BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi Geografis

Sistem Informasi Geografis adalah sistem komputer yang digunakan untuk mengumpulkan, memeriksa, mengintegrasikan dan menganalisa informasi-informasi yang berhubungan dengan permukaan bumi. Pada dasarnya, istilah sistem informasi geografi merupakan gabungan dari tiga unsur pokok yaitu sistem, informasi dan geografi. Dengan demikian, pengertian terhadap ketiga unsur-unsur pokok ini akan sangat membantu dalam memahami Sistem Informasi Geografis. Dengan melihat unsur-unsur pokoknya, maka jelas Sistem Informasi Geografis merupakan salah satu sistem informasi. Sistem Informasi Geografis merupakan suatu sistem yang menekankan pada unsur informasi geografi. Istilah "geografis" merupakan bagian dari spasial keruangan. (Koko Mukti Wibowo, dkk, 2015 : 2).

II.2. *Restaurant*

Restaurant dibagi menjadi dua pengertian yang dibagi menjadi *Onsite food service* yang secara operasional menjual makanan hanya untuk mendukung aktifitas utama dan biasanya tergolong *non profit*, sedangkan *commercial foodservice* secara operasional menjual makanan adalah prioritas utama dan keuntungan diinginkan. (Kevianto Setiawan, 2015 : 2).

II.3. Peta

Peta adalah gambaran sebagian atau seluruh muka bumi baik yang terletak di atas maupun bawah permukaan dan disajikan pada bidang datar pada skala dan proyeksi tertentu (secara matematis). (Aab Abdus Salam, dkk, 2016 : 5).

II.4. PHP

PHP adalah sebuah bahasa pemrograman scripting untuk membuat halaman web yang dinamis. PHP dikatakan sebagai sebuah *server side embedded script language* artinya sintak-sintak dan perintah yang kita berikan akan sepenuhnya dijalankan oleh *server* tetapi disertakan pada halaman HTML yang seperti biasa. Aplikasi-aplikasi yang dibangun oleh PHP pada umumnya akan memberikan hasil pada tampilan *web browser*, tetapi prosesnya secara keseluruhan dijalankan di *server*. (Ayu Rizka Inayah, dkk, 2012 : 5).

II.5. MySQL

MySQL adalah *Relation Database Management System* yang didistribusikan secara gratis di bawah lisensi GPL (*General Public License).* MySQL merupakan turunan dari salah satu konsep utama dalam *database* sejak lama, yaitu *Structure Query Language*. *Structure Query Language* merupakan salah satu konsep pengoperasian *database*, terutama sebagai seleksi dan pemasukan data, yang memungkinkan pengoperasian datanya dikerjakan dengan mudah secara otomatis. (Ayu Rizka Inayah, dkk, 2012 : 5).

II.6. *Guidelines for Rappid Application Engineering*

Metode yang digunakan adalah GRAPPLE (*Guidlines for Rappid Application Engineering*), yang tujuan untuk menghasilkan sistem berorientasi objek dalam waktu yang singkat tanpa mengurangi kualitas sistem yang dibangun. GRAPPLE merupakan sebuah pemodelan proses dalam pengembangan *software* yang menekankan pada aksi-aksi yang dilakukan pada sejumlah tahapan, setiap tahap akan menghasilkan produk kerja dengan bentuk yang berorientasi objek. Dalam GRAPPLE, tahapan dapat disusun dalam bentuk yang tidak statis, sehingga setiap tahap dapat dikerjakan dengan urutan kerja yang tidak harus sesuai dengan urutan yang ada. Tahapan yang digunakan dalam GRAPPLE mencakup analisis kebutuhan sistem, pengembangan model dan diagram, pembuatan *code* sampai tahap instalasi dan evaluasi.

1. *Requirements Gathering*

Tahap ini melakukan analisis terhadap masalah, fungsi dan komponen produk yang akan dibuat (*system requirements*). Tahap ini penting, karena tahap lain tidak dapat dibuat sesuai dengan yang diinginkan jika tidak memahami produk yang akan dibuat.

2. *Analysis*

Tahap pengembangan model dari data dan informasi yang diperoleh dari *requiremens gathering*. Model merupakan bentuk transisi dari informasi dasar kedalam bentuk model dan diagram.

3. *Design*

Merupakan tahap implementasi dan perancangan dari model serta diagram yang telah dianalisis. Dalam tahap ini akan dikembangkan sejumlah objek diagram dengan fungsi, interaksi dan operasi tertentu. Diagram-diagram tersebut antara lain akan menunjukkan proses dan aktifitas pada sistem, rancangan data dan penyimpanan data, serta rancangan antar muka.

4. *Development*

Merupakan tahap penerapan model dan diagram yang telah terbentuk, antara lain dengan melakukan pengembangan *source code*, pengecekan dan *test code*, serta pembuatan *user interface*.

5. *Deployment*

Merupakan tahap terakhir yang dilakukan setelah *development*. Sistem yang terbentuk akan integrasikan dengan *hardware* maupun dengan sistem operasi yang digunakan (Novrido dan Hajad, 2011 : 10).

II.7. Adobe Dreamweaver

Adobe Dreamweaver adalah sebuah aplikasi yang dikembangkan oleh *Adobe Corporation* yang ditujukan untuk mempermudah membuat sebuah situs atau *web*. Aplikasi ini banyak diminati para pembuat *web* pemula ataupun yang sudah senior karena banyak fitur yang memang memudahkan untuk membuat situs dan *web* (Christianus, 2013 : 1).

II.8. Basis Data Dan *DBMS*

Untuk mengelola basis data diperlukan perangkat lunak yang disebut *Database Management System* (DBMS). DBMS adalah perangkat lunak sistem yang memungkinkan para pemakai membuat, memelihara, mengontrol dan mengakses basis data dengan cara yang praktis dan efisien. DBMS dapat digunakan untuk mengakomodasikan berbagai macam pemakai yang memiliki kebutuhan akses yang berbeda-beda. (Kadir, 2014 : 218).

Umumnya DBMS menyediakan fitur-fitur sebagai berikut (Kadir, 2014 : 219) :

1. Independensi data-program

   Karena basis data ditangani oleh DBMS, program dapat dipilih sehingga tidak tergantung pada struktur data dalam basis data. Dengan perkataan lain, program tidak akan terpenaruh sekiranya bentuk fisik data diubah.

2. Keamanan

   Keamanan dimaksudkan untuk mencegah pengaksesan data oleh orang yang tidak berwenang.

3. Integritas

   Hal ini ditujukan untuk menjaga agar data selalu dalam keadaan yang valid dan konsisten.

4. Konkurensi

   Konkurensi memungkinkan data dapat diakses oleh banyak pemakai tanpa menimbulkan masalah.

5. Pemulihan (*recovery*)

   DBMS menyediakan mekanisme untuk mengembalikan basis data ke keadaan semula yang konsisten sekiranya terjadi gangguan perangkat keras atau kegagalan perangkat lunak.

6. Katalog Sistem

   Katalog Sistem adalah deskripsi tentang data yang terkandung dalam basis data yang dapat diakses oleh pemakai.

7. Perangkat Produktifitas

   Untuk menyediakan kemudahan bagi pemakai dan meningkatkan produktifitas, DBMS menyediakan sejumlah perangkat produktifitas seperti pembangkit *query* dan pembangkit laporan.

II.9. *Unified Modeling Language* (UML)

Menurut Windu Gata (2013) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

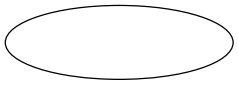
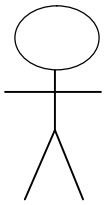
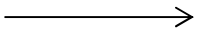
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. (Gellysa Urva dan Helmi Fauzi Siregar, 2015 : 93).

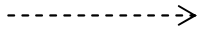
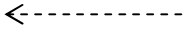
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut (Gellysa Urva dan Helmi Fauzi Siregar, 2015 : 94) :

1. *Use case Diagram*

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1.

Tabel II.1. Simbol *Use Case*

Gambar	Keterangan
 *Use case*	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
 Aktor	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki kontrol terhadap *use case*.
Asosiasi	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
 Asosiasi	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.

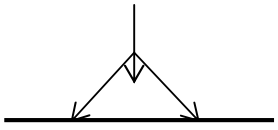
 *Include*	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
 *Extend*	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

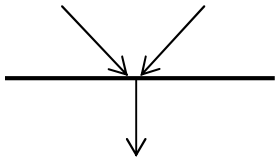
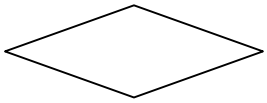
(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2.

Tabel II.2. Simbol *Activity Diagram*

Gambar	Keterangan
 *Start point*	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
 *End point*	*End point*, akhir aktifitas.
 *Activites*	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
 *Fork* (Percabangan)	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara *parallel* atau untuk menggabungkan dua kegiatan pararel menjadi satu.

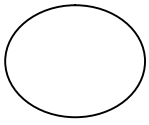
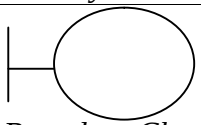
 *Join* (penggabungan) atau rake	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
 *Decision Points*	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false*.
 *New Swimline*	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

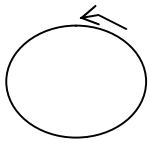
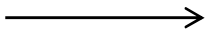
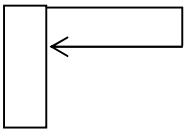
(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3.

Tabel II.3. Simbol *Sequence Diagram*

Gambar	Keterangan
 *Entity Class*	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
 *Boundary Class*	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.

 *Control class*	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
 *Message*	*Message*, simbol mengirim pesan antar *class*.
 *Recursive*	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
 *Activation*	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
 Lifeline	*Lifeline*, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan.

Class diagram secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/ Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut.

Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti yang dapat dilihat pada tabel II.4

Tabel II.4. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Gellysa Urva dan Helmi Fauzi Siregar; 2015)

II.10. Google Map API

Google Maps adalah layanan gratis Google yang cukup popular. Kita dapat menambahkan fitur Google Map dalam *web* kita sendiri dengan Google Map API. Google Map API merupakan library *JavaScript*. Untuk melakukan pemrograman Google Map API yang kita butuhkan adalah pengetahuan tentang HTML dan *JavaScript* serta koneksi Internet. Kita bisa mulai menulis program Google Map API dengan urutan sebagai berikut :

- Memasukkan Maps API *JavaScript* ke dalam HTML kita.
- Membuat *element div* dengan nama *map_canvas* untuk menampilkan peta.

- Membuat beberapa objek literal untuk menyimpan properti-properti pada peta.
- Menuliskan fungsi *JavaScript* untuk membuat objek peta.
- Menginisialisasi peta dalam *tag body* HTML dengan *event onload.* (Umi : 2010 : 2).

II.11. Normalisasi

Normalisasi merupakan parameter digunakan untuk menghindari duplikasi terhadap tabel dalam basis data dan juga merupakan proses mendekomposisikan sebuah tabel yang masih memiliki beberapa anomali atau ketidakwajaran sehingga menghasilkan tabel yang lebih sederhana dan struktur yang bagus, yaitu sebuah tabel yang tidak memiliki *data redundancy* dan memungkinkan *user* untuk melakukan *insert*, *delete*, dan *update* pada baris (*record*) tanpa menyebabkan inkonsistensi data. (Gandung Triyono, 2012 : 19).

Proses penormalan tabel ada beberapa tahap yang harus dilakukan yaitu (Gandung Triyono, 2012 : 20) :

1. *First Normal Form* (1 NF)

Sudah tidak ada *repeating group* yaitu pengulangan yang terjadi pada beberapa atribut atau kolom dalam sebuah tabel, dan juga setiap atribut harus bernilai tunggal. Atribut *multivalued*, *composite, derive* tidak tunggal. Setiap nilai dari atribut hanya mempunyai nilai tunggal.

2. *Second Normal Form* (2 NF)

Untuk menjadikan tabel normal tingkat ke 2 maka sudah 1NF dan setiap atribut yang bukan *primary key* sepenuhnya secara fungsional tergantung pada semua atribut pembentuk *primary key*.

3. *Third Normal Form* (3 NF)

Tabel sudah 2NF dan tidak memiliki *transitive dependencies*, *Transitive dependency* adalah ketika ada atribut yang secara tidak langsung tergantung pada *primary key* dan atribut tersebut juga tergantung pada atribut lain yang bukan *primary key*.

4. *Boyce-codd Normal Form* (BCNF)

Tabel dalam BCNF jika sudah 3NF dan semua *determinants* adalah *candidate keys*. Perbedaan 3NF dan BCNF adalah untuk *functional dependency* A→B, 3NF memperbolehkan ketergantungan ada dalam relasi jika B adalah *Primary Key* dan A bukan merupakan *candidate key*. Sedangkan BCNF menuntut untuk ketergantungan tetap ada dalam relasi, A harus menjadi *candidate key*.

5. *Fourth Normal Form* (4 NF)

Relasi berada pada bentuk normal keempat apabila memenuhi syarat BCNF dan tidak mempunyai *multivalue dependency.*

6. *Fifth Normal Form* (5 NF)

Tabel bentuk normal kelima sering disebut PJNF (*Projection Join Normal Form*), penyebutan PJNF karena untuk suatu relasi akan berbentuk normal kelima jika tabel tersebut dapat dipecah atau diproyeksikan menjadi beberapa tabel dan dari proyeksi-proyeksi itu dapat disusun kembali (*join*) menjadi tabel yang sama dengan keadaan semula. Jika penyusunan ini tidak mungkin dilakukan dikatakan pada relasi itu terdapat *join dependencies* dan dikatakan bersifat *lossy join*.