Seminar Nasional Ilmu Komputer (SNIK 2016) - Semarang, 10 Oktober 2016
ISBN: 978-602-1034-40-8

Penerapan Algoritma *Dijkstra* pada Aplikasi Pencarian Rute Bus Trans Semarang

Dwi Ardana[1], Ragil Saputra[2]

[1,2]Departemen Ilmu Komputer/Informatika, FSM, Universitas Diponegoro
Email: [1]dwiardana13@gmail.com,[2]ragil.saputra@undip.ac.id

Abstrak

Pemerintah Kota semarang menyediakan fasilitas Bus Trans Semarang sebagai solusi untuk memenuhi kebutuhan transportasi umum yang besar dan sebagai upaya penanggulangan kemacetan di daerah kota metropolitan ini. Akan tetapi, calon penumpang menghadapi kesulitan untuk mendapatkan informasi rute bus yang akan diambil untuk menuju lokasi yang ditentukan, karena informasi yang tersedia masih berupa informasi statis berupa poster tempel. Pada penelitian ini, kami mengajukan solusi aplikasi digital untuk pencarian rute Bus Trans Semarang menggunakan Algoritma *Dijkstra*. Algoritma *Dijktra* digunakan untuk menentukan rute dan lokasi perpindahan koridor atau *transfer point*. *Waterfall* dengan pendekatan *Object Oriented* digunakan dalam proses pengembangan perangkat lunak. Dalam aplikasi ini, *Google Maps* API digunakan sebagai data spasial, sedangkan data non spasial berupa informasi detail *shelter* dan koridor. Calon penumpang Bus Trans Semarang dapat memanfaatkan aplikasi ini dengan memasukkan informasi lokasi yang ingin dituju untuk mendapatkan rute. Kemudian aplikasi akan menampilkan peta rute yang akan dilalui dari titik awal menuju lokasi tujuan. Aplikasi juga dilengkapi fitur melihat jadwal, melihat rute, penentuan rute dan lokasi *shelter*. Berdasarkan pengujian, aplikasi tersebut dapat memberikan informasi lokasi perpindahan koridor pada semua rute perjalanan yang melewati semua lokasi *transfer point*.
Kata Kunci: Algoritma *dijkstra*, rute, bus, data spasial, non spasial

Abstract

Municipal government of Semarang provides a Bus Trans Semarang as a solution to accommodate a facility's requirement for large public transportation and to reduce a congestion in this metropolitan city. However, the passengers faced difficulties to obtain an information of the bus route to or from the specific destination, because the information provided was in the form of static information such as poster's outboard. In this paper, we propose solutions for route searching of Trans Semarang Bus in digital applications using Dijkstra algorithm. Dijktra algorithm was used to determine the best route and migration corridors location or transfer point. Waterfall with Object Oriented approach was used in the software-development process. In the applications, the Google Maps API is used as a spatial data, while the non-spatial data is the form of shelter, and corridors detail information. Trans Semarang Bus passengers can utilize this application by input the destination information to get the route. The application will display a route map that will be passed from the starting point to the destination location. Application also has features to get information about the schedule, the route, determining routes and shelter locations. Based on testing experiment, the app can provide all migration corridor routes that passes for all transfer point locations.
Keyword: *Dijkstra algorithm, route, bus, spatial, non-spatial*

1. PENDAHULUAN

Kemajuan transportasi kini menjadi pendukung aktifitas utama dalam kegiatan manusia. Berbagai alat transportasi kini banyak disediakan, terlebih transportasi darat yang menggunakan jalan raya sebagai media transportasi untuk mencapai tujuan. Seiring berkembangnya alat transportasi maka jalan yang dibutuhkan akan semakin banyak. Kota Semarang sebagai salah satu kota metropolitan yang memiliki pengguna transportasi dalam jumlah besar sehingga menjadi salah satu kota padat di Indonesia. Pemerintah Kota Semarang memberikan fasilitas Bus Trans Semarang untuk para penggunanya sebagai upaya penanggulangan kemacetan yang terjadi pada Kota Semarang.

Bus Rapid Transit (BRT) atau *busway* merupakan bus dengan kualitas tinggi yang berbasis sistem transit yang cepat, nyaman, dan biaya murah untuk mobilitas perkotaan dengan menyediakan jalan untuk pejalan kaki, infrastrukturnya, operasi pelayanan yang cepat dan seiring, perbedaan dan keunggulan pemasaran dan layanan kepada pelanggan. Dalam setiap jalur BRT sendiri terdapat beberapa koridor untuk tujuan tertentu, dan *shelter* sebagai tempat pemberhentian bus [1].

Bus Trans Semarang memiliki jalur untuk beroperasi, tidak seperti bus umum pada lainya sehingga Bus Trans Semarang memiliki koridor atau pemberhentian sesuai dengan aturan jalur beroperasi.

ISBN: 978-602-1034-40-8

Sistem Informasi Geografis (SIG) merupakan suatu teknologi informasi yang menggabungkan pengumpulan data, teknologi sistem basis data dan sistem komputer yang berbasis keruangan untuk memperoleh informasi [2]. Sistem Informasi Geografis (SIG) dapat digunakan untuk membantu penggunanya mendapatkan informasi khususnya mengenai jalur pada jalan raya. Dengan menggunakan Sistem Informasi Geografis (SIG), masalah mengenai informasi jalur operasi Bus Trans Semarang dapat ditangani.

Dijkstra merupakan salah satu algoritma yang efektif dalam memberikan lintasan terpendek dari suatu lokasi ke lokasi yang lain. Prinsip algoritma *Dijkstra* adalah dengan pencarian dua lintasan yang paling kecil. Algoritma *Dijsktra* memiliki iterasi untuk mencari titik yang jaraknya dari titik awal adalah paling pendek [3].

Beberapa penelitian telah mengembangkan Sistem Informasi Geografis (SIG), salah satunya dilakukan oleh Priyantoro, penelitian ini berhasil membangun aplikasi pencarian rute terbaik berdasarkan jadwal keberangkatan, jarak terdekat dan jumlah pergantian armada berbasis *Google Map* pada sistem operasi Android dengan studi kasus Trans Jogja [4]. Selain itu pada penelitian yang dilakukan oleh Tirastitam mengenai sistem perencanaan transportasi umum berbasis algoritma *Dijkstra* dengan studi kasus Bangkok Metropolitan Area dengan berfokus pada bus, BTS dan MRT jadwal/rute untuk memberikan informasi kepada penumpang. Penumpang dapat memilih jalan dan rute dengan mudah menggunakan *Dijkstra* STAR Algoritma, dan diperoleh tarif perjalanan [5].

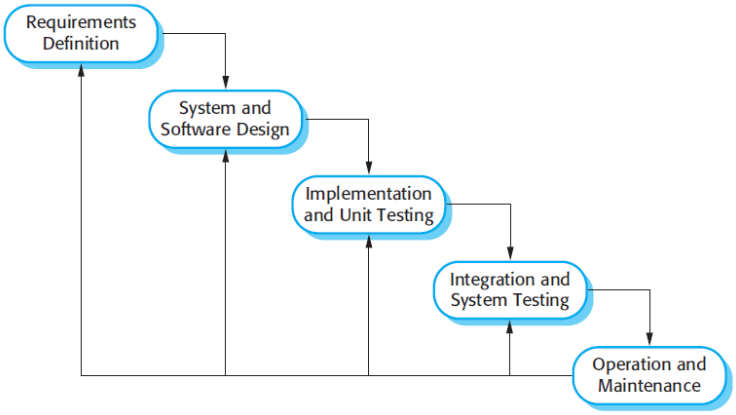
Berdasarkan permasalahan tersebut, maka pada penelitian ini akan dibangun aplikasi pencarian rute Bus Trans Semarang, dan menerapkan metode *Dijkstra* pada perpindahan jalur pada *transferpoint*. Tujuan yang ingin dicapai dalam penelitian ini adalah aplikasi yang mampu memberikan informasi rute dan penjadwalan Bus Trans Semarang. Sedangkan manfaatnya antara lain sebagai sarana yang dapat membantu calon pengguna Bus Trans Semarang untuk dapat mengetahui koridor-koridor yang ada pada Kota Semarang dan dapat menentukan pilihan koridor tujuan yang dituju dengan pasti. Serta mampu memberi informasi titik-titik koridor Bus Trans Semarang yang sesuai koridor yang tersedia.

2. METODE

2.1 Model Proses

Proses perangkat lunak (*software process*) merupakan serangkaian kegiatan dan hasil yang berhubungan dengannya, yang menuju pada dihasilkannya produk perangkat lunak. Model proses perangkat lunak (*software process model*) merupakan deskripsi yang disederhanakan dari proses perangkat lunak yang dipresentasikan dengan sudut pandang tertentu [6].

Model *waterfall* merupakan salah satu metode pengembangan perangkat lunak yang digunakan untuk ketelitian. Disebut dengan *waterfall* karena pada proses ini pengembangan dilakukan dengan aturan jika satu *requirement* belum selesai maka tidak bisa maju ke *requirement* selanjutnya dan juga ditengah pengerjaan tidak bisa diberikan *requirement* lagi. Fase-fase dalam *waterfall* menurut referensi Sommerville ditunjukkan pada Gambar 1 [6].



Gambar 1. Model w*aterfall*

ISBN: 978-602-1034-40-8

2.2 *Object Oriented Programming* (OOP)

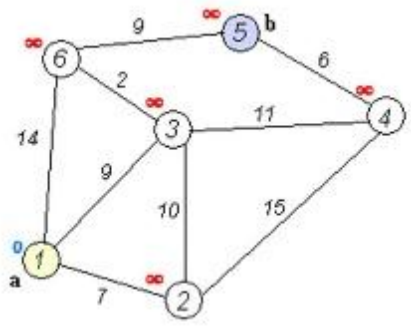
Object Oriented Programming (OOP) merupakan paradigma pemrograman yang berorientasikan kepada objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek. Bandingkan dengan logika pemrograman terstruktur. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya. Model data berorientasi objek dikatakan dapat memberi fleksibilitas yang lebih, kemudahan mengubah program, dan digunakan luas dalam teknik piranti lunak skala besar [7].

2.3 *Unified Modeling Language* (UML)

Unified Modeling Language(UML) adalah bahasa standar yang digunakan untuk menulis *blueprint* perangkat lunak. UML dapat digunakan untuk memvisualisasi, menspesifikasikan, membangun dan melakukan dokumentasi artifak dari sistem perangkat lunak.UML hanya merupakan sebuah bahasa, sehingga UML hanya salah satu bagian dari metode pengembangan perangkat lunak. UML terdiri atas 3 *building block*, yaitu *things*, *relationships*, *dan diagrams* [8].

2.4 Algoritma *Dijkstra*

Algoritma *Dijkstra* adalah sebuah algoritma *greedy* yang dipakai dalam memecahkan permasalahan jarak terpendek untuk sebuah graf berarah dengan bobot-bobot sisi (*edge)* yang bernilai tak negatif [9]. Ide dasar algoritma *Dijkstra* sendiri ialah pencarian nilai *cost* yang terdekat dengan tujuan yang berfungsi pada sebuah graf berbobot, sehingga dapat membantu memberikan pilihan jalur. Pada Algoritma *Dijkstra*, *node* digunakan karena Algorima *Dijkstra* menggunakan *graph* berarah untuk penentuan rute lintasan terpendek. Algoritma ini bertujuan untuk menemukan jalur terpendek berdasarkan bobot terkecil dari satu titik ke titik lainnya. Misalkan titik mengambarkan gedung dan garis menggambarkan jalan, maka Algoritma *Dijkstra* melakukan kalkulasi terhadap semua kemungkinan bobot terkecil dari setiap titik. Pada Gambar 2 disajikan contoh graf dengan bobotnya dalam menentukan jalur menggunakan Algoritma *Dijkstra.*



Gambar 2. Contoh menemukan jalur menggunakan algoritma *Dijkstra*

Pertama-tama tentukan titik mana yang akan menjadi *node* awal, lalu beri bobot jarak pada *node* pertama ke *node* terdekat satu per satu, Algoritma *Dijkstra* akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap. Urutan logika dari Algoritma *Dijkstra* sebagai berikut.

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 *node* 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 selanjutnya dan lanjutkan dengan kembali ke *step* 3.

ISBN: 978-602-1034-40-8

2.5 Metode *Haversine*

Metode *Haversine* digunakan untuk menghitung jarak antara titik di permukaan Bumi menggunakan garis lintang (*longitude*) dan garis bujur (*latitude*) sebagai variabel *input*-an. *Haversine* formula adalah persamaan penting pada navigasi, memberikan jarak lingkaran besar antara dua titik pada permukaan bola (Bumi) berdasarkan bujur dan lintang. Dengan mengasumsikan bahwa bumi berbentuk bulat sempurna dengan jari-jari r 6.367,45 km, dan lokasi dari 2 titik di koordinat bola (lintang dan bujur) masing-masing adalah γ_1, ϕ_1, dan γ_2, ϕ_2, maka rumus *Haversine* dapat ditulis dengan rumus (1) [10].

$$d = 2r \sin^{-1}\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\, sin^2\left(\frac{\gamma_2 - \gamma_2}{2}\right)}\right) \quad (1)$$

Keterangan:
ϕ = *Longitude*
γ = *Latitude*
d = Jarak
r = Radius Bumi = 6371 km
1 derajat= 0.0174532925 radian

2.6 *Google Maps*

Google Maps adalah jasa peta gratis dan *online* disediakan oleh *google* yang dapat ditemukan di http://maps.google.com. Pada situs tersebut kita dapat melihat informasi geografis pada hampir semua wilayah di muka bumi. Layanan ini interaktif, karena di dalamnya peta dapat digeser sesuai keinginan pengguna, mengubah tingkat *zoom*, serta mengubah tampilan peta. *Google Maps* juga menawarkan peta yang dapat diseret dan gambar satelit untuk seluruh dunia, serta menawarkan rute perjalanan. *Google Maps* dibuat dengan menggunakan kombinasi dari gambar peta, *database*, serta objek-objek interaktif yang dibuat dengan bahasa pemrograman HTML, *Javascript*, dan AJAX, serta beberapa bahasa pemrograman lainnya. Seluruh citra yang ada diintegrasikan ke dalam suatu *database* pada *google server*. Bagian-bagian gambar peta yang merupakan gabungan dari gambar-gambar yang berukuran 256 x 256 pixel [11].

3 HASIL DAN PEMBAHASAN

Aplikasi Pencarian Rute Bus Trans Semarang ini dibangun untuk diimplementasikan pada Dinas Perhubungan dengan berbasis *web* yang digunakan untuk memberikan informasi pada calon pengguna Bus Trans Semarang untuk dapat mengetahui rute perjalanan, jam keberangkatan dan jalur yang dilalui. Sistem ini dapat memberikan informasi kepada calon pengguna berupa letak *shelter*, jalur yang dilewati dan rute perjalanan Bus Trans Semarang.

3.1 Deskripsi Umum

Bus Trans Semarang memiliki jalur untuk beroperasi, tidak seperti bus umum pada lainya sehingga Bus Trans Semarang memiliki koridor atau pemberhentian sesuai dengan aturan jalur beroperasi. Akan tetapi, belum ada sarana untuk mengetahui jalur operasi tempat pemberhentian penumpang sesuai dengan tujuan. Untuk memberikan informasi pada penumpang mengenai jalur operasi Bus Trans Semarang tanpa terkendala batas ruang, Pemerintah Kota Semarang membutuhkan suatu sistem informasi dalam upaya menanggulangi masalah tersebut. Oleh karena itu, aplikasi pencarian rute Bus Trans Semarang dirancang untuk memberikan kemudahan kepada calon pengguna umum informasi jadwal keberangkatan, letak *shelter*, dan rute Bus Trans Semarang.

3.2 Implementasi Algoritma *Dijkstra*

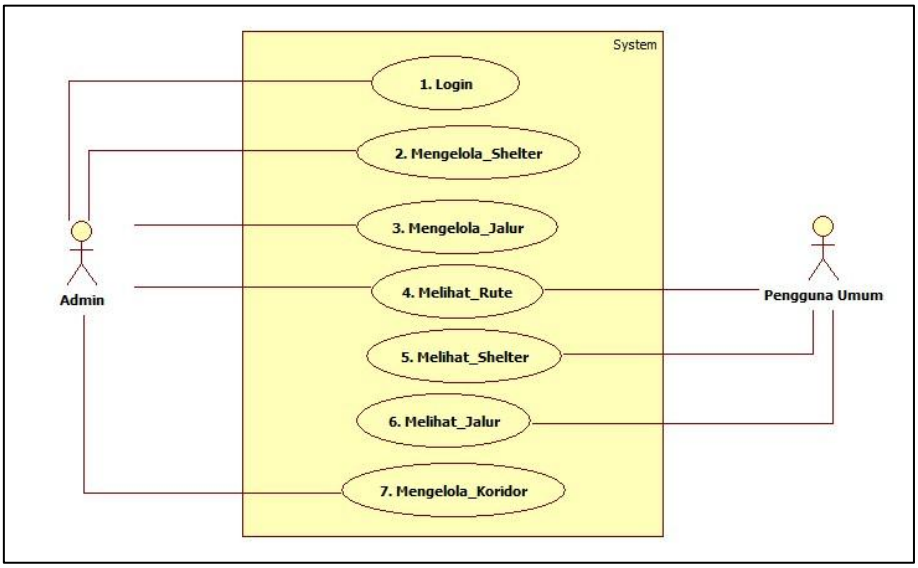
Algoritma *Dijkstra* dalam aplikasi ini diakses untuk mendapatkan rute jalur Bus Trans Semarang pada saat melakukan pindah koridor. *Input*-an untuk pencarian rute ini berupa *node* awal dan *node* tujuan. Pada menu rute, disini pengguna diminta untuk memasukkan titik awal dan titik tujuan secara manual. Pada algoritma *Dijkstra* ini *input*-an awal dan tujuan yang berupa *node* di proses dengan cara menghitung nilai atau *cost* fungsi Dijkstra dan nilai *heuristic* pada *node-node* yang termasuk pada *openset*. *Openset* sendiri adalah *node-node* yang memungkinkan untuk dilalui, setelah dilakukan perhitungan dan didapat *node-node* yang memungkinkan untuk dilalui, setelah dilakukan perhitungan dan didapat *node-node* rute ini ditampilkan pada peta sebagai *path* jalan yang dipilih dengan menggunakan Algoritma *Dijsktra*. Fungsi *Dijkstra* disajikan pada *pseudocode* berikut ini:

ISBN: 978-602-1034-40-8

```
function Dijkstra(Graph, source):

  for each vertex v in Graph:

     dist[v] := infinity

     previous[v] := undefined

  dist[source] := 0

  Q := semua node dalam Graph

  while Q is not empty:

     u := node dalam Q dengan jarak terkecil

     hapus u dari Q

     for each tetangga v dari u:

        alt := dist[u] + dist_between(u, v)

        if alt < dist[v]

     dist[v] := alt

     previous[v] := u

  return previous[ ]
```

3.3 Analisis dan Perancangan

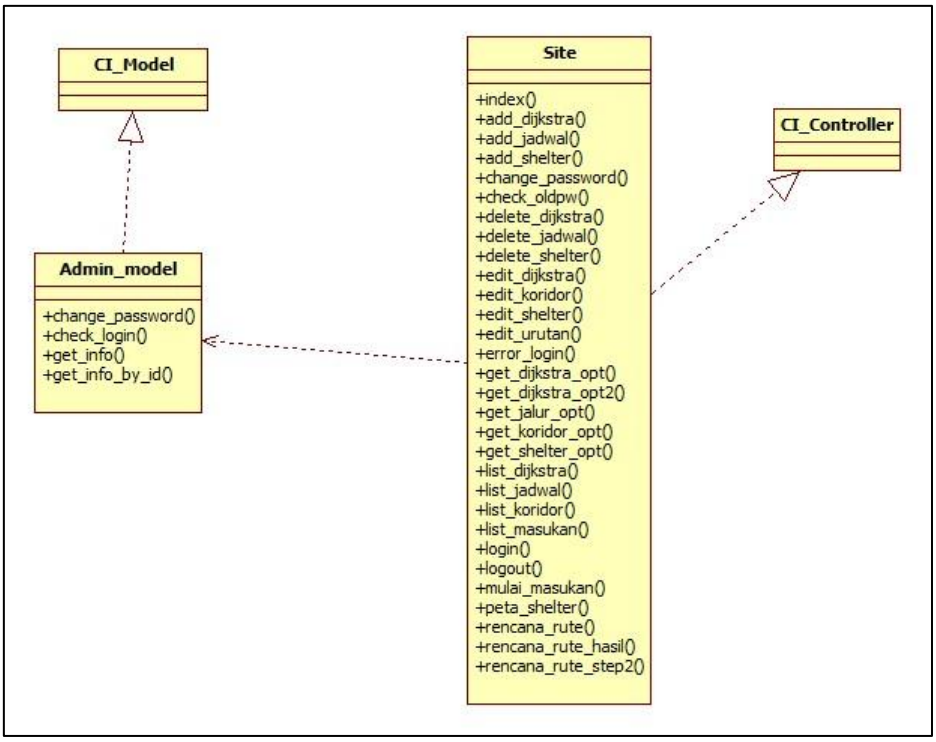
Model *use case* mendifinisikan *requirements* yang dibutuhkan khususnya kebutuhan fungsional. Model *use case* terdiri dari definisi aktor, *use case*, dan relasi antar keduanya. *Use case* diagram sistem menggambarkan fungsi-fungsi yang dapat dilakukan oleh sistem. Batasan sistem, aktor yang terlibat dan *use case* ditentukan berdasarkan kebutuhan fungsional. Berdasarkan aktor yang terlibat dalam sistem dan daftar *use case* yang telah dirancang, maka dapat disimpulkan bahwa terdapat 6 *use case* dan 2 aktor yang terlibat pada perancangan *use* diagram yang digambarkan pada Gambar 3.



Gambar 3.*Use case diagram*

Class Diagram terdiri dari 4 buah *class* yaitu CI_Model, Admin_Model, *Site*, CI_*Controller*. Detail *class* diagram dijabarkan pada Gambar 4.

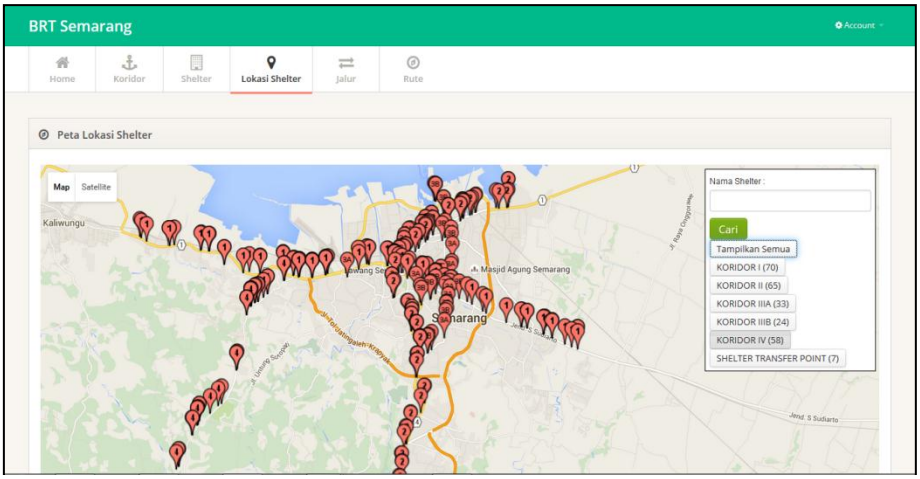
ISBN: 978-602-1034-40-8



Gambar 4. *Class diagram*

3.4 Implementasi Antarmuka

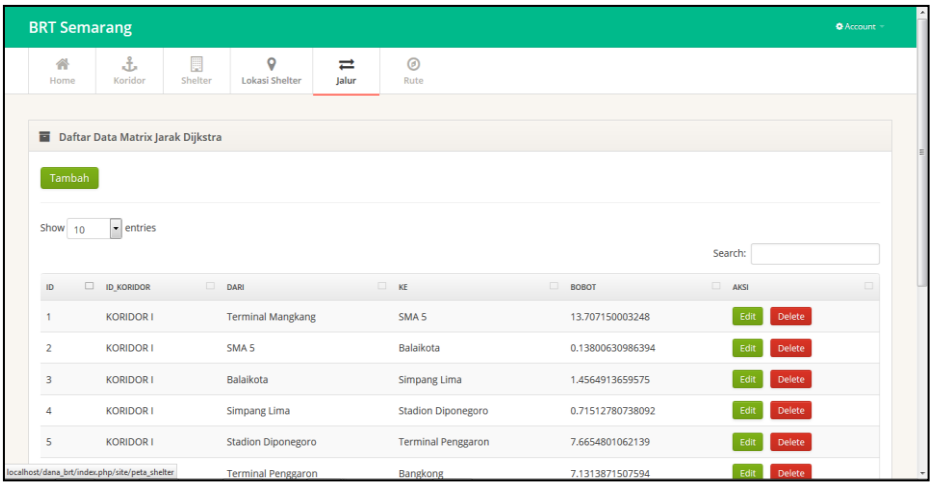
Implementasi Antarmuka Menu Utama disajikan halaman awal Sistem Informasi Jalur Bus Trans Semarang. Implementasi Antarmuka Menu Utama ditunjukan pada Gambar 5.



Gambar 5. Implementasi aplikasi

Pada halaman utama aplikasi terdapat beberapa menu, diantaranya Koridor, *Shelter* Lokasi *Shelter* dan Jalur. Pada menu Koridor disajikan daftar Koridor I sampai IV dan sebuah *Shelter* Transfer *Point*. Menu jalur menyediakan informasi lokasi awal dan tujuan tiap-tiap koridor, disajikan pada Gambar 6 yang merupakan detail untuk koridor I. Pada bagian menu *Shelter* terdapat jadwal keberangkatan pada masing-masing *shelter*. Semua data bersifat dinamis menyesuaikan kondisi sesungguhnya, dan dilakukan manajemen data oleh admin sistem.

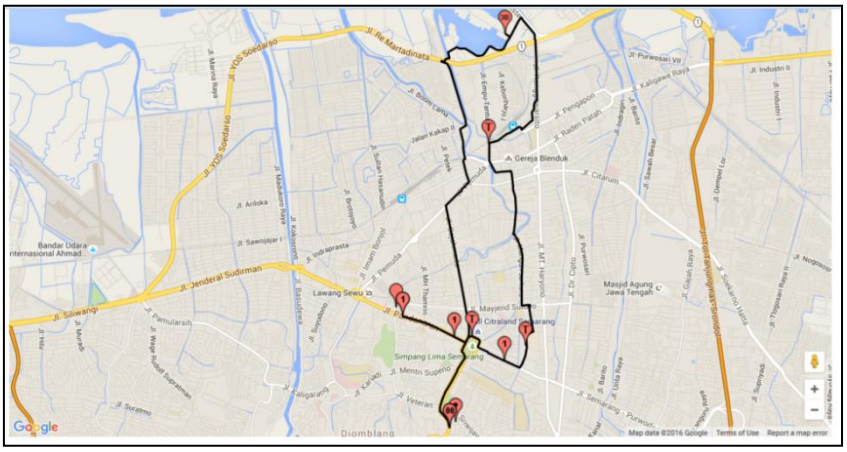
ISBN: 978-602-1034-40-8



Gambar 6. Detail koridor I

3.5 Pengujian Aplikasi Pencarian Rute

Pada langkah pertama pilih titik awal dengan beberapa cara, dapat secara manual memasukkan nama tempat yang dipilih pada kolom nama tempat di menu rute, atau dengan cara mengklik posisi tempat pada *map* yang tersedia. Pada Gambar 7 dilakukan pengujian pemilihan posisi awal dan tujuan, kemudian aplikasi akan menampilkan rute yang dihasilkan.



Gambar 7. Hasil pencarian rute

Dari Gambar 7 diketahui posisi awal pada Pandanaran dan menuju kearah Siranda. Setelah itu didapatkan *longitude* dan *latitude*nya disajikan pada Tabel 1.

Tabel 1. Posisi awal dan tujuan

Lokasi	*Longitude*	*Latitude*
Awal : Pandanaran	110.414108	-6.985853
Akhir : Siranda	110.421489	-6.996544

Proses pencarian rute dimulai setelah diketahui *shelter* terdekat dari titik awal maka akan dilakukan pemberian jalur perjalanan dengan menggunakan *next id*, yaitu setelah id 184 yaitu 187 akan menjadi tujuan selanjutnya hingga membaca transfer *point* selanjutnya, jika jalurnya masih pada koridor 1 maka akan dilanjutkan pada *next id* selanjutnya, sampai *shelter transfer point* stadion diponegoro. Diketahui sebelumnya bahwa *endpoint* kedua adalah 412 yaitu pelabuhan yang terletak pada koridor 3B. Maka *Dijkstra* akan membaca jalur menuju koridor 3B, maka setelah di *shelter transfer point* stadion diponegoro pengguna akan diarahkan pada id koridor 3B setelah *shelter transfer Point* Stadion Diponegoro yaitu *sheltter transfer Point* Stasiun Tawang dengan id 368. Jalur akan terus mengikuti *next*

ISBN: 978-602-1034-40-8

id koridor 3B hingga *shelter* terdekat dengan titik tujuan yaitu *shelter* Siranda 2 dengan id 404. Berikut adalah daftar *path*-nya dapat dilihat pada Tabel 2.

Tabel 2. Tabel hasil *path* dalam perhitungan

Id	Nama *shelter*	Koridor
184	Pandanaran 2/DKK	1
187	Gramedia 2	1
369	Simpang Lima	*Transfer Point*
189	RRI 1	1
371	Stadion Diponegoro	*Transfer Point*
368	Stasiun Tawang	*Transfer Point*
412	Pelabuhan	3B
404	Siranda 2	3B

4 SIMPULAN

Telah berhasil dibuat sebuah Aplikasi Pencarian Rute Bus Trans Semarang dengan memanfaatkan Algoritma *Dijkstra* untuk menentukan *shelter* terdekat dan perpindahan koridor pada *transfer point*. Aplikasi ini dapat menampilkan informasi berkaitan dengan Bus Trans Semarang dan rute perjalanan mulai dari titik awal hingga akhir tujuan.

5 REFERENSI

[1] Barat, D. P. P. J. 2015. *Dishub Provinsi Jawa Barat.* http://dishub.jabarprov.go.id/inc/data/info/566, diakses 19 juni 2016.
[2] Chang, K. 2004. *Introduction to Geographic Information System, Second Edition.* Mc Graw Hill, New York.
[3] Satyananda, D. 2010. *Struktur Data*. Universitas Negeri Malang. Malang.
[4] Priyantoro, A., Mustofa, K. 2014. Pengembangan Aplikasi Pencarian Rute Terbaik Pada Sistem Operasi Android (Studi Kasus Rute Trans-Jogja). *Berkala MIPA*. Vol. 24(1): 72-88.
[5] Tirastittam, P., Waiyawuththanapoom, P. 2014. Public Transport Planning System by Dijkstra Algorithm: Case Study Bangkok Metropolitan Area. *International Journal of Social, Behavioral, Educational, Economic, Business and Industrial Engineering*. Vol. 8(1): 54-59.
[6] Sommerville, I. 2010. *Software Engineering Sixth Edition*. Addison-Wesley, United Kingdom.
[7] Wu, C. T. 2010. *An Introduction to Object-Oriented Programming with Java, Fifth Edition*. Mc Graw Hill, New York.
[8] Booch, G., Rumbaugh, J., Jacobson, I. 1999. *The Unified Modeling Language*. Addion Wesley Longman, Massachusetts.
[9] Munir, R. 2009. *Matematika Diskrit Edisi Ketiga*. Informatika, Bandung.
[10] Chopde, N., Nichat, M. 2013. Landmark Based Shortest Path Detection by Using A* and Haversine Formula. *International Journal of Innovative Research in Computer and Communication Engineering*. Vol.1, Issue 2: 298-302.
[11] Huda, M. 2014. *Pengenalan dan cara kerja Google Maps,* http://www.mkhuda.com. diakses 19 Juni 2016.