BAB II

LANDASAN TEORI

2.1. PHP

Menurut Arief (2011c:43) PHP adalah Bahasa server-side –scripting yang menyatu dengan HTML untuk membuat halaman web yang dinamis. Karena PHP merupakan server-side-scripting maka sintaks dan perintah perintah PHP akan diesksekusi diserver kemudian hasilnya akan dikirimkan ke browser dengan format HTML.

Dengan demikian kode program yang ditulis dalam PHP tidak akan terlihat oleh user sehingga keamanan halaman web lebih terjamin. PHP dirancang untuk membuat halaman web yang dinamis, yaitu halaman web yang dapat membentuk suatu tampilan berdasarkan permintaan terkini, seperti menampilkan isi basis data ke halaman web [1].

2.2. Sistem Informasi Geografis (GIS)

Sistem Informasi Geografis (SIG) atau Geographic Information System (GIS) menurut Irwansyah (2013:1) adalah sebuah sistem yang didesain untuk menangkap, menyimpan, memanipulasi, menganalisa, mengatur dan menampilkan seluruh jenis data geografis. Kata GIS yang terkadang dipakai sebagai istilah untuk geographical information science atau geospatial information studies yang merupakan ilmu studi atau pekerjaan yang berhubungan dengan Geographic Information System.

Sistem informasi geografis dapat disimpulkan sebagai gabungan kartografi, analisis statistik dan teknologi sistem basis data (database) [2].

2.3. Google Maps API

Menurut Svennerberg (2010:2-3), Google Map API merupakan salah satu solusi pemetaan yang populer di dalam jaringan internet.Kegunaan dari Google Map ini sendiri untuk melihat lokasi dari suatu tempat, untuk mencari posisi dari suatu alamat, untuk navigasi, dan beberapa hal lainnya. Google Map didasarkan pada prinsip, mayoritas informasi memiliki informasi suatu lokasi, jika memiliki lokasi, maka dapat ditampilkan di dalam peta. Google Map API juga menyediakan fasilitis penggunaan GoogleMap dalam kebutuhan aplikasi yang dikembangkan. Google Mapdikembangkan oleh dua suadara kandung dari Denmark, yaitu Lars danJens Rasmussen. Mereka mendirikan Where 2 Technologies , yang dimana merupakan suatu perusahaan yang didedikasikan untuk membuat solusi pemetaan. Perusahaan tersebut didapatkan oleh Google pada bulanOktober 2004, dan kedua bersaudara tersebut menciptakan Google Map. Google Map dikembangkan dengan menggunakan HTML, CSS,dan Javascript yang bekerja bersamaan. Ubin dalam peta merupakan gambar-gambar yang dimuatkan ke dalam background dengan menggunakan Ajax calls dan kemudian dimasukkan ke dalam tag suatu halaman HTML. Ketika navigasi dalam peta, API mengirim informasi tentang koordinat baru dan tingkat pembesaran peta dengan menggunakan Ajax calls yang mengembalikan gambar yang baru [3].

2.4. Framework

Menurut Pressman (2005, p282), *framework* adalah kerangka kode yang dapat disempurnakan dengan *classes* yang spesifik atau dengan fungsi yang telah dirancang untuk mengatasi masalah yang dihadapi.

Dapat disimpulkan bahwa *framework* biasanya bersifat *object oriented* dan merupakan suatu desain system yang dapat digunakan kembali. Tujuannya untuk mengurangi pembuatan kembali kode yang sama sehingga *programmer* dapat lebih fokus mengerjakan bagian lainnya[4].

2.4.1. CodeIgniter

Menurut Hakim (2010:8), CodeIgniter adalah sebuah framework PHP yang dapat membantu mempercepat developer dalam pengembangan aplikasi web berbasis PHP dibanding jika menulis semua kode program dari awal.

CodeIgniter merupakan PHP framework yang menerapkan sistem berbasis MVC (Model-View-Controller) yang secara sederhana dapat diartikan bahwa CodeIgniter memisahkan komponen-komponen didalam pengkodean aplikasi berbasis web, sehingga diharapkan nantinya lebih mudah untuk dikelola[6].

2.5. Unified Modeling Language (UML)

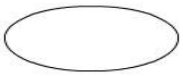
Unified Modeling Language (UML) adalah seperangkat ketentuan pemodelan yang digunakan untuk menspesifikasikan atau menjelaskan sebuah sistem *software* yang berhubungan dengan objek (Whitten dan Bentley, 2007:371) [5].

2.5.1. Use Case Diagram

Mengacu pada pendapat Whitten dan Bentley (2007:246) *use case diagram* adalah sebuah diagram yang menggambarkan interaksi antara sistem dan eksternal sistem, dan sistem dengan *user*. Secara grafis, *use case diagram* menjelaskan siapa yang menggunakan sistem dan bagaimana *user* berinteraksi dengan sistem. *Use case diagram* mempunyai notasi-notasi sebagai berikut:

a. *Use Case*

Use case diwakili secara grafis dalam bentuk elips dengan nama *use case* yang menjelaskan sistem yang ada pada diagram *use case*. Sebuah *use case* merepresentasikan tujuan tunggal dari sistem dan menggambarkan urutan kegiatan dan interaksi pengguna dalam berusaha untuk mencapai tujuan. Sistem tersebut berhubungan dengan *actor* yang ada pada diagram *use case* tersebut.



Simbol 2.1 Notasi use case symbol

b. *Actor*

Actor diwakili secara grafis dalam bentuk orang yang digunakan untuk menginisiasi sebuah sistem dengan tujuan mendapatkan informasi dari proses interaksi yang dilakukan dengan sistem. *Actor* adalah *user* yang berperan dalam sebuah sistem.

Simbol 2.2 Notasi *actor*

c. *Relationships*

Relationship diwakili secara grafis dalam bentuk garis antara dua simbol pada *use case* diagram. Definisi dari relasi yang ada dapat berbeda tergantung bagaimana garis yang digunakan dan jenis simbol yang terhubung. Hubungan-hubungan tersebut meliputi *associations*, *extends*, *uses* (*include*), *depends on*, dan *inheritance*.

1. *Associations*

Hubungan antara *actor* dan *use case* yang menggambarkan interaksi antara *actor* dan *use case*.

Hubungan ini disebut sebagai sebuah *association*. *Association* dimodelkan sebagai garis yang solid untuk menghubungkan *actor* dan *use case*. *Association* yang mengandung sebuah mata panah di ujung berhubungan dengan *use case* menunjukkan *use case* sebagai penerima. Sedangkan, *association* tanpa panah menunjukkan adanya interaksi antara *use case* dengan *actor* dengan *actor* sebagai penerima.

Gambar 2.1 Notasi *associations* pada *use case*

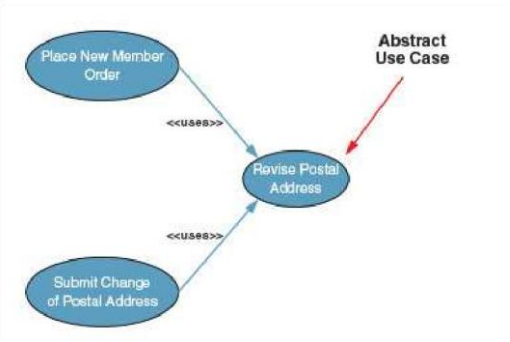
2. *Extends*

Extension use case adalah *use case* yang terdiri dari langkah-langkah yang diambil dari *use case* yang lebih kompleks untuk mempermudah *use case* sehingga memperluas fungsionalitasnya dan lebih mudah dimengerti. Setiap relasi *extends* diberi keterangan "<<extends>>".

Gambar 2.2 Notasi *extends* pada *use case*

3. *Uses (or Includes)*

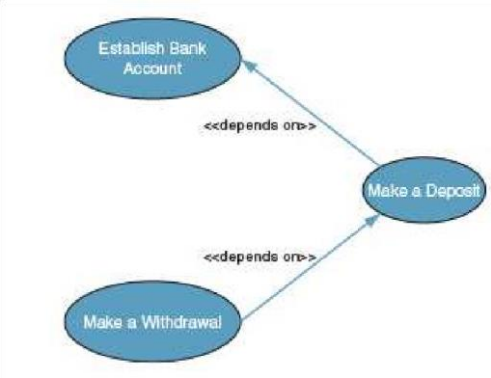
Dalam sebuah *use case diagram* akan terdapat 2 (dua) atau lebih *use case* yang memiliki *functionality* yang sama, untuk mengurangi perulangan tersebut dapat mengkombinasikan langkah yang sama tersebut yang disebut *abstract use case*. Relasi antara *abstract use case* dan *use case* yang menggunakannya disebut dengan *uses (includes) relationship* dan diberi label "<<uses>>" atau "<<include>>".



Gambar 2.3 Notasi *user* pada *use case*

4. *Depends On*

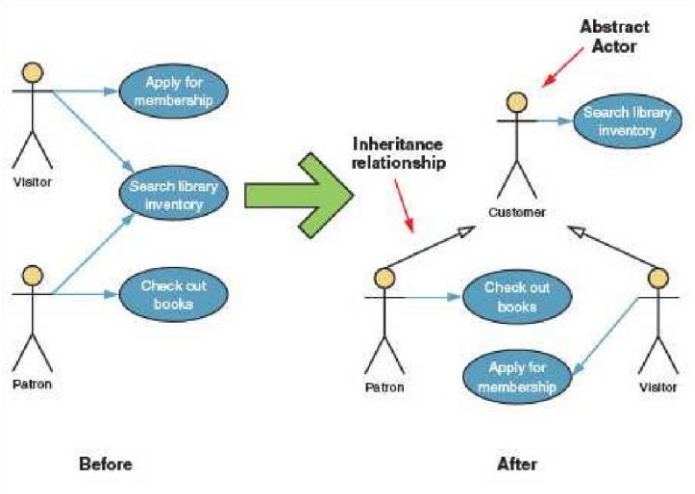
Depends on adalah relasi antara *use case* yang menunjukkan bahwa suatu *use case* tidak bisa dilakukan sampai *use case* yang sebelumnya dilakukan. Digambarkan dengan garis anak panah, mulai dari satu *use case* dan menuju *use case* yang tergantung padanya. Masing-masing garis *depends on relationship* diberi label "<<depends on>>".



Gambar 2.4 Notasi *depends on* pada *use case*

5. *Inheritance*

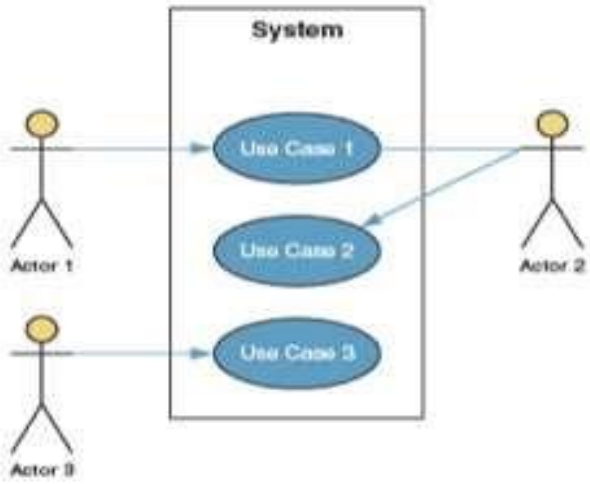
Inheritance dalam *use case*, merupakan suatu hubungan antara *abstract actor* yang dibuat untuk menyederhanakan aktivitas pada *use case* yang sama ketika ditunjuk oleh dua atau lebih *actor*.



Gambar 2.5 Notasi *inheritance* pada *use case*

6. *Use Case System*

Disimbolkan dengan bentuk persegi panjang yang mendefinisikan nama dari *use case* tersebut dan sistemsistem yang bekerja pada diagram *use case* tersebut.



Gambar 2.6 *use case model diagram*

2.5.2. Activity Diagram

Mengacu pada pendapat Whitten dan Bentley (2007:390), *activity diagram* merupakan diagram yang dapat digunakan secara grafis untuk menggambarkan aliran dari proses bisnis,langkah-langkah dari *use case*, atau logika karakteristik objek.

Berikut penjelasan notasinya.

1. *Initial node*

   Titik solid yang menggambarkan awal sebuah proses.

Simbol 2.3 Notasi *initial node* pada *activity diagram*

2. *Actions*

Segi empat bersudut tumpul menggambarkan sebuah kegiatan atau tugas yang perlu dilakukan.

Simbol 2.4 Notasi *actions* pada *activity diagram*

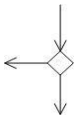
3. *Flow*

Panah menggambarkan jalur dari satu kegiatan ke kegiatan lainnya.

Simbol 2.5 Notasi *flow* pada *activity diagram*

4. *Decision*

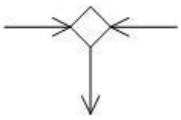
Diamond menggambarkan sebuah kegiatan keputusan.



Simbol 2.6 Notasi *Decision* pada *activity diagram*

5. *Merge*

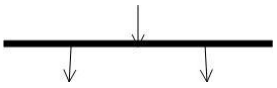
Diamond dengan dua atau lebih arus masuk dan satu arus keluar. Ini menggabungkan aliran yang sebelumnya dipisahkan oleh keputusan. Pemrosesan berlanjut dengan salah satu aliran yang masuk ke penggabungan.



Simbol 2.7 Notasi *merge* pada *activity diagram*

6. *Fork*

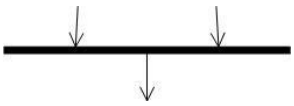
   Bar hitam dengan dua atau lebih arus masuk dan satu arus keluar. Tindakan ini untuk menggambarkan kegiatan yang dapat muncul sebagai paralel.



Simbol 2.8 Notasi *fork* pada *activity diagram*

7. *Join*

   Bar hitam dengan dua atau lebih arus masuk dan satu arus keluar, mencatat berakhirnya proses yang telah berjalan bersamaan sebelumnya. Semua tindakan yang masuk ke *join* harus diselesaikan sebelum proses berlanjut.



Simbol 2.9 Notasi *join* pada *activity diagram*

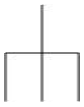
8. *Activity final*

   Titik solid di dalam sebuah lingkaran berlubang menggambarkan akhir dari sebuah proses.

Simbol 2.10 Notasi *activity final* pada *activity diagram*

9. *Subactivity indicator*

Simbol yang menunjukkan bahwa tindakan ini pecah dalam diagram lain dengan kegiatan terpisah.

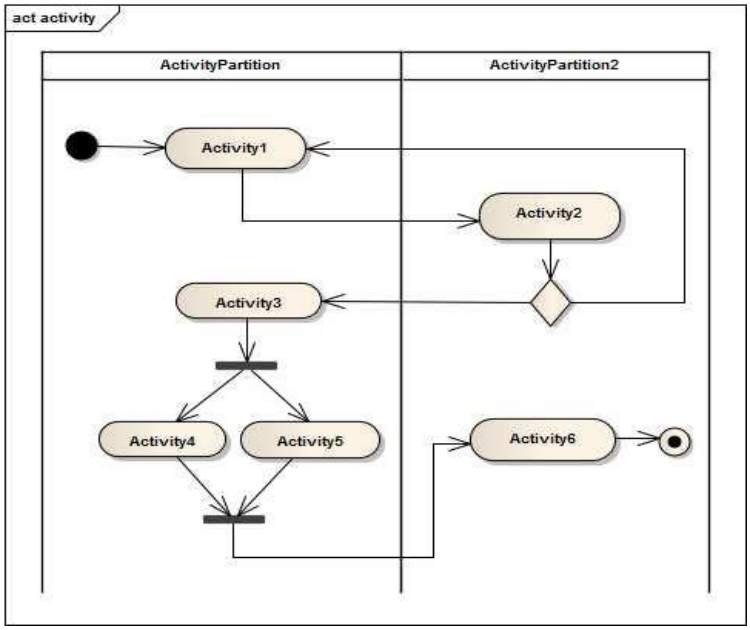


Simbol 2.11* Notasi *subactivity indicator* pada *activity diagram

10. *Connector*

Huruf di dalam lingkaran memberi alat lain untuk mengelola kompleksitas. Arus masuk ke konektor akan melompat ke arus keluar dari konektor dengan huruf yang cocok dalam lingkaran tersebut.

Simbol 2.12 Notasi *connector* pada *activity diagram*



Gambar 2.7 Contoh *activity diagram*