digunakan ketika fungsi *use case* terdiri dari beberapa tahap yang sulit sehingga mudah dimengerti. *Extends* akan menghasilkan *use case* baru yang mewakili fungsi dari *use case* awal.

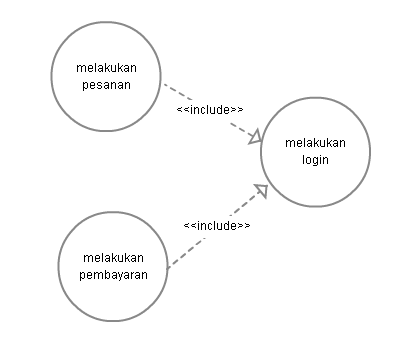


(*Whitten & Bentley., 2007*)

Gambar 2.5 *Extend*

1. *Includes/uses*

*Includes* digunakan ketika *use case* melakukan beberapa langkah yang sama yang menjadi sebuah entitas baru dengan sebutan *use case* abstrak sehingga mengurangi redundansi dalam *use case* diagram.

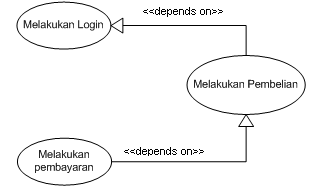


(*Whitten & Bentley., 2007*)

Gambar 2.6 *Includes*

1. *Depends* O*n*

*Depends On* digunakan untuk menggambarkan hubungan *use case* dimana suatu *use case* tidak dapat dilakukan Spabila *use case* yang lain belum dilakukan.

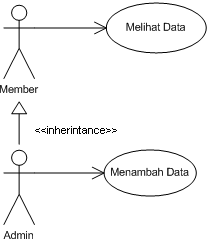


(*Whitten & Bentley., 2007*)

Gambar 2.7 *Depend On*

1. *Inheritance*

*Inheritance* digunakan untuk menggambarkan hubungan dua atau lebih aktor dalam satu sistem dengan sebuah aktor abstrak untuk menyederhanakan diagram karena aktor abstrak memiliki langkah yang dimiliki aktor lain.

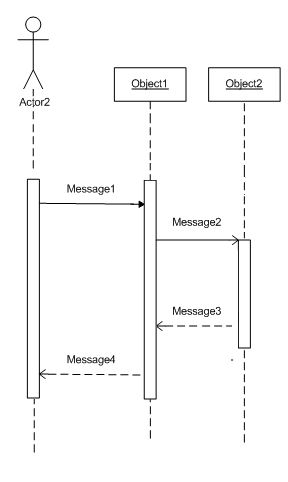


(*Whitten & Bentley., 2007*)

Gambar 2.8  *Inheritance*

#### 2.1.3.4 *Sequence* *Diagram*

*Sequence* diagrammenggambarkan model logika dari sebuah *use case* dengan pesan yang dikirimkan antar objek dalam waktu tertentu (*Whitten & Bentley., 2007,412*).



(*Whitten & Bentley., 2007*)

Gambar 2.9 *Sequence Diagram*

Berikut elemen-elemen pada *sequence* diagram :

1. Aktor

Aktor merupakan simbol yang digunakan untuk mewakili pengguna dalam berinteraksi dengan objek kelas *interface*.

1. Objek

Objek merupakan simbol yang digunakan untuk mewakili kelas-kelas pada *class* diagram*.*

1. Aktivasi

Aktivasi merupakan simbol yang digunakan untuk menggambarkan lamanya waktu dari objek saat digunakan.

1. Pesan

Pesan merupakan simbol yang digunakan untuk menyampaikan *method* dari setiap objek yang ada.

1. *Self-call*

*Self-call* merupakan simbol yang digunakan untuk menunjukkan bahwa sebuah objek mengirimkan pesan kepada objek itu sendiri.

1. *Return*

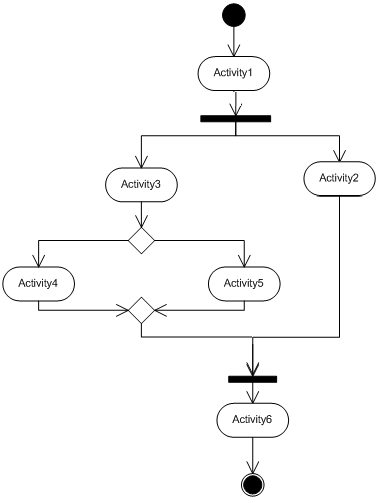
*Return* merupakan simbol yang digunakan untuk menandakan respon dari pesan yang disampaikan oleh suatu objek.

1. *Frame*

*Frame* merupakan simbol yang digunakan untuk menandakan area yang mengalami perulangan, seleksi, atau ketentuan.

#### 2.1.3.5 *Activity Diagram*

*Activity* diagram merupakan gambaran mengenai alur sebuah proses bisnis, langkah dalam sebuah *use case,* dan logika sebuah tindakan objek (*Whitten & Bentley, 2007,408*).



(*Whitten & Bentley, 2007,408*)

Gambar 2.10 *Activity Diagram*

Elemen-elemen pada *activity* diagram:

1. *Initial* *node*

*Initial node* merupakan simbol yang merepresentasikan mulainya

1. *Action*

*Action* merupakan simbol yang merepresentasikan suatu aktivitas yang digambarkan sesuai urutan *actions*.

1. *Flow*

*Flow* merupakan simbol yang merepresentasikan jalur dari satu aktivitas ke aktivitas lain

1. *Decision/Merge*

*Decision/Merge* merupakan simbol yang merepresentasikan keadaan kondisional. *Decision* mengindikasikan kondisi untuk memilih salah satu alur aktivitas sedangkan *merge* mengindikasikan bergabungnya alur aktivitas yang dipisahkan oleh *decision* sebelumnya.

1. *Fork/ Join*

*Fork* merupakan simbol yang mengindikasikan adanya dua atau lebih *actions* yang berlangsung secara bersamaan sedangkan *join* mengindikasikan berakhirnya sebuah proses paralel.

1. *Activity final*

*Activity final* merupakan simbol yang merepresentasikan akhir sebuah proses.

**2.1.4 Delapan Aturan Emas Yang Digunakan Untuk Merancang Antarmuka**

Menurut (*Shneiderman and Plaisant, 2010,313*), Ada delapan aturan yang digunakan untuk merancang sebuah aplikasi. Yang dibagi diantaranya :

1. *Consistant*

*Consistant* terhadap urutan tindakan dan rancangan antarmuka sehingga pengguna dapat lebih mudah dan terbiasa menggunakan aplikasi.

1. Kegunaan yang *universal*

Mengenal kebutuhan dari berbagai pengguna yang berbeda dan mendesain konten yang dinamis sesuai kebutuhan. Memahami perbedaan antara pemula dengan ahli, perbedaan usia, kondisi pengguna, dan perbedaaan teknologi dalam melakukan desain.

1. Memberikan umpan balik yang informatif

Memberikan umpan balik kepada pengguna ketika melakukan aksi misalnya memberikan kotak dialog peringatan dan suara ketika terjadi kesalahan.

1. Memberikan dialog untuk penutupan

Urutan aksi harus terorganisir menjadi beberapa bagian seperti bagian awal, bagian tengah dan bagian akhir. Dari bagian tersebut seharusnya diberikan umpan balik sehingga pengguna dapat mengetahui hasil dari aksi dan melanjutkan ke tahap berikutnya.

1. Adanya penanganan kesalahan

Sistem yang dibuat sebaiknya dapat mendeteksi kesalahan supaya pengguna tidak melakukan kesalahan yang fatal dan memberikan solusi untuk kesalahan tersebut.

1. Mudah kembali ke tindakan sebelumnya

Adanya fitur untuk membatalkan tindakan sebelumnya untuk menghilangkan kekhawatiran pengguna ketika mengetahui adanya kesalahan.

1. Mendukung pusat pengendalian internal

Pengguna ahli biasanya ingin menjadi pengontrol sistem bukan sebaliknya. Sehingga sistem sebaiknya dirancang seakan-akan pengguna adalah pihak memulai aksi bukan perespon..

1. Mengurangi beban ingatan jangka pendek

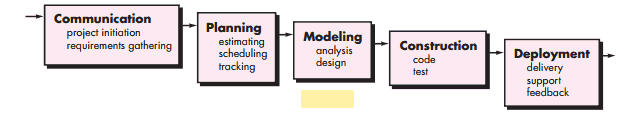
Pengguna memiliki keterbatasan kapasitas dalam memproses informasi dan mengingatnya dalam jangka pendek sehingga aplikasi yang dikembangkan sebaiknya tidak membuat pengguna harus mengingat informasi layar-layar sebelumnya.

**2.1.5 *Software Engineering***

Menurut *(Roger S. Pressman , 2010, 5)* adalah penggunaan dan pembentukan dalam prinsip – prinsip rekayasa suara dimana yang berfungsi untuk mendapatkan suatu *software* yang ekonomis sehingga dapat bekerja secara efisien pada mesin nyata.

**2.1.6 *Waterfall Model***

Menurut (*Pressman*, *2010*, *39*) *Waterfall Model* disebut juga dengan *Classic Life Cycle*, menyediakan sebuah pendekatan sistematik dan *sekuensial* pada pengembangan *software* yang dimulai dengan spesifikasi kebutuhan *customer* dan berlangsung melalui *planning, modeling, construction, deployment*, dan terakhir *ongoi*ng *support* pada *software* yang telah jadi.



*(Roger S. Pressman , 2010, 5*

Gambar 2.11 *Waterfall Model*

**2.1.6.1 *Communication***

Dalam tahap ini terdiri dari dua, yaitu *Project Initiation* dan *Requirements Gathering.* *Requirements Gathering* yang dilakukan harus secara intensif dan difokuskan kepada *software*.

**2.1.6.2 *Planning***

Setelah *Communication*, maka akan masuk ke tahap berikutnya yaitu *Planning*. Terdiri dari tiga, yaitu *Estimating*, Scheduling, *Tracking*. Pada tahap ini *software* *engineer* melakukan penjadwalan untuk menyelesaikan tahap – tahap pembuatan *software* berdasarkan waktu yang telah dijadwalkan.

**2.1.6.3 *Modeling***

Setelah *Planning*, maka akan masuk ke tahap berikutnya yaitu *Modeling*. Terdiri dari 2, yaitu *Analysis* dan *Design*. Pada tahap *Design* akan *focus* pada empat atribut program yaitu Data *Structure, Software Architecture, Interface* *Representation*, and *Algorithm Detail.* Pada tahap ini mengubah *requirements* menjadi *software* representation sebelum penulisan *code* dimulai.

**2.1.6.4 *Construction***

Setelah *Modeling*, maka akan masuk ke tahap berikutnya yaitu *Construction*. Terdiri dari 2, yaitu *Code* dan *Test*. Pada tahap ini *Design* akan diubah ke dalam bentuk bahasa yang dimengerti oleh mesin. Setelah penulisan *code* sudah selesai, maka program tersebut akan diuji. Pengujian ini dilakukan agar segala bentuk kesalahan dapat dibetulkan dan juga agar input yang dimasukkan dapat menghasilkan keluaran yang sesuai.

**2.1.6.5 *Deployment***

Setelah *Construction*, maka akan masuk ke tahap berikutnya, yaitu *Deployment*. Terdiri dari tiga, yaitu *Delivery*, *Support*, *Feedback*. Pada tahap ini *Software* *Engineer* menyediakan dokumentasi untuk semua fitur dan fungsi yang ada dalam program yang sudah dibuat sehingga memudahkan *user* dalam menggunakan program tersebut. Dan juga *Software* *Engineer* mendapat umpan balik dari *user* terhadap *software* yang dibuat untuk perbaikan yang lebih baik dengan modifikasi fungsi dan fiturnya.

## 2.2 Teori Khusus

## Teori khusus merupakan teori yang berhubungan dengan topik yang dibahas dalam penelitian ini. Teori ini menjadi pendukung dalam penyusunan penelitian.

**2.2.1 XAMPP**

XAMPP (X(Windows/Linux) Apache MySQL PHP dan Perl) merupakan paket *server web* PHP dan *database* MySQL yang paling populer di kalangan pengembang *web* dengan menggunakan PHP dan MySQL sebagai *database*-nya. Dimana sudah ada server yang siap dipakai saat akan menjalankan suatu program.

**2.2.2 *Adobe Dreamweaver***

*Software* yang kami gunakan untuk membuat website “Aplikasi Pengadaan Barang dan Jasa Pada PDAM Kota Cirebon” kami. Dengan bahasa yang digunakan adalah PHP.

**2.2.3 *Phpmyadmin***

Merupakan salah satu fitur yang kami gunakan sebagai database nantinya. Banyak sekali fitur yang bisa digunakan untuk membuat suatu database. *Easy Interface* juga membantu *user* untuk merancang suatu database.

**2.2.4 *CodeIgniter***

Adalah [aplikasi *open source*](http://id.wikipedia.org/w/index.php?title=Aplikasi_open_source&action=edit&redlink=1) yang berupa framework dengan *model* [MVC](http://id.wikipedia.org/wiki/MVC) (*Model, View, Controller*) untuk membangun [website](http://id.wikipedia.org/wiki/Website) dinamis dengan menggunakan [PHP](http://id.wikipedia.org/wiki/PHP). *CodeIgniter* memudahkan [developer](http://id.wikipedia.org/w/index.php?title=Developer&action=edit&redlink=1) untuk membuat aplikasi web dengan cepat dan mudah dibandingkan dengan membuatnya dari awal. CodeIgniter dirilis pertama kali pada [28 Februari](http://id.wikipedia.org/wiki/28_Februari) 2006. Keuntungan menggunakan *CodeIgniter* adalah gratis, berukuran kecil. URL yang sederhana, memiliki paket *library* yang lengkap, dokumentasi lengkap dan jelas, dan komunitas *CodeIgniter* luas