Aplikasi Mobile WEB Map Service Pada Mobile Device Dengan SVGT

Sigit Priyanta, Ghulam Imanuddin dan Suci Karunia Prilistya

Abstract— Teknologi informasi geografi berkembang sangat pesat dalam beberapa dekade terakhir ini. Sekarang ini, WMS (Web Map Services) tidak hanya dapat menghasilkan gambar raster tetapi juga gambar vektor. Contohnya adalah SVGT (Scalable Vector Graphics Tiny) yang merupakan bagian dari SVG (Scalable Vector Graphics) yang digunakan pada piranti mobile device. Mahalnya biaya komunikasi antara handheld device dengan jaringan internet melalui GPRS (Global Pocket Radio System) menimbulkan sebuah masalah dalam penerapan aplikasi mobile mapping. Untuk mengeliminasi masalah tersebut, harus dibuat sebuah aplikasi mobile mapping yang dapat mentransfer data sekecil mungkin.

Didalam penelitian ini, dicoba untuk dibuat sebuah server pemetaan yang memanfaatkan teknologi SVGT dan XML. Kedua format data tersebut digunakan untuk menghasilkan data sekecil mungkin agar dapat menghemat biaya komunikasi antara klien dan server. Pada akhirnya, aplikasi yang dibangun akan menjadi sebuah server yang memiliki klien berupa handheld / mobile device.

Aplikasi ini dibangun dengan bahasa pemrograman PHP dan memanfaatkan database PostgreSQL beserta ekstensi PostGIS-nya pada sisi server dan J2ME (Java 2 Micro Edition) pada sisi mobile client. Aplikasi yang telah dibangun mampu untuk menampung peta jalan dari sebuah daerah, menampilkannya dalam format SVGT kepada klien, serta mencari rute terpendek antara dua buah jalan dengan menggunakan algoritma Floyd-Warshall. Mobile device sebagai client mempunyai beberapa fungsi utama, yaitu download map dengan format SVGT dari server, menyimpan map ke dalam record store, menampilkan map dan juga melakukan fungsi pan dan zoom terhadap map. Fungsi lainnya adalah searching atau mencari titik tertentu pada map sesuai dengan permintaan user, request path ke server untuk mendapatkan jalur terpendek antara dua titik, dan set mark atau menyimpan titik-titik pada peta yang telah diberi tanda oleh user.

Keywords— *SVG, SVGT, Mobile Mapping, Web Map Services, J2ME, Java 2 Micro Editon.*

S. Priyanta, Ilmu Komputer Fakultas MIPA Universitas Gadjah Mada, Yogyakarta 55281, e-mail : seagatejogja@ugm.ac.id

G. Imanuddin, Ilmu Komputer Fakultas MIPA Universitas Gadjah Mada, Yogyakarta 55281.

S. Karunia Prilistya, Ilmu Komputer Fakultas MIPA Universitas Gadjah Mada, Yogyakarta 55281.

I. PENDAHULUAN

I.I LATAR BELAKANG MASALAH

Teknologi informasi geografi berkembang sangat pesat dalam beberapa dekade terakhir ini. Teknologi informasi mengubah secara cepat penggunaan *Geographic Information System (GIS)* dari aplikasi *dekstop* klasik menjadi *business service market*. Dengan komputasi yang ada dimana-mana, beberapa tahun lagi setting berubah secara dramatis dan pasar *GIS* klasik akan berubah secara drastis [7].

Hingga tahun sembilanpuluhan, arsitektur *GIS* difokuskan pada lingkungan *standalone* (*static*). Arsitektur ini telah diubah agar aplikasi *GIS* pada *workstation* berubah menjadi aplikasi *GIS* pada *mobile device* [7].

Aplikasi *mobile mapping* bukan merupakan *conventional GIS* yang dimodifikasi agar bisa dioperasikan pada *mobile device*, melainkan merupakan sistem yang dibangun menggunakan paradigma baru yang fundamental.

Piranti *mobile* yang mendukung *J2ME (Java 2 Micro Edition)* menawarkan kepada pengguna dan pengembang sebuah *open interface*, sehingga pengembang aplikasi dan pengguna dapat men-*download* sesuai dengan perminataan. Sekarang ini *J2ME* merupakan salah satu dari *platform* pengembangan *mobile device* yang paling terkenal [1].

Saat ini *WMS (Web Map Service)* tidak hanya dapat menghasilkan gambar raster tetapi juga gambar vektor. *SVGT (Scalable Vector Graphics Tiny)* merupakan bagian dari *SVG (Scalable Vector Graphics)*, yang digunakan pada *mobile device*. Format *SVGT* sangat cocok digunakan untuk *mapping* geografi, khususnya untuk *mobile device* yang mempunyai *bandwidth* yang sangat terbatas. Hal ini karena *SVGT* merupakan gambar yang berbasis vektor dan mempunyai ukuran file yang kecil [1].

I.II PERUMUSAN MASALAH

Rumusan masalah dalam penelitian ini adalah :

1.Bagaimana pengembangan aplikasi *WMS (Web Map service)* menggunakan PHP dan progresql sebagai *server*, yang akan menghasilkan sebuah data peta yang berformat *SVGT*.

2.Bagaimana mengembangkan aplikasi *mobile web map service* menggunakan teknologi *J2ME (Java 2 Micro Edition)* dan *SVGT (Scalable vector Graphics Tiny)* untuk penentuan jarak terpendek.

I.III BATASAN MASALAH

Batasan masalah yang akan dibahas pada penelitian ini adalah :

Hasil dari penelitian ini, akan diujikan menggunakan emulator pada sistem lokal.

Server dapat menyajikan peta dan kemungkinan rute terpendek yang dapat dilalui antara dua tempat dalam satu wilayah.

Perhitungan rute terpendek dilakukan berdasarkan panjang suatu ruas jalan tanpa memperhitungkan arah jalan (searah atau dua arah).

Implementasi WMS pada sisi server menggunakan bahasa pemrograman PHP dan database PostgreSQL dengan ekstensi PostGIS.

Aplikasi client berjalan pada Mobile Device yang mendukung MIDP 2.0 dan CLDC 1.1, JSR 226 API yang merupakan API untuk menampilkan gambar 2D berbasis vektor seperti SVGT, dan GPRS(General Packet Radio Service) untuk koneksi internet.

I.IV TUJUAN PENELITIAN

Tujuan penelitian ini adalah untuk mengembangkan *WMS (Web Map Services)* server menggunakan PHP dan PostgreSQL yang dapat memberikan data peta dan rute terpendek antara dua tempat dalam format *SVGT* serta mengembangkan aplikasi *client map* atau peta pada *mobile device* menggunakan teknologi *J2ME (Java 2 Micro Edition)* dan *SVGT (Scalable vector Graphics Tiny)*.

I.V TINJAUAN PUSTAKA

Hui [1], dalam tesisnya yang berjudul *Design and Implement a Cartographic Client Application For Mobile Devices using SVG Tiny and J2ME* mengulas pengembangan kartografi *client* untuk *mobile devices* menggunakan teknologi *SVGT* dan *J2ME*. *Mobile device* yang disimulasikan dalam komputer *desktop* mendapatkan data sebagai respon dari *WMS* berupa format vektor (*SVG*) maupun raster (*PNG*), melakukan *parsing XML* dan kemudaian menampilkan data tersebut. Dalam tesis ini juga terdapat pembahasan mengenai analisa dan desain dari struktur sistem seperti *user interface* dan *code structure*.

Liu [4], dalam tesisnya yang berjudul *Mobile Map: A Case Study in the Design & Implementation of a Mobile Application* memaparkan desain dan implementasi aplikasi *client*/*server* pada *mobile device*. Selain itu juga dipaparkan teknik untuk menambah *performance client* pada sistem komputasi *mobile client*/*server*, serta mengidentifikasikan teknik yang paling menjanjikan, dan dapat diadaptasikan ke dalam aplikasi kepariwisataan.

Luqun [5], dalam sebuah jurnal yang berjudul *A Research on Development of mobile GIS Architecture* memaparkan tentang konsep dan arsitektur dari *Mobile GIS.* Dalam tulisan tersebut dijelaskan juga mengenai bagaimana cara komunikasi antara *client* dan *server GIS*, serta *logic diagram* dari *Mobile GIS.* Paper ini merepresentasikan *four tier client/server Mobile GIS architecture.*

II. TEORI, MODEL DAN DESAIN

II.I SISTEM INFORMASI GEOGRAFI (SIG)

Sistem informasi geografi merupakan sistem yang berbasiskan komputer yang digunakan untuk menyimpan dan memanipulasi informasi-informasi geografis. Literatur lain menyebutkan, sistem informasi geografi adalah sistem komputer yang digunakan untuk memasukkan (*capturing*), menyimpan dan menampilkan data-data yang berhubungan dengan posisi-posisi di permukaan bumi.

II.II MODEL DATA SPASIAL

Ada 2 macam pendekatan yang digunakan untuk merepresentasikan komponen data spasial yaitu model raster dan model vektor.

II.III POST GIS

PostGIS adalah sebuah modul tambahan untuk *RDBMS* PostgreSQL *server*, yang digunakan untuk menangani data-data spasial beserta operasi-operasi spasial (Wikipedia,http://en.wikipedia.org/wiki/PostGis, 2006).

Objek spasial yang didukung oleh PostGIS adalah "*Simple Feature*" yang didefinisikan oleh *OpenGIS Consortium (OGC)*. OpenGIS mendefinisikan dua cara untuk merepresentasikan objek spatial, yaitu *Well-Known Text* (*WKT*) dan *Well-Known Binary* (*WKB*). Keduanya mengandung informasi tentang tipe objek beserta koordinatnya.

OpenGIS juga memerlukan format penyimpanan internal untuk objek spasial termasuk *Spatial Referencing System Identifier* (*SRID*). *SRID* ini diperlukan ketika membuat objek spasial untuk dimasukkan dalam *database* [6].

II.1V *MOBILE DEVICE*

Mobile device merupakan istilah yang sangat luas yang membungkus terminal informasi, alat-alat informasi, *Personal Digital Assistant (PDA), Mobile phone* dan lain sebagainya. *Mobile device* dapat diklasifikasikan secara luas ke dalam dua kategori yaitu :

1. Ponsel dengan kemampuan komputasi yang bertambah, termasuk tampilan gambar dan kemampuan interaksi via gambar (*Smartphone*, *Communicator*)
2. Komputer *portable*, yang dapat di-*upgrade* untuk kemampuan komunikasi suara (misalnya *PDA*).

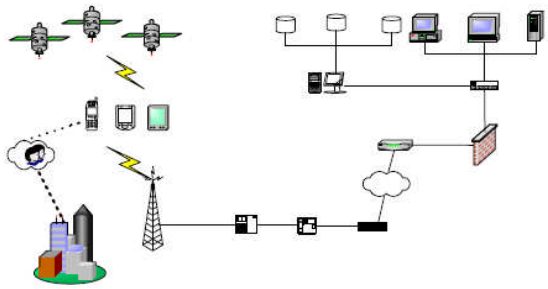
Masing-masing *Mobile device* mempunyai *Operation System* (*OS*) yang memberikan *interface* dan mengendalikan sinkronisasi piranti. Sekarang ini ada empat *OS* besar di pasaran, yaitu *Symbian*, *Smartphone*, *Palm OS*, dan *Linux*. *Symbian* digunakan oleh *Nokia*, *Motorala* dan *Ericsson*. Pada tahun 2005, *Mobile phone* yang telah menginstal *Symbian OS* menempati 51% dari *smartphone* di dunia [2].

II.V KONSEP *MOBILE MAPPING*

Dari sudut pandang tradisional, *Geographics Information System* (*GIS*) dapat didefinisikan sebagai sistem komputer yang digunakan untuk mengumpulkan, mensimulasi, memproses, mencari, menganalisa dan mendeskripsikan data geografi [5].

Mobile GIS merupakan sistem aplikasi dari sistem informasi geografi dan komunikasi *Mobile* yang mengintregasikan *GPS*, *Wireless LBS*, *Mobile communication* (seperti *GSM*, *GPRS*, dan *CDMA*, dan lain sebagainya) [5].

Sebagai pengganti *software* tradisional dan data yang besar, pengembangan *Mobile GIS* akan berubah menjadi jenis baru dari layanan dimana konsumen dapat mendapatkannya dari *wireless internet*, sebagai contoh *Navigate service*, *Moving object monitor Service* dan lain sebagainya.



Gambar 1. Arsitektur dari of Mobile GIS (Luqun & Minglu, 2004)

Ada tiga model aplikasi *map* pada *Mobile device* berdasarkan konektifitasnya, yaitu [7]. :

1. *Offline mode* (*standalone*), aplikasi dan data disimpan secara lokal seperti *phone memory* ataupun *memory card.*
2. *Server connected mode*, aplikasi diinstal secara lokal, tetapi semua data diambil dari *centralized database* yang disimpan di server.
3. *Hybrid mode*, beberapa data diinstal secara lokal dan *frequently business data* akan diambil dari *server*.

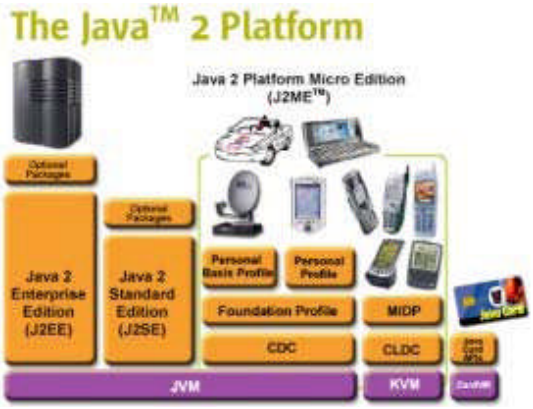
Masing-masing model mempunyai kelebihan dan kekurangan. *Offline mode* dapat bekerja sendiri (kapan saja dan dimana saja). Dalam mode ini, semua data harus diinstal terlebih dahulu, oleh karena itu ini akan menyediakan informasi yang dinamik. Di lain pihak, *server connected mode* lebih mahal dalam pengaksesan aplikasi dan biasanya tergantung pada kekuatan sinyal [7].

II.V.I *PLATFORM JAVA*

Sun microsystem telah mendefinisikan tiga *platform Java* yang masing-masing

diarahkan untuk tujuan tertentu dan lingkungan komputasi yang berbeda-beda yaitu :

1. *Standar Edition* (*J2SE*) : Didesain untuk berjalan pada komputer *dekstop* dan komputer *workstation*.
2. *Enterprise Edition* (*J2EE*) : Dengan *built-in* mendukung untuk *Servlet*, *JSP*, dan *XML*, edisi ini ditujukan untuk aplikasi berbasis server.
3. *Micro edition* (*J2ME*) : Didesain untuk piranti dengan memori terbatas, layar *display* terbatas dan kekuatan pemrosesan yang juga terbatas.



Gambar 2. Tiga platform Java 2

II.V.II PENGENALAN SVG

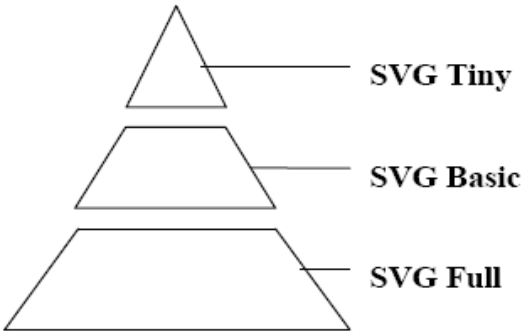
SVG (*Scalable Vector Graphics)* merupakan format file baru untuk menampilkan gambar dalam pengembangan web yang berbasis *XML* (*eXtensible Markup Language*). *SVG* berfungsi untuk menampilkan gambar 2 dimensi ke dalam kode *XML* [8].

SVG menyediakan tiga tipe dari obyek gambar yaitu *vector graphic shapes* (misalnya *path* yang terdiri atas garis lurus dan kurva), *multimedia* (seperti gambar *raster* and *video*) dan *text*. Obyek gambar dapat di kelompok-kelompokkan, diberi *style* dan digabung ke dalam obyek yang telah ada sebelumnya [8].

Kelebihan *SVG* yang paling utama adalah gambar bisa diperbesar atau diperkecil tanpa mengurangi kualitas (*scalable*), karena dibuat berdasarkan metode vektor (*vector*), bukan *pixel* (seperti format *gambar* pada umumnya, *GIF*, *JPEG* dan *PNG*). Sehingga memungkinkan *developer web* dan juga *designer* untuk membuat *gambar* dengan kualitas yang tinggi [3].

II.V.III *SVG MOBILE*

Karena adanya perkembangan *Mobile* yang sangat pesat, ada permintaan untuk *SVG* yang lebih cocok untuk lingkungan *wireless*, *W3C* mendefinisikan dua "*Mobile profiles*" dari *SVG* yaitu *SVG Basic* (untuk *handset* yang *higher-end* dan *PDA*) dan *SVG Tiny* (untuk *smartphone* dan piranti yang *lower-end*). *SVG Tiny* merupakan bagian dari *SVG Basic*, yang merupakan bagian dari *full SVG*. *SVG Tiny* and *SVG Basic* dikenal sebagai "*Mobile SVG*" [9].



Gambar 3. Piramida dari SVG, SVGB dan SVGT (Hui, 2006)

SVG Mobile merupakan format *vector graphic* yang terdiri atas deskripsi geometrik dari semua bentuk dalam image dengan semua atributnya, termasuk warna, ukuran, dan ketebalan *outline* diantaranya. *Mobile SVG* adalah *vendor-neutral* dan *open*, berbasis *XML* dan didesain khusus untuk pengiriman dan tampilan *wireless*. Dibuat oleh *W3C*, *Mobile SVG* merupakan format standar untuk mendeskripsikan gambar vektor 2D dalam *XML* [9].

Keuntungan dari *Mobile SVG* adalah untuk *encoding* dan menampilkan *content* seperti animasi, *map* dan gambar yang interaktif. *Scalability* dari *Mobile SVG* memperbolehkan *resizing* gambar agar sesuai dengan ukuran atau *resolusi* layar atau *printer* tanpa kehilangan kualitas. Ini merupakan keuntungan dalam dunia *wireless*, dimana *mobile device* mempunyai bentuk dan ukuran yang kecil.

Keuntungan *Mobile SVG* untuk aplikasi *mobile* yang sensitif terhadap *bandwidth*, yaitu ukuran filenya lebih kecil dari *bitmap*, sehingga mengurangi waktu *download* melalui jaringan *wireless* [9].

II.V.IV *SVGT*

SVGT merupakan gambar yang berbasis *XML*, *source code* untuk gambar *SVGT* tidak ditangani dalam format teks yang berubah-ubah,

tetapi gambar *SVGT* atau dokumen *SVGT* adalah dokumen *XML* yang "*well-formed*".

Elemen *SVG* terdiri atas semua elemen-elemen yang lain. Ini merupakan elemen dokumen dari dokumen. Di dalam elemen tersebut, *developer* dapat menambah gambar dengan menggunakan elemen dan atribut *shape* yang standar yang disediakan oleh *SVGT* [1].

Obyek lain bisa ditempatkan di dalam obyek *SVGT* untuk membentuk *graphic*. Disediakan beberapa komponen dasar seperti *rectangle*, *circle*, *ellipse*, *line*, *polyline* dan *polygon*. Parameter *style* yang berhubungan juga didefinisikan. Selanjutnya, perintah yang lain yaitu path disediakan untuk mendeskripsikan garis dan kurva diantara titik. Sebagai tambahan, transformasi, *masking*, *linking*, *temporal effect* dan animasi juga memungkinkan [1].

III. Analisis Masalah

III.I Analisis Kebutuhan

Elemen yang paling penting dalam sebuah *map server* yang sukses adalah adanya konten yang berkualitas serta pelayanan yang baik dan cepat terhadap klien. Masalah akses konten yang cepat bergantung pada kecepatan transfer media koneksi serta ukuran konten yang akan diberikan atau dikirim kepada klien. *Server* juga harus dapat melayani *request* peta dan *request* rute dari klien.

Dalam penelitian ini juga dibutuhkan sebuah klien untuk melakukan ujicoba *server*. Client merupakan aplikasi mobile yng dikembangkan menggunakan J2ME.

III.II Spesifikasi Kebutuhan Sistem

III.II.I Kebutuhan Fungsional

Berdasarkan analisis kebutuhan di atas, maka aplikasi yang dibangun mempunyai kebutuhan fungsional sebagai berikut :

Aplikasi ini dibangun sebagai sebuah *server* pemetaan yang menampung data geografis berupa peta, dan dapat diakses oleh klien yang berupa *handheld device*.

Aplikasi dapat melakukan perhitungan jarak terpendek untuk rute yang diminta klien, dan mengembalikan kepada klien data rute yang dilalui dalam bentuk *SVGT*.

Aplikasi dapat membuat file berformat *SVGT* dari data spasial yang terdapat dalam *DBMS* PostgreSQL, yang nantinya file tersebut akan dikirim ke klien.

Aplikasi ini merupakan *client* yang berjalan pada *mobile device* yang dapat menampilkan *map* dengan format *SVGT* serta melakukan fungsi *pan*, *zoom* dan *view all* pada gambar tersebut.

Aplikasi dapat men-*download map* dengan format *SVGT* dari *server*. File yang di*download* akan disimpan dalam database *client*.

1. Melakukan *Request Path* yaitu permintaan *path* yang menunjukkan jarak terpendek berdasarkan titik asal dan tujuan yang diberikan *user* .
2. Melakukan *set mark* atau menyimpan titik-titik pada peta yang telah diberi tanda oleh *user*.

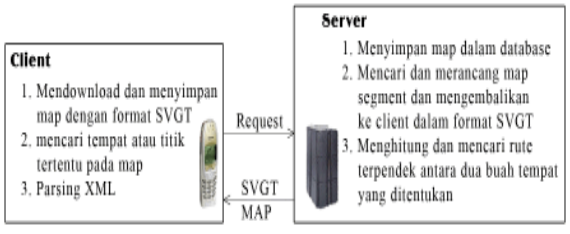
III.III Arsitektur Sistem

III.III.I Arsitektur Model *Client Server*

Mobile Mapping merupakan aplikasi jaringan *client-server*. Aplikasi yang akan dibuat merupakan *server* tunggal. Klien terletak pada *mobile device* dengan memori, tempat penyimpanan, tampilan, *network bandwith* dan fungsionalitas yang kecil serta metode penginputan yang kurang. *Server* terletak pada komputer atau *workstation* dalam internet yang mempunyai memori dan *storage* yang besar serta kemampuan pemrosesan yang cepat.

Mobile device sebagai klien akan melakukan permintaan ke *server*, mendapat respon dari *server* dan kemudian menampilkan atau menyimpan file *SVGT* yang diberikan oleh *server*.

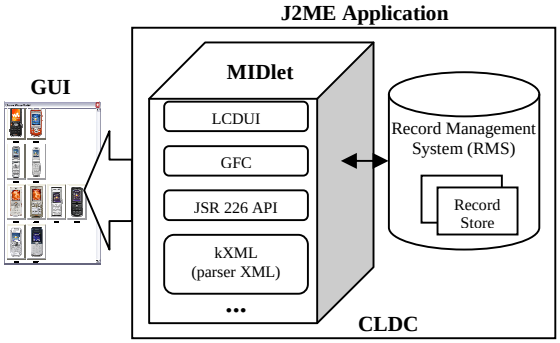
Dalam aplikasi yang dibuat, *server* dapat menghitung dan mencari rute terpendek antara dua buah tempat yang ditentukan berdasarkan *request* dari klien, dan dapat menyajikannya dalam format *SVGT* .



Gambar 4. Model mobile mapping client-server

III.III.II ARSITEKTUR SISTEM *CLIENT*

Aplikasi yang dibuat difokuskan pada sisi *client*. *Mobile device* sebagai *client* mempunyai beberapa fungsi utama, yaitu *download map* dengan format *SVGT* dari server, menyimpan *map* ke dalam *record store*, menampilkan *map* dan juga melakukan fungsi *pan* dan *zoom* terhadap *map*. Fungsi lainnya adalah *searching* atau mencari titik tertentu pada *map* sesuai dengan permintaan *user*, *request path* jarak terpendek ke server, dan *set mark* atau menyimpan titik-titik pada peta yang telah diberi tanda oleh *user*.

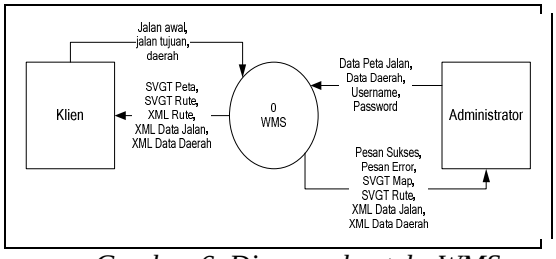


Gambar 5. Arsitektur Mobile mapping Client

III.IV RANCANGAN PROSES WMS

III.IV.I DIAGRAM KONTEKS

Diagram Alur Data (DAD) Konteks memperlihatkan sistem sebagai sebuah proses dan memiliki tujuan memberi pandangan umum mengenai sistem. Dari DAD konteks dapat dilihat proses interaksi dengan lingkungannya. Ruang lingkup sistem yang yang akan dibuat dapat diketahui prosesnya melalui Diagram Alur Data konteks yang ditunjukkan pada gambar 6.



Gambar 6. Diagram konteks WMS

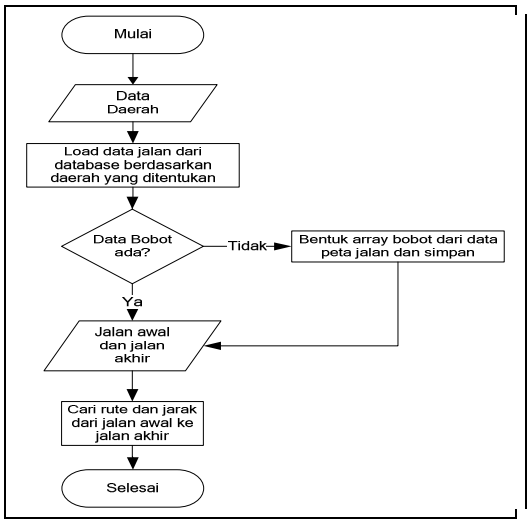
Diagram konteks *WMS* (Gambar 6) menggambarkan diagram arus data dasar. Dari diagram konteks di atas dapat dilihat 2 entitas luar yang berhubungan dengan *WMS* yaitu klien dan administrator. Dimana klien dapat mengirimkan data jalan awal, data jalan akhir, dan data daerah kedalam sistem, dan menerima data *SVGT* peta jalan, *SVGT* data rute, *XML* data rute, *XML* data daerah, dan *XML* data jalan dari sistem. Administrator dapat memasukkan data berupa peta jalan dan data daerah kedalam sistem, dan menerima pesan sukses dan error dari sistem, tampilan peta yang diinput dalam format SVGT, tampilan rute dalam format SVGT, data daftar daerah, serta data daftar jalan.

III.IV.II FLOWCHART PROSES

III.IV.II.I FLOWCHART PROSES PRA KALKULASI

Proses ini digunakan untuk menghitung bobot dan rute terpendek antara dua jalan dalam daerah yang ditentukan. Pertama, proses akan meminta input nama daerah yang akan diproses. Kemudian, proses akan mengecek apakah data bobot sudah ada atau belum. Jika belum maka akan dibuat data bobot terlebih dahulu. Jika sudah ada, proses akan me-*load* data jalan dalam daerah tersebut dari database.

Kemudian, untuk setiap pasangan jalan yang ditentukan, dicari rute dan jarak terpendeknya. Hasilnya akan disimpan ke dalam tabel tbRute. Gambaran *flowchart* dari proses ini ditunjukkan pada gambar 7.



Gambar 7. Flowchart proses pra-kalkulasi

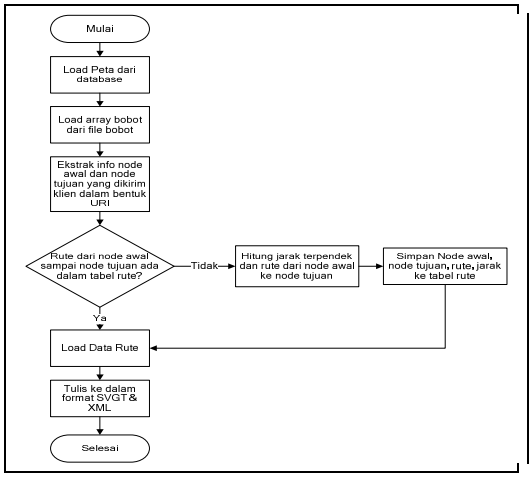
III.IV.II.II *FLOWCHART* PROSES PENENTUAN RUTE JALAN

Proses ini berguna untuk menjawab *request* dari klien yang ingin menentukan rute terpendek antara dua buah tempat. Pertama, proses akan me-*load* semua data jalan dari tabel

tbJalan dan data rute dari tbRute. Selanjutnya di-*load* juga data bobot dari file bobot sesuai dengan daerah yang ditentukan.

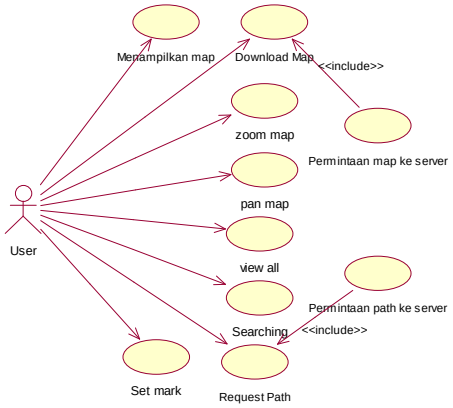
Lalu proses akan meng-ekstrak informasi tempat awal dan tempat tujuan dari *URI* yang diberikan *user*. Selanjutnya proses akan mencari apakah rute tersebut ada dalam tabel rute. Jika iya, maka data rute tersebut diambil, disusun, dan ditulis kedalam format *SVGT* dan *XML*.

Namun jika data rute yang diminta belum ada, maka proses akan melakukan perhitungan jarak terpendeknya. Ketika hasil perhitungan jarak terpendek didapatkan, maka hasil tersebut akan disimpan dalam *database* dan dikembalikan pada *user* yang me-*request*. Gambaran *flowchart* dari proses penentuan rute jalan ini ada pada gambar 8 dibawah ini.



Gambar 8. Flowchart proses penentuan rute jalan

III.IV.III *USE CASE DIAGRAM*



Gambar 9. Use case diagram

Use case diagram diatas memodelkan perilaku *(behavior)* dari sistem dan subsistem pada *mobile mapping*. Dalam diagram tersebut terdapat sebuah aktor yaitu pengguna (*user)* dan sepuluh *use case* yaitu :

Menampilkan *map* dengan format *SVGT*

Melakukan *download map* dengan format *SVGT* dari *server*

Melakukan permintaan *map* ke *server*

Melakukan penggeseran (*pan*) terhadap *map* yang telah dibuka

Melakukan *zoom* terhadap *map* yang telah dibuka

Menampilkan *map* secara keseluruhan (*view all*) terhadap *map* yang telah dibuka

Melakukan pencarian(*searching*) dan menampilkan tempat atau titik tertentu pada *map* sesuai dengan permintaan *user* .

1. Melakukan *Request Path* dari titik asal dan tujuan tertentu.
2. Melakukan permintaan *path* yang menunjukkan jarak terpendek berdasarkan titik asal dan tujuan yang diberikan *user* .
3. Melakukan *set mark* atau menyimpan titik-titik pada peta yang telah diberi tanda oleh *user*.

III.V RANCANGAN BASIS DATA WMS

Aplikasi WMS server menggunakan database dengan rancangan berikut :

Tabel 1. *Tabel tbDaerah*

Field	Tipe
Id	Int4
Nama_Daerah	Varchar(255)

Tabel 2. *Tabel tbJalan*

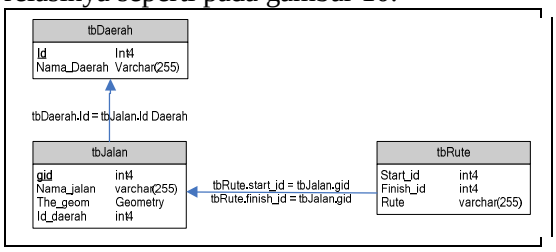
Field	Tipe
Gid	Int4
Nama_Jalan	Varchar(255)
Length_kil	Float8
The_geom	Geometry
Id_Daerah	Int4

Tabel 3. *Tabel Rute*

Field	Tipe
Start_id	Int4
Finish_id	Int4
Rute	Varchar (255)

III.V.I RELASI ANTAR TABEL

Dari ketiga tabel diatas, dapat dibentuk relasi dari tabel-tabel tersebut. Relasi yang terjadi adalah antara tbDaerah dan tbJalan pada tbDaerah.id = tbJalan.id_daerah, tbRute dan tbJalan pada tbRute.start_id = tbJalan.gid dan tbRute.finish_id = tbJalan.gid. adapun diagram relasinya seperti pada gambar 10.



Gambar10. Relasi antar tabel yang terbentuk

III.V.II RANCANGAN DATABASE *MOBILE DEVICE*

Karena keterbatasan *mobile device*, aplikasi *mobile mapping* yang akan dibangun menyimpan *map* dan data index pencarian dalam sebuah database sederhana menggunakan *RMS* (*Record Management System*) yang terdapat pada *J2ME* yang merupakan *record-oriented database*. Data yang diperlukan dalam aplikasi ini disimpan dalam *record store* yang terdiri atas record. Masing-masing *record* terdiri atas *record id* dengan tipe *integer* yang merupakan *primary key* dalam *database*, dan *array of bytes* untuk menyimpan data.

MIDlet yang akan dibuat mempunyai tiga macam *record store* yaitu *record store* untuk menyimpan *map*, *record store* untuk menyimpan *index* dan *record store* untuk menyimpan titik-titik pada map yang telah diberi tanda oleh *user*.

Pada saat proses download *map*, file *map* yang di-download disimpan dalam *record store map*. Nama *map* sesuai dengan nama yang diberikan oleh server. Kemudian file *map* diparsing dan data jalan disimpan dalam *record store index*.

III.V.III *RECORD STORE MAP*

Record store map terdiri atas tiga *kolom* yaitu *ID*, *idDaerah*, *mapName* dan *mapFile*. ID merupakan record id yang dibuat oleh RMS secara otomatis. *idDaerah* merupakan ID yang diberikan *server*, *mapName* merupakan nama dari *map* dan *mapFile* berisi file *SVGT*. Ketiga kolom terakhir ini bertipe *array of byte*. Jumlah *record* dalam *Record store* ini dibatasi lima buah dan setiap record bersifat unik artinya satu dengan yang lain tidak boleh memiliki data yang sama. Hal ini dimaksudkan untuk menghemat memory mobile device.

Tabel 4. Record store map

IDMap	idDaerah	MapName	MapFile

III.V.IV *RECORD STORE INDEX*

Record store yang kedua adalah *record store index* yang terdiri atas empat kolom yaitu *ID*, *StreetName*, *x*, dan *y*. ID merupakan record id, indexName menyimpan nama jalan, x menyimpan absis dari jalan dan y menyimpan ordinat dari jalan. Jumlah *record store index* sesuai dengan jumlah *map* yang disimpan dan namanya sesuai dengan nama *map* yang bersangkutan.

Tabel 5. Record store index

ID	indexName	x	y

III.V.V *RECORD STORE MARK*

Record store yang ketiga adalah *record store mark* yang terdiri atas empat kolom yaitu *ID*, *markName*, *x*, dan *y*. ID merupakan record id, *markName* menyimpan nama jalan, x menyimpan absis dan y menyimpan ordinat dari titik. Jumlah *record store mark* sesuai dengan jumlah *map* yang disimpan dan namanya sesuai dengan nama *map* yang bersangkutan.

Tabel 6. Record store mark

IDStreet	markName	x	y

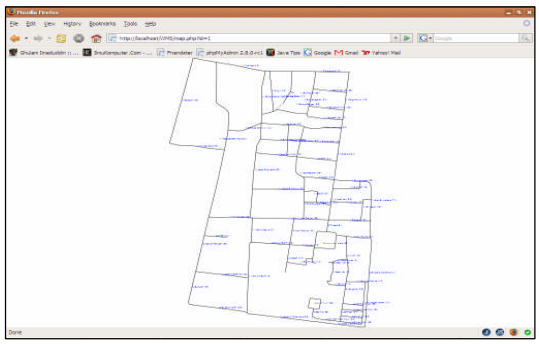
IV. HASIL DAN ANALISIS

IV.I IMPLEMENTASI PROGRAM *WMS* (SISI SERVER)

Aplikasi yang telah dibangun terdiri atas dua bagian utama, yaitu bagian *user* dan bagian administrator data. Pada bagian *user*, ada empat buah proses utama, yaitu proses lihat peta, cari rute, *download* daftar jalan, dan *download* daftar daerah. Pada bagian administrator terdapat lima proses utama, yaitu proses *login*, input peta, lihat peta, pra kalkulasi, dan *logout*.

IV.I.I PROSES LIHAT PETA

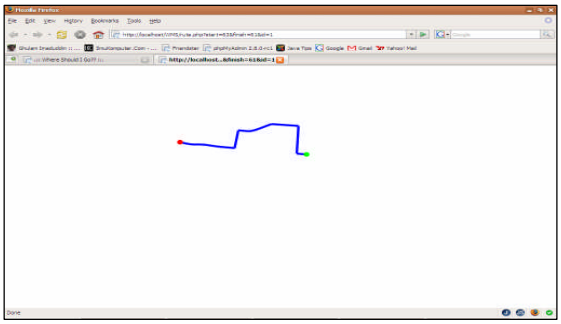
Proses ini akan menghasilkan peta dari daerah yang ditentukan dalam format *SVGT*. Untuk mengakses halaman ini, klien dapat me-*request* halaman ini dengan alamat **[alamat_server]/[skrip_path]/map.php?id=[id_daerah]**. Dalam pengujian, digunakan *server* lokal. Jadi, untuk mengakses fitur ini, digunakan alamat : http://localhost/WMS/map.php?id=1. Dimana id daerah yang digunakan adalah satu, yang merupakan contoh data dalam penelitian ini, yaitu data jalan sebagian Kota Yogyakarta. Contoh tampilan dari halaman ini (pada *browser*) seperti pada gambar 11.



Gambar 11. Hasil SVGT peta pada browser

IV.I.II PROSES CARI RUTE

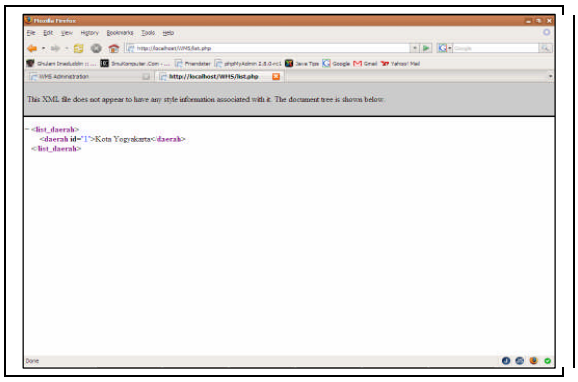
Proses ini akan menghasilkan rute jalan dari daerah yang diminta dalam format *SVGT*. Untuk mengakses halaman ini, klien dapat me-*request* halaman ini dengan alamat **[alamat_server]/[skrip_path]/rute.php?start=[id_jalan_mulai] &finish=[id_jalan_akhir]&id=[id_daerah]**. Dalam pengujian, digunakan *server* lokal. Jadi, untuk mengakses fitur ini, digunakan alamat http://localhost/WMS/rute.php?start=63&finish=43&id=1. Contoh tampilan dari halaman ini (pada *browser*) seperti pada gambar 12.



Gambar 12. Tampilan SVGT rute jalan pada browser

IV.I.III *DOWNLOAD* DAFTAR DAERAH

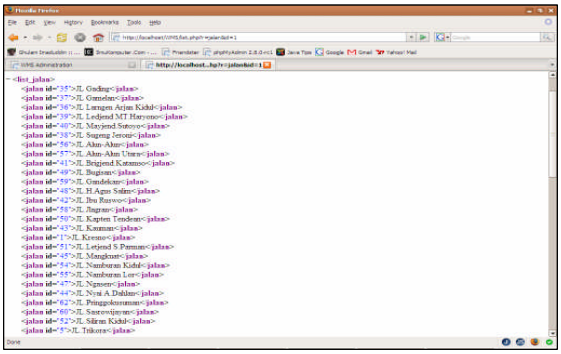
Proses ini akan menghasilkan daftar daerah yang ada dalam *database*. Untuk mengakses halaman ini, klien dapat me-*request* halaman ini dengan alamat **[alamat_server]/[skrip_path]/list.php**. Dalam pengujian, digunakan *server* lokal. Jadi, untuk mengakses fitur ini, digunakan alamat http://localhost/WMS/list.php. Contoh tampilan dari halaman ini (pada *browser*) seperti pada gambar 13.



Gambar 13. Tampilan data daerah berformat XML pada browser

IV.I.IV *DOWNLOAD* DAFTAR JALAN

Proses ini akan menghasilkan daftar jalan yang ada dalam *database* berdasarkan id daerah yang diminta. Untuk mengakses halaman ini, klien dapat me-*request* halaman ini dengan alamat : **[alamat_server] /[skrip _path]/list.php ?r=jalan&id=[id_daerah]**. Dalam pengujian, digunakan *server* lokal. Jadi, untuk mengakses fitur ini, digunakan alamat http://localhost/WMS/list.php ?r=jalan&id=1. Contoh tampilan dari halaman ini (pada *browser*) seperti pada gambar 14.

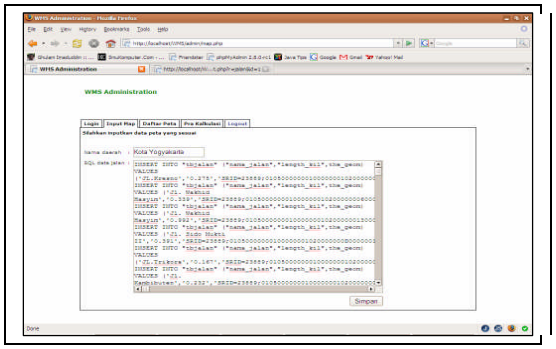


Gambar14. Tampilan XML data jalan pada browser

IV.I.V PROSES *LOGIN* ADMINISTRATOR

Proses ini digunakan untuk masuk kedalam menu administrasi data peta. *User* harus memasukkan *username* dan *password* yang sesuai untuk dapat masuk. **Proses *Input* Peta**

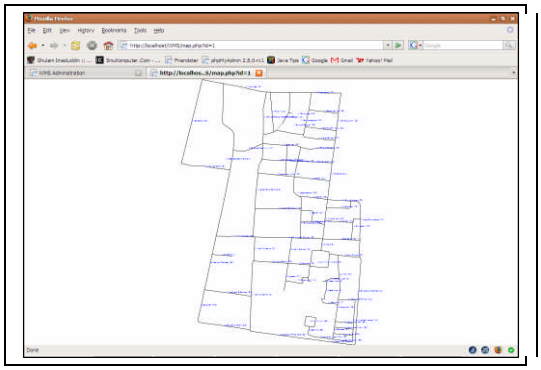
Proses ini digunakan untuk menginputkan data peta jalan ke dalam *database*. Admin harus memasukkan data daerah dan data peta jalan dalam format SQL yang sesuai. Tampilan dari halaman login ini seperti pada gambar 15.



Gambar15. Tampilan halaman input peta

IV.I.VI PROSES LIHAT DAFTAR PETA

Proses ini mengambil semua data daerah dari *database*. Ketika *link* daerah diklik, maka akan ditampilkan peta daerah tersebut, seperti pada gambar 16.



Gambar 16. Tampilan peta yang dipilih

IV.I.VII FUNGSI-FUNGSI LAIN YANG DIGUNAKAN

Dalam proses-proses diatas, digunakan fungsi-fungsi pendukung untuk membantu proses tersebut. Fungsi-fungsi pendukung yang dibuat adalah :

1. Fungsi untuk membuat *header* dari *XML*
2. Fungsi untuk membuat *DOCTYPE* dari *SVG*
3. Fungsi untuk membuat *Header* dari *SVG*
4. Fungsi untuk mengambil *viewbox* dari peta
5. Fungsi untuk menentukan jarak antara dua titik dalam peta
6. Fungsi untuk menghitung bobot jalan pada peta
7. Fungsi untuk mengambil skala dari peta
8. Fungsi untuk menggambar *path* berdasarkan dua titik yang ditentukan
9. Fungsi *Floyd-Warshall* untuk menentukan jarak terpendek antara dua titik

   Fungsi ini akan mencari semua pasangan node yang telah terboboti. Dari pasangan node tersebut akan dicari setiap jarak minimum dan rute-nya. Setelah itu, jarak dan rutenya akan dijadikan nilai kembalian dari fungsi ini. Adapun kode dari fungsi ini ditunjukkan pada gambar 17.

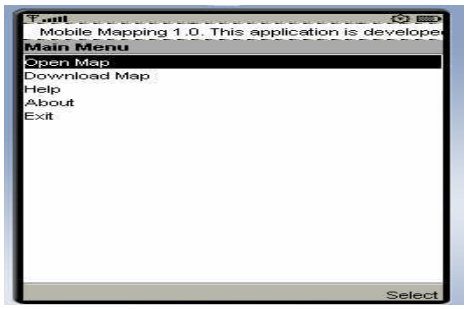
```
function fw($graph,$s,$f,$infinity){
    //Initialization
    $dist = $graph;
    $pred = array();
    $node = array();

    if($s==$f){
        return array("length"=>$infinity,"path"=>array());
    }
    foreach($graph as $i=>$x){
        $node[] = $i;
    }

    foreach($node as $i){
        foreach($node as $j){
            $pred[$i][$j] = null;
            if ($dist[$i][$j] > 0 && $dist[$i][$j] < $infinity)
                $pred[$i][$j] = $i;
        }
    }

    foreach($node as $k){
        foreach($node as $i){
            foreach($node as $j){
                if ($dist[$i][$j] > $dist[$i][$k] + $dist[$k][$j]){
                    $dist[$i][$j] = $dist[$i][$k] + $dist[$k][$j];
                    $pred[$i][$j] = $pred[$k][$j];
                }
            }
        }
    }

    $stop = false;
    $pred = $pred[$s];
    $jalur = array();
    while(!$stop){
        if($s == $pred[$f]){
            $stop = true;
        }else{
            $jalur[] = $f;
            $f = $pred[$f];
        }
    }

    $jalur = array_reverse($jalur);
    array_unshift($jalur,$f);
    array_unshift($jalur,$s);
    return array( "length"=>$dist[$s][$f], "path"=>$jalur);
}
```

IV.II IMPLEMENTASI PROGRAM SISI *CLIENT* (J2ME)

Aplikasi yang telah dibangun terdiri atas sembilan belas *class* dengan *class* utama adalah *class MobileMapping*.

IV.II.I MENU UTAMA

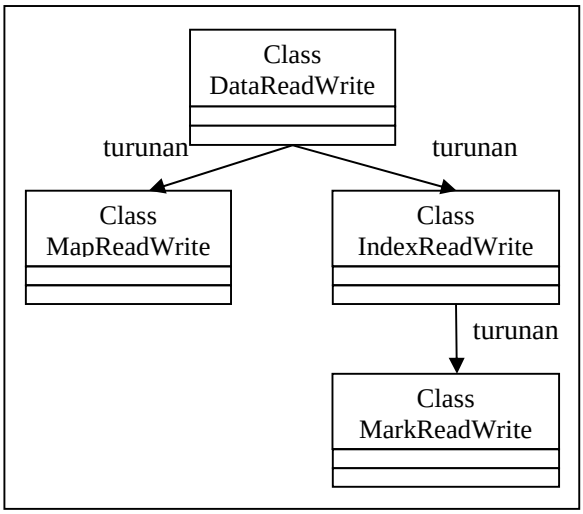
Class yang berfungsi untuk menampilkan menu utama adalah *class MobileMapping*. Class tersebut merupakan class yang pertama kali dieksekusi ketika aplikasi dijalankan. Class tersebut merupakan turunan dari class MIDlet dan mengimplementasikan *Interface Command Listener* yang akan menampilkan lima menu utama dari aplikasi yang akan dibangun yaitu Open Map, Download, Help, About, dan Exit.



Gambar 18. Tampilan menu utama aplikasi Mobile Mapping pada emulator JWTK 2.5beta

IV.II.II PENYIMPANAN DATA

Fungsi penyimpanan pada aplikasi yang dibuat melibatkan empat *class* yaitu *class DataReadWrite*, *MapReadWrite*, *IndexReadWrite* dan *MarkReadWrite*. Hirarki dari *class-class* tersebut seperti pada gambar 19.



Gambar 19. Hirarki class-class untuk fungsi penyimpanan

IV.II.III PENAMPILAN *MAP* YANG ADA PADA *RECORD STORE*

IV.II.III.I CLASS MAP LIST SCREEN

Class ini mengimplementasikan *interface CommandListener* dan berfungsi untuk menampilkan daftar nama-nama *map* yang terdapat dalam *record store.*

Method MapList() memberikan nilai kembalian *List*. *Method* membuat obyek *List* dengan item berupa nama semua *map* yang disimpan di *record store* dan tiga buah *Command* dengan label *back*, *view* dan *delete*.

```
public List MapList() {
   list = new List("Map List",
Choice.IMPLICIT);
   numRecord = mapRW.getNum();
  if (numRecord != 0){
      mapName = new String[numRecord];
      mapName = mapRW.getMapNames();
         mapID = new int[mapRW.getNum()];
         mapID = mapRW.getMapIDs();
      if (mapName != null){
         for (int
i=1;i<=mapRW.getNum();i++){
            list.append(mapName[i],null);
         }
      }
   }
   list.addCommand(backCommand);
   list.addCommand(viewCommand);
   list.addCommand(deleteCommand);
   list.setCommandListener(this);
   return list;
}
```

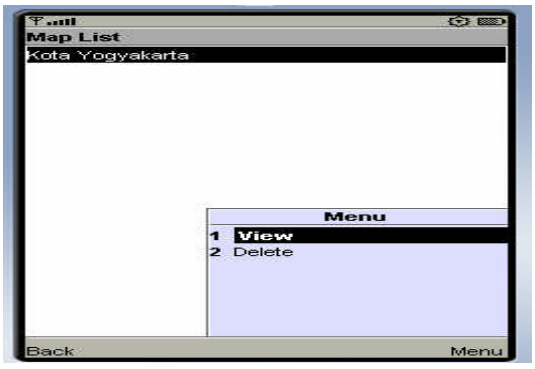
Gambar 20. Method MapList() pada class MapListScreen

```
public void commandAction(Command c,
Displayable d){
   if (c==backCommand){
      midlet.mainMenuScreenShow();
   } else if (c==viewCommand) {
      if (list.getSelectedIndex() >=0){
         mapID = new
int[mapRW.getNum()];
         mapID = mapRW.getMapIDs();
         int ID =
mapID[list.getSelectedIndex()+1];
         String mapName =

list.getString(list.getSelectedIndex());
         idDaerah =
mapRW.getIdDaerah(ID);
         mapScreen = new
MapCanvasFromStore(midlet, ID,
            idDaerah, mapName,
mapRW.getMapFile(ID));

midlet.display.setCurrent(mapScreen);
      }
   } else if (c==deleteCommand){
      alert = new Alert ("Confirmation");
      alert.setString("Are you sure to delete
this map?");
      alert.addCommand(yesCommand);
      alert.addCommand(noCommand);
      alert.setTimeout(alert.FOREVER);
      alert.setCommandListener(this);
      midlet.display.setCurrent(alert);
   } else if (c==yesCommand){
      if (list.getSelectedIndex() >=0){
         mapID = new
int[mapRW.getNum()];
         mapID = mapRW.getMapIDs();
         int ID =
mapID[list.getSelectedIndex()+1];
         mapRW.deleteMap(ID);
         IndexReadWrite indexRW = new

IndexReadWrite(list.getString(list.getSelect
edIndex()));
         indexRW.deleteRecStore();
         MarkReadWrite markRW = new

MarkReadWrite(list.getString(list.getSelecte
dIndex()));
         markRW.deleteRecStore();

midlet.display.setCurrent(MapList());
      }
   }else if (c==noCommand){
      midlet.display.setCurrent(MapList());
   }
}
```

Gambar 21. Method commandAction(Command c, Displayable d) pada class MapListScreen

Method commandAction(Command c, Displayable d) yang merupakan implementasi dari *class abstract CommandAction* didefinisikan tindakan yang akan dilakukan jika ada *Command* yang dipilih oleh *user*.

Jika dipilih *viewCommand* maka *map* yang dipilih akan ditampilkan. Jika *deleteCommand* dipilih maka akan dilakukan pemanggilan *method* deleteMap(ID) dari *class MapReadWrite* yang berarti *map* yang dipilih dihapus dari *record store*.

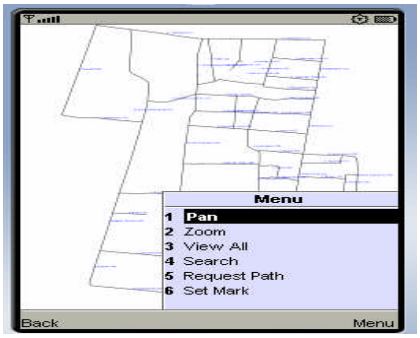


Gambar 22. Tampilan daftar map yang disimpan, pada emulator JWTK2.5beta

IV.II.III.II CLASS MAP CANVAS

Class MapCanvas merupakan turunan dari *class Canvas* dan mengimplementasikan *interface CommandListener*. *Class* ini berfungsi untuk menampilkan *map* dengan format *SVGT* dan menampilkan perintah-perintah yang didefinisikan.

Gambar 23 merupakan tampilan *map* yang datanya diambil dari *record store* dengan enam pilihan menu. Emulator yang digunakan adalah pada emulator *J2ME wireless toolkit 2.5 beta*.



Gambar 23. Tampilan halaman view map pada emulator JWTK2.5beta

IV.II.IV PAN (PENGGESERAN *MAP*)

Class MapPan berfungsi untuk menggeser map yang ditampilkan. *Class* ini mengimplementasikan *interface SVGEventListener* yang mengirim *event* yang terjadi pada *map SVGT*, ke aplikasi.

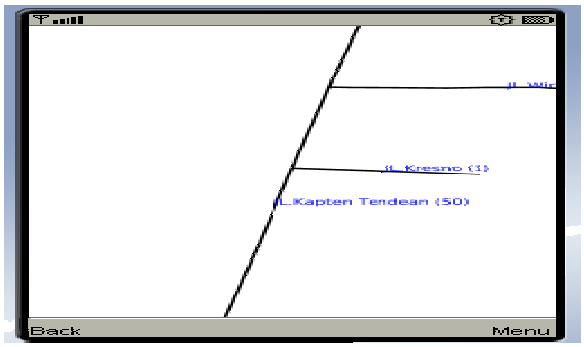
IV.II.V ZOOM (PERBESARAN *MAP*)

Class MapZoom berfungsi untuk memperbesar atau memperkecil *map* yang ditampilkan. *Class* ini mengimplementasikan *interface SVGEventListener* seperti pada *class MapPan*.

IV.II.VI SEARCHING (PENCARIAN) TEMPAT TERTENTU PADA *MAP* (*CLASS SEARCH SCREEN*)

Class yang mengimplementasikan *interface CommandListener* ini menampilkan daftar tempat atau jalan yang terdapat pada *map* yang ditampilkan. Data yang ditampilkan diambil dari tabel jalan yang namanya sesuai dengan nama *map* yang bersangkutan.

Gambar2 4. a Tampilan Search Path pada emulator JWTK2.5beta

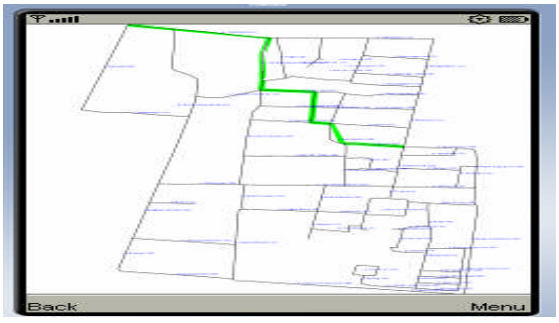


Gambar2 4. b Tampilan Search Path pada emulator JWTK2.5beta

IV.II.VII *REQUETH PATH*

Proses *request path* melibatkan dua *class*, yaitu *class PathCanvas* dan *PathScreen* dan sub proses yang dilakukan adalah sebagai berikut :

1. Melakukan *request* daftar jalan ke *server* mengirimkan ID map yang bersangkutan.
2. Melakukan parsing dokumen XML yang berisi daftar jalan yang merupakan respon dari server.
3. Melakukan *request path* jarak terpendek ke *server* dengan mengirim id jalan asal dan tujuan.
4. Menggabungkan data sebagai respon dari *server* dengan map *SVGT* yang bersangkutan.
5. Menampilkan hasil gabungan di atas ke layar.



Gambar 25. Tampilan map hasil RequestPath pada emulator JWTK2.5beta

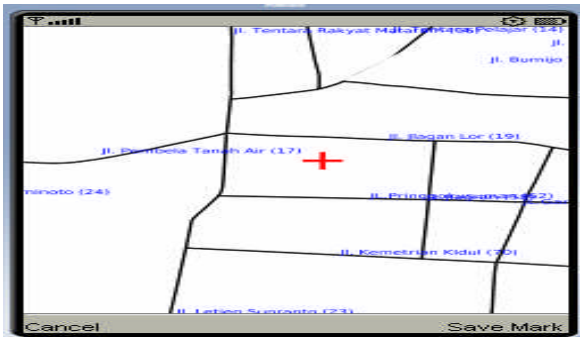
IV.II.VIII FUNGSI *SET MARK*

Pada aplikasi yang dibuat juga terdapat fungsi *set mark* yaitu fungsi agar *user* bisa menandai titik-titik tertentu. Satu *map* akan berelasi dengan satu *record store* yang menyimpan titik-titik tersebut.

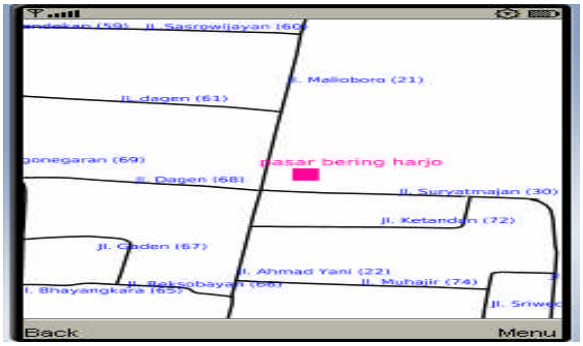
IV.II.VIII.I CLASS MAP MARK

```
public MapMark(SVGImage inImage, SVGAnimator inAnimator, MapCanvas mapCanvas) {
   this.svgMap = inImage;
   this.animator = inAnimator;
   this.canvas = mapCanvas;
   pan = new MapPan(inImage, inAnimator, mapCanvas);
   this.document = svgMap.getDocument();
   this.root = (SVGSVGElement) document.getDocumentElement();
   this.scale =

root.getBBox().getWidth()/root.getScreenBBox().getWidth();
   createPointer();
   root.appendChild(pointer);
}
```

Gambar 26. Constructor pada class MapMark



Gambar 27. Tampilan pointer set mark pada emulator JWTK2.5beta

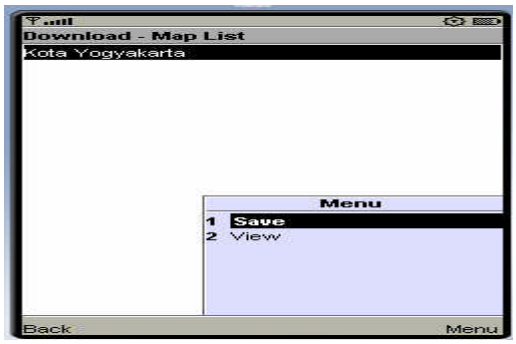


Gambar 28. Tampilan pointer hasil set mark pada emulator JWTK2.5beta

IV.II.IX FUNGSI *DOWNLOAD MAP*

IV.II.IX.I *CLASS DOWNLOAD SCREEN*

Class ini juga mengimplementasikan *interface CommandListener*. Jika dieksekusi, maka *class* ini akan mengirim permintaan data *map* ke *server*. *Server* akan merespon dengan mengirimkan daftar map yang disediakan *server* dengan format *XML*. *Class* ini akan melakukan *parsing XML* dan menampilkan data tersebut ke layar. Kemudian pilihan dari *user* akan dikirim ke *server* dan *server* akan merespon dengan mengirimkan *map* sesuai dengan permintaan.



Gambar 29. Tampilan halaman download pada emulator JWTK2.5beta

IV.II.IX.II *CLASS MAP CANVAS FROM URL*

Class MapCanvasFromURL merupakan turunan dari *class MapCanvas*. *Class* ini mengimplementaiskan *interface Runnable* untuk menghindari deadlock pada saat melakukan koneksi *HTTP*.

V. PENUTUP

Dari penelitian ini dapat ditarik beberapa kesimpulan :

Telah berhasil dikembangkan aplikasi *WMS* (*Web Map Services*) menggunakan *PHP* dan database *PostgreSQL* sebagai sebuah *server* yang dapat menghasilkan peta berformat *SVGT dan client mobile mapping* menggunakan teknologi *J2ME* dan *SVGT* Aplikasi yang dibuat mempunyai beberapa fungsi utama, yaitu *download map* dengan format *SVGT* dari server, menyimpan *map* ke dalam *record store*, menampilkan *map* dan juga melakukan fungsi *pan* dan *zoom* terhadap *map*. Fungsi lainnya adalah *searching* atau mencari titik tertentu pada

map sesuai dengan permintaan *user*, *request path* ke *server* untuk mendapatkan jalur terpendek antara dua titik, dan *set mark* atau menyimpan titik-titik pada peta yang telah diberi tanda oleh *user*.

Telah berhasil dikembangkan sebuah server yang dapat menghitung dan mencari rute terpendek berdasarkan dua buah tempat yang ditentukan berdasarkan request dari klien, dan dapat menyajikannya dalam format SVGT.

Format data SVGT dan XML cocok diterapkan pada aplikasi WMS ini karena ukuran datanya yang kecil, terstruktur, dan akurat.

VI. DAFTAR PUSTAKA

[1] Hui, L., 2006, *Design and Implement a Cartographic Client Application For Mobile Devices using SVG Tiny and J2ME*, Stuttgart.

[2] Internetnews,2006, *Symbian at a Mobile Loss* ,http://www.internetnews.com/wireless/article.php/3584431, 20 Februari 2006.

[3] Januar, A.M., 2003, *Pengantar Scalable VectorGraphics(SVG)*,http://ilmukomputer.com, 12 Oktober 2006.

[4] Liu, J., 2002, *Mobile Map: A Case Study in the Design & Implementation of a Mobile Application*, Carleton University, Canada

[5] Luqun, L. and Minglu, L. 2004, *A Research on Development of mobile GIS architecture*, China.

[6] Ramsey, P., 2005. *PostGIS Manual*. http://postgis.refractions.net/docs/postgis.pdf. Tanggal download : 28 September 2006.

[7] Solyman, A..A., 2006, *Mobile map - Technology for Application*, Deutsche Gesellschaft Fuer Technische Zusammenarbeit (GTZ), http://www.gisdevelopment.net/technology/mobilemapping/soly_2.htm, 20 September 2006

[8] W3C, *Scalable Vector Graphics (SVG)*, http://www.w3.org/TR/SVGMobile/, 12 Oktober 2006.

[9] Wender, J.and Ronayne,D.,2003, *Mobile SVG*.

[10] Wiki,2006 ,JSR226,http://www.wiki.svg.org/jsr_226, 20 Desember 2006

[11] Wikipedia,2006.*Post GIS*. http://en.wikipedia.org/wiki/PostGIS. Tanggal akses : 29 September 2006.

[12] Wikipedia, 2006. *Floyd-Warshall Algorithm*. http://en.wikipedia.org/wiki/Floyd-Warshall_algorithm. Tanggal Akses : 2 Januari2007