# BAB II

# TINJAUAN PUSTAKA

## II.1. Penelitian Terkait

Penelitian terdahulu yang berhubungan penulisan skripsi dapat dilihat pada berikut ini :

Berdasarkan penelitian Mega Yuda Rukmana, Fatwa Ramdani (2018) dengan judul penelitian "Implementasi Algoritme Dijkstra pada Webgis untuk Pencarian Lokasi SPBU di Kota Malang" Kota Malang merupakan Kota terbesar kedua di Jawa Timur yang setiap tahunnya mengalami peningkatan jumlah penduduk sebesar 9.98% atau 81,934 jiwa. Warga pendatang (pengunjung) sebagai penduduk baru merasa kesulitan untuk menemukan SPBU sesuai dengan yang dibutuhkan di tengah padatnya lalu lintas serta minimnya informasi tentang lokasi dan rute terpendek yang bisa ditempuh. Untuk mengatasi permasalahan tersebut penelitian ini menggunakan aplikasi Sistem Informasi Geografis (SIG) yang berbasis web yang mampu menyalurkan informasi tentang jalur yang bisa ditempuh untuk mempercepat proses pencarian dengan mengikuti jalur rute terpendek menggunakan algoritma Dijkstra yang telah dihasilkan.

Berdasarkan penelitian Antonio Gusmao, Sholeh Hadi Pramono, Sunaryo (2013) dengan judul penelitian "Sistem Informasi Geografis Pariwisata Berbasis Web Dan Pencarian Jalur Terpendek Dengan Algoritma Dijkstra" Telah dilakukan penelitian yang berjudul sistem informasi geografis pariwisata berbasis web dan

pencarian jalur terpendek dengan algoritma Dijkstra di Timor Leste. Untuk membantu kementerian pariwisata Timor Leste dalam mengembangkan industri pariwisata untuk memperoleh informasi yang mudah diakses dari berbagai tempat melaluir internet. Dapat meningkatkan jumlah pengunjung wisatwan yang menurun pada tahun 2011. Pemetaan SIG pariwisata berbasis web mengunakan Google Maps dan algoritma Dijkstra untuk mencari jalur terpendek dari satu titik ke titik lain pada suatu graf.

## II.2. Uraian Teoritis

### II.2.1. Sistem Informasi Geografis

Menurut Adil (2017), Sistem Informasi Geografis dapat menghubungkan berbagai data pada satu titik tertentu dibumi, menggabungkannya, men nganalisis, memetakan hasilnya dan menampilkannya dalam bentuk format grafik dan tabel. Data yang diolah pada Sistem Informasi Geografis adalah data spasial yaitu data beorientasi geografis dan merupakan lokasi yang memiliki sistem koordinat tertentu. SIG dapat di uraikan menjadi beberapa sub-sistem sebagai berikut:

1) Data Input: Sub-Sistem ini bertugas untuk mengumpulkan, mempersiapkan, dan menyimpan data yang spasial dan atributnya dari berbagai sumber, sub-sistem ini pula yang bertanggungJawab mengonversikan atau mentransformasikan format-format data aslinya ke dalam format (native) yang dapat di gunakan oleh perangkat SIG yang bersangkutan.

2) Data Output: sub-sistem ini bertugas untuk menampilkan atau menghasilkan keluaran (termasuk mengekspornya ke format yang di kehendaki) seluruh atau sebagian basis data (Spasial) baik dalam bentuk softcopy maupun hardcopy seperti halnya tabel, grafik, report, peta dan lain sebagainya.
3) Data Management: sub-sistem ini mengorganisasikan baik data spasial maupun tabel-tabel atribut terkait kedalam sebuah system basis data sedemikian rupa hingga mudah dipanggil kembali atau di retrieve (d-load ke memori), di-update, dan di-edit.

Data Manipulation Dan Analisis: Sub-sistem ini menentukan informasi-informasi yang dapat dihasilkan oleh SIG. Selain itu sub-sistem ini jga melakukan manipulasi (Evaluasi dan pengunaan fungsi-fungsi dan operator matematis dan logika) dan pemodelan untuk menghasilkan informasi yang di harapkan.

### II.2.2. Aplikasi

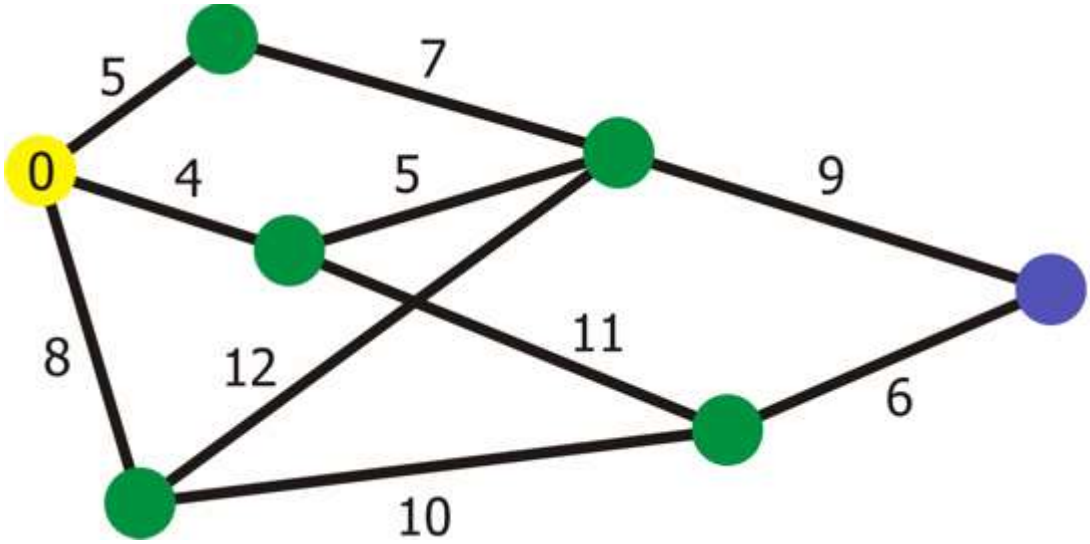
Menurut Sanjaya (2015) adalah software yang dibuat oleh suatu perusahaan komputer untuk mengerjakan tugas-tugas tertentu, misalnya Microsoft Word, Microsoft Excel. Aplikasi berasal dari kata applicationyang artinya penerapan lamaran penggunaan.

### II.2.3. Algoritma Dijkstra

Menurut (GalanGarcia et al., 2014) Algoritma Dijkstra adalah algoritma yang sangat terkenal untuk memecahkan masalah rute terpendek. Ketika diterapkan pada situasi nyata, meskipun rute terpendek dapat dihitung dengan

algoritma Dijkstra, hasil yang diperoleh tidak selalu menjadi salah satu yang dipilih. Dalam situasi lalu-lintas misalnya, pengemudi mungkin tidak tahu tepat rute terpendek untuk diikuti. Bahkan yang lebih menarik adalah kenyataan bahwa meskipun pengemudi tahu rute terpendek, dia mungkin lebih suka memilih rute yang berbeda.

Algoritma ini bertujuan untuk menemukan jalur terpendek berdasarkan bobot terkecil dari satu titik ke titk lainnya. Misalnya titik mengambarkan gedung dan garis menggambarkan jalan, maka algoritma Dijkstra melakukan kalkulasi terhadap semua kemungkinan bobot terkecil dari setiap titik.




**Gambar II.1. Algoritma Dijkstra**

Pertama-tama tentukan titik mana yang akan menjadikan node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu persatu, Dijkstra akan

melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap inilah urutan logika dari algoritma Dijkstra :

1. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi)
2. Set semua node “Belum Terjamah” dan set node awal sebagai “Node keberangkatan”
3. Dari no keberangkatan, pertimbangkan node tetangga yang belum terjamah dan hitung jaraknya dari titik keberangkatan. Sebagai contoh, jika titik keberangkatan A ke B memiliki bobot jarak 6 dan dari B ke node C berjarak 2, maka jarak ke C melewati B menjadi 6+2=8. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.
4. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah terjamah sebagai “Node terjamah”. Node terjamah tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
5. Set “Node belum terjamah” dengan jarak terkecil (dari node keberangkatan) sebagai “Node Keberangkatan” selajutnya dan lanjutkan dengan kembali ke step 3

### II.2.4. *Phpmyadmin*

*PhpMyAdmin* adalah sebuah aplikasi Open Source yang berfungsi untuk memudahkan manajemen MySQL. Dengan menggunakan *PhpMyAdmin*, dapat membuat *database*, membuat tabel, meng-*insert*, menhapus dan meng-*update* data dengan GUI dan terasa lebih mudah, tanpa perlu mengetikkan perintah SQL secara manual. (MADCOMS 2016: 186)

### II.2.5. Database MySQL

Menurut Hidayatullah dan Jauhari (2015:180) MySQL adalah salah satu aplikasi DBMS yang sudah banyakoleh para pemogram aplikasi web. Contoh DBMS lainnya adalah : PostgreSQL (freeware), SQLServer,MS Access dari Microsoft,DB2 dari IBM,Oracledan Oracle Corp, Dbase, FoxPro, dsb".

### II.2.6. Normalisasi

Normalisasi merupakan sebuah teknik dalam logical desain sebuah basis data / database, teknik pengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik tanpa redudansi. Tujuan normalisasi adalah mengorganisasikan data kedalam tabel-tabel untuk memenuhi kebutuhan pemakai, menghilangkan kerangkapan data, mengurangi kompleksitas, mempermudah modifikasi data. (Mukhlisulfatih Latief : 2016)

1. Proses Normalisasi

   a. Data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu kebeberapa tingkat.

   b. Apabila tabel yang diuji belum memenuhi persyaratan tertentu maka tabel tersebut pelu dipecah menjadi beberapa tabel yang lebih sederhana sampai memenuhi bentuk yang optimal.

2. Tahapan Normalisasi :

   1) Bentuk tidak normal : Menghilangkan perulangan grup.

**Tabel II.1. Contoh bentuk tidak normal (Unnormal)**

| No-Mhs | Nama Mhs | Jurusan | Kode-MK | Nama-MK | Kode Dosen | Nama Dosen | Nilai |
|---|---|---|---|---|---|---|---|
| 2683 | Welli | MI | M1350 | Manajemen DB | B104 | Ati | A |
| | | | M1465 | Analisis Perc. Sistem | B317 | Dita | B |
| 5432 | Bakti | AK | M1350 | Manajemen DV | B104 | Ati | C |
| | | | Akn201 | Akuntansi | D310 | Lia | B |
| | | | MKT300 | Dasar Pemasaran | B212 | Lola | A |

***(Sumber : Mukhlisulfatih Latief : 2016)***

2) Bentuk Normal pertama (1NF) : Menghilangkan ketergantungan sebagian.

Yaitu : suatu relasi dikatakan sudah memenuhi bentuk normal kesatu bila setiap data bersifat atomik yaitu setiap irisan baris dan kolom hanya mempunyai satu nilai data.

**Tabel II.2. Contoh Bentuk Normal Pertama (1NF)**

| No-Mhs | Nama Mhs | Jurusan | Kode-MK | Nama-MK | Kode Dosen | Nama Dosen | Nilai |
|---|---|---|---|---|---|---|---|
| 2683 | Welli | MI | M1350 | Manajemen DB | B104 | Ati | A |
| 2683 | Welli | MI | M1465 | Analisis Perc. Sistem | B317 | Dita | B |
| 5432 | Bakti | AK | M1350 | Manajemen DV | B104 | Ati | C |
| 5432 | Bakti | AK | Akn201 | Akuntansi | D310 | Lia | B |
| 5432 | Bakti | AK | MKT300 | Dasar Pemasaran | B212 | Lola | A |

***(Sumber : Mukhlisulfatih Latief : 2016)***

3) Bentuk Normal kedua (2NF) : Menghilangkan ketergantungan transitif.

Yaitu : suatu relasi dikatakan sudah memenuhi bentuk normal kedua bila relasi tersebut sudah memenuhi bentuk normal kesatu dan atribut yang bukan key sudah tergantung penuh terhadap key-nya.

**Tabel II.3. Contoh Bentuk Normal Kedua (2NF)**

| Kode-MK | Nama-MK | Kode Dosen | Nama Dosen |
|---|---|---|---|
| M1350 | Manajemen DB | B104 | Ati |
| M1465 | Analisis Perc. Sistem | B317 | Dita |
| M1350 | Manajemen DV | B104 | Ati |
| Akn201 | Akuntansi | D310 | Lia |
| MKT300 | Dasar Pemasaran | B212 | Lola |

***(Sumber : Mukhlisulfatih Latief : 2016)***

4) Bentuk Normal ketiga (3NF) : Menghilangkan anomali-anomali hasil dari ketergantungan fungsional. Yaitu : suatu relasi dikatakan sudah memenuhi bentuk normal ketiga bila relasi tersebut sudah memenuhi bentuk normal kedua dan atribut yang bukan key tidak tergantung transitif terhadap key-nya.

**Tabel II.4. Contoh Tabel Mahasiswa Dan Tabel Kuliah (3NF)**

| No_Mhs | Nama Mhs | Jurusan |
|---|---|---|
| 2683 | Welli | MI |
| 5432 | Bakti | AK |

***(Sumber : Mukhlisulfatih Latief : 2016)***

### II.2.7. UML (*Unified Modeling Language*)

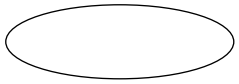
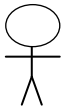
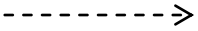
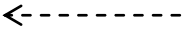
Menurut Windu Gata (2013) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (*UML*). *UML* adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. *UML* merupakan metodologi dalam mengembangkan sistem 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, Hal : 93).

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan *UML* adalah sebagai berikut:

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.5 dibawah ini:

**Tabel II.5. Simbol *Use Case***

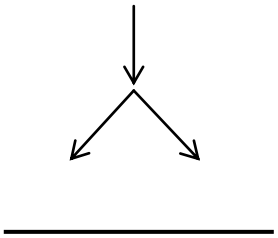
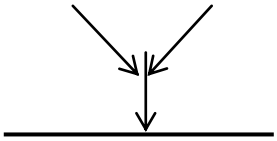
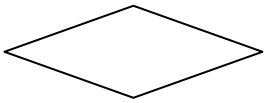
| Gambar | Keterangan |
|---|---|
|  | *Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*. |
|  | Aktor adalah *abstraction* dari orang atau sistem yang 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 control terhadap *use case*. |
|  | 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 antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem. |
|  | *Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program. |
|  | *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.6 dibawah ini:

**Tabel II.6. Simbol *Activity Diagram***

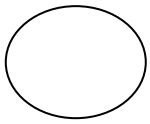
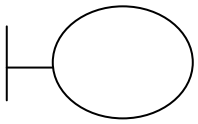
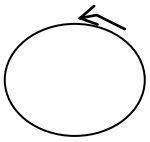
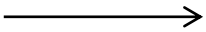
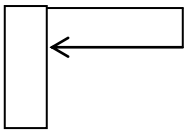
| Gambar | Keterangan |
|---|---|
|  | *Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas. |
|  | *End point*, akhir aktifitas. |
|  | *Activites*, menggambarkan suatu proses/kegiatan bisnis. |
|  | *Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu. |
|  | *Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi. |
|  | *Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.* |
|   | *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.7 dibawah ini :

**Tabel II.7. Simbol *Sequence Diagram***

| Gambar | Keterangan |
|---|---|
|  | *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*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak. |
|  | *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*, simbol mengirim pesan antar *class*. |
|  | *Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri. |
|  | *Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi. |
|  | *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 tanggng 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.8 dibawah ini :

**Tabel II.8. *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)***