APLIKASI PENELUSURAN OBJEK LOKASI DAN JALUR TERDEKAT GEREJA DI KOTA PALEMBANG MENGGUNAKAN ALGORITMA *DIJKSTRA*

(Study Kasus: Wilayah Kota Palembang)

Salomo Silaban[1], A. Haidar Mirza [2], Ahmad Syazili [3]
Fakultas Teknik Ilmu Komputer, Universitas Bina Darma
Email: salib2slb@gmail.com[1], haidarmirza@binadarma.ac.id[2], syazili@binadarma.ac.id[3]

Abstrak

Pencarian rute terpendek *(shortest path)* untuk menemukan tempat lokasi ibadah umat nasrani adalah permasalahan untuk mencari rute minimum dari titik awal ke titik tujuan. Pencarian rute terpendek dilakukan berdasarkan titik awal, titik tujuan, dan jarak tempuh. Penelitian ini dibuat untuk merancang sebuah aplikasi android yang memberikan informasi serta petunjuk arah tempat ibadah di Kota Palembang dengan menerapkan algoritma *dijkstra* dan motode pengembangan aplikasi yakni *Agile Methods* dengan model *eXtreme Programming (XP),* juga memanfaatkan *Geograpihc Information System (GIS)* untuk pemetaan lokasi tempat ibadah. Algoritma *dijkstra* sangat cocok diterapkan dalam penelitian ini. Dengan Algoritma *dijkstra* dapat merekomendasikan jalur terpendek jarak tempuh titik awal ke titik tujuan. Hasil akhir penelitian ini adalah sebuah aplikasi android yang dapat memudahkan masyarakat di wilayah Kota Palembang untuk mencari tempat ibadah, rute terdekat dan informasi tempat ibadah agar lebih mudah, cepat dan efisien.

Kata Kunci: *shortest path*, aplikasi android, *eXtreme Programing (XP), GIS, Dijkstra*

Abstract

Searching for the shortest path to find a place of worship for Christians is a problem to find the minimum route from the starting point to the destination point. Searching for the shortest route is based on the starting point, destination point and distance traveled. This research was designed to design an android application that provides information and directions for places of worship in Palembang by applying the dijkstra algorithm and application development methods, namely Agile Methods with eXtreme Programming (XP), also utilizing the Geographic Information System (GIS) to map location locations worship. The dijkstra algorithm is very suitable to be applied in this study. With the dijkstra Algorithm, it can recommend the shortest path to the distance from the starting point to the destination point. The final result of this study is an android application that can make it easier for people in the city of Palembang to find places of worship, the nearest route and information on places of worship to make it easier, faster and more efficient.

Keywords: shortest path, android application, eXtreme Programing (XP), GIS, Dijkstra.

I. PENDAHULUAN

Kota Palembang menyimpan banyak sejarah, budaya, dan tempat wisata yang menarik perhatian banyak orang. Tidak jarang ditemui orang-orang yang dari dalam maupun luar negeri sedang berkunjung ke kota Palembang untuk hiburan, pekerjaan, maupun pendidikan bahkan ada juga yang menetap. Khususnya bagi masyarakat yang beragama Nasrani yang tinggal ataupun hanya sekedar singgah, biasanya akan mengalami kesulitan untuk mencari gereja di sekitar mereka untuk beribadah. Karena mereka belum begitu mengenal dan mengetahui setiap lokasi gereja di kota ini.

Dalam penelitian ini akan menerapkan algoritma *dijkstra* untuk melakukan penelusuran jalur terdekat, pemetakan menggunakan *Geograpich Information System (GIS)*, juga menambahkan informasi tentang tempat ibadah pada aplikasi. Sebenarnya ada banyak algoritma yang dapat menyelesaikan masalah penelusuran jalur terdekat, antara lain algoritma *Floyd-Warshall*, algoritma *dijkstra*, algoritma *Bellman-Ford*, dan lain-lain. Namun peneliti menggunakan algoritma *dijkstra* dalam penilitian ini dan merujuk pada penelitian yang dilakukan oleh Yudi Retanto (2009) dimana "Algoritma *dijkstra* lebih rekomendasi untuk penelusuran jalur terdekat karena algoritma ini dapat dimodifikasi sehingga dapat menelusuri dan menampilkan *path* dari suatu simpul ke simpul lainnya dengan jarak terdekat. Juga karena pada dasarnya jarang ditemui *graph* yang memiliki bobot negatif pada jarak dalam dunia nyata".

Berdasarkan uraian di atas, penulis tertarik untuk membangun sebuah *Aplikasi Penelusuran Objek Lokasi dan Jalur Terdekat Gereja Di Kota Palembang Menggunakan Algoritma Dijkstra* untuk memecahkan permasalahan tersebut. Aplikasi ini akan dibangun pada platfom *android mobile*. Karena selain lebih simpel, *android mobile* juga salah satu perangkat yang banyak digunakan saat ini dan selalu dibawa oleh pemiliknya kemanapun. Hampir setiap kalangan dari remaja hingga orang dewasa menggunakannya. Hal ini juga akan menjadi salah satu faktor pendukung ketepatan sasaran aplikasi yang akan dibangun. Dengan aplikasi ini, pengguna akan semakin mudah dalam mencari dan menelusuri gereja yang diinginkan dan aplikasi juga akan menampilkan jalur terdekat menuju gereja setelah memilih salah satu gereja yang ditampilkan.

II. METODOLOGI PENELITIAN

II.1.Metode Pengumpulan Data

Adapun metode penelitian yang digunakan utnuk memperoleh data dalam penelitian ini adalah:

1. Studi Pustaka *(Library Research)*
   Kegiatan yang dilakukan peneliti dengan menganalisis dan mempelajari literatur-literatur, catatan-catatan, karya tulis dari perpustakaan maupun sumber-sumber lainnya yang berhubungan dengan objek penelitian ini sebagai bahan referensi.

2. Penelitian Lapangan *(Riset)*
   Penelitian lapangan dilakukan untuk memperoleh data *primer* dan data *sekunder* yang dibutuhkan seperti titik koordinat lokasi objek dan jalur.
3. Observasi

Peneliti mengumpulkan data dan informasi serta mengamati aplikasi-aplikasi pencarian jalur terdekat berbasis web maupun aplikasi berbasis android dan sejenisnya sebagai bahan referensi.

II.2.Metode Pengembangan Sistem

Metode dalam penelitian ini menekankan pada metode pengembangan aplikasi yang akan dibangun, yakni *Agile Methods* dengan model *eXtreme Programming (XP). EXtreme Programming (XP)* merupakan suatu metode yang paling banyak digunakan untuk pengembangan perangkat lunak cepat. Model ini cenderung menggunakan *Object-Oriented*. Tahap – tahapan *eXtreme Programming(XP)* dalam pengerjaannya yaitu:

1. Perencanaan *(Planning)*
   Dalam tahap ini peniliti akan merencanakan system dengan merangkum semua permasalahan dan mengumpulkan data yang dibutuhkan. Tahap ini juga mendefinisikan *output* yang akan dihasilkan, fitur yang dimiliki dan fungsi dari aplikasi yang akan dibangun.
2. Perancangan *(Design)*
   Pada tahap *design*, peneliti akan merancang sistem dan menerapkannya kedalam model UML berupa *class diagram, use case diagram* dan *activity diagram*. Hasil dari tahap ini akan digunakan sebagai panduan untuk mempermudah dalam pembangunan sistem.
3. *Coding*

Tahap *coding* merupakan implementasi dari hasil rancangan kedalam bahasa pemograman untuk membangun aplikasi android. Pembangunan aplikasi menggunakan bahasa pemograman android seperti *Eclipse*, *ADT bundle* dan *tools* pendukung lainnya. Pada tahap ini juga akan menyertakan tahap *refactoring*. *Refactoring* merupakan tahap proses mengubah eksternal kode untuk memperbaiki struktur internalnya. Dengan begitu, kemungkinan terdapatnya *bug* akan semakin kecil.

4. *Testing*

   Pada tahapan ini akan lebih difokuskan pada pengujian fitur dan fungsionalitas dari aplikasi yang dibangun, apakah aplikasi dapat memenuhi kebutuhan *user*. Jika kebutuhan *user* terpenuhi, maka aplikasi siap dirilis.

III. HASIL DAN PEMBAHASAN

III.1. Hasil

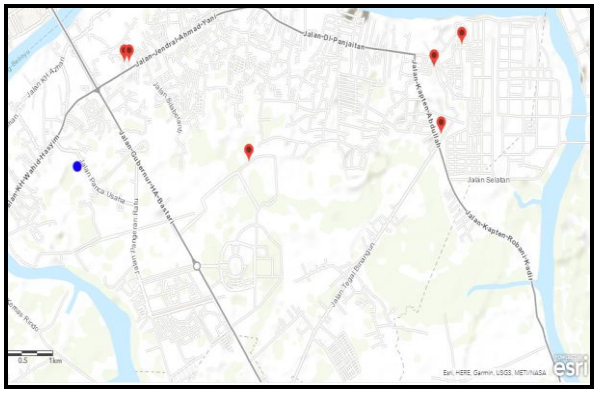
Berdasarkan hasil penelitian yang telah dilakukan, hasil akhir dari semua kegiatan dan tahapan-tahapan pengembangan sistem yang telah dilakukan ini. mulai dari analisa, menerapkan algoritma dijkstra dan penerapan rancangan-rancangan yang telah diuraikan pada bab sebelumnya yang terdiri dari desain *input* dan *output*. Bahasa pemograman yang digunakan dalam pembuatan program aplikasi ini adalah *JAVA*, *Mysql* sebagai basis data dan serta dibantu *software Eclipse*.

Tujuan utama dalam pembuatan program ini adalah untuk membangun sebuah aplikasi penelusuran objek lokasi dan jalur terdekat gereja di kota Palembang yang berbasis android dengan memanfaatkan algoritma dijkstra untuk membantu khususnya setiap pengguna yang berada di kota Palembang yang beragama nasrani melakukan penelusuran gereja dan jalur terdekat di sekitarnya.

III.2. Pembahasan

III.2.1. Penelusuran Objek Gereja

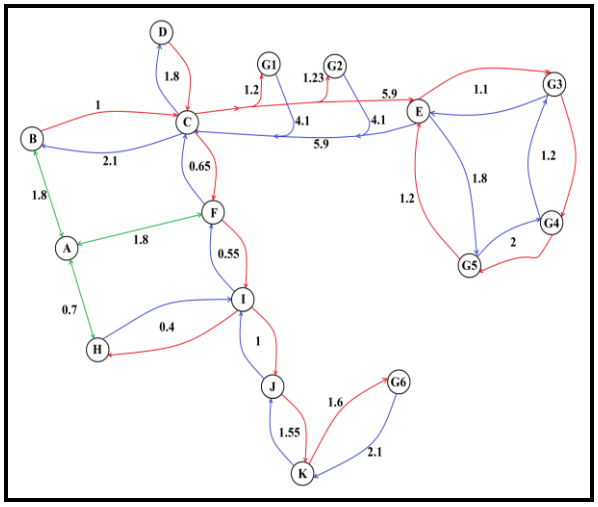
Penelusuran gereja terdekat akan dilakukan dengan menelusuri tiap *node* yang terhubung ke *node* awal, dan nilai yang lebih kecil dari *node* awal akan diberi label dan akan menjadi *node* awal berikutnya, penelusuran pun akan terus dilakukan dan setiap kali menuju suatu *node* maka nilai akan ditotal dan akan membandingkannya ke node yang telah terkunjungi sebelumnnya, jika ternyata node yang sebelumnya adalah yang lebih kecil, maka node tersebut akan diberi label dan akan menjadi *node* awal berikutnya. Perhitungan akan tetap dilakukan berulang kali hingga *node* tujuan terkunjungi dan *node-node* yang dilalui hingga ke *node* akhir akan menjadi jalur terdekat menuju *node* tujuan tersebut. Proses perhitungan pada aplikasi akan disimulasikan pada gambar berikut.



Gambar 3. 1 Lokasi gereja yang akan ditelusuri

Pada gambar diatas terdapat 6 lokasi gereja, pengguna hendak mencari lokasi gereja terdekat, maka algoritma *dijkstra* pada aplikasi akan bekerja seperti gambar 3.2. Perhitungan algoritma dijkstra akan di gambarkan dalam bentuk graf berarah seperti dapat dilihat pada gambar 3.2.

Terdekat 1= G5 dengan jalur melewati *node* : A-H-I-F-C-E-G5



Gambar 3. 2 Grap jalur dan *node* jalan penelusuran gereja

Dalam graf terdapat 16 *node* (termasuk *node* gereja) dan 33 sisi berarah. Posisi pengguna berada pada *node* A dan lokasi gereja pada *node* G1, G2, G3,

G4, G5 dan G6. Maka perhitungannya dapat dilihat pada tabel 3.1

Tabel 3. 1 Perhitungan *dijkstra* untuk penelusuran gereja

G	V	A	B	C	D	E	F	G1	G2	G3	G4	G5	G6	H	I	J	K
1	**A**	0_A	1.8_A	∞	∞	∞	1.8_A	∞	∞	∞	∞	∞	∞	0.7_A	∞	∞	∞
2	**H**		1.8_A	∞	∞	∞	1.8_A	∞	∞	∞	∞	∞	∞	0.7_A	1.1_A	∞	∞
3	**I**		1.8_A	∞	∞	∞	1.65_I	∞	∞	∞	∞	∞	∞		1.1_A	2.1_I	∞
4	**F**		1.8_A	2.3_F	∞	∞	1.65_I	∞	∞	∞	∞	∞	∞			2.1_I	∞
5	**B**		1.8_A	2.3_F	∞	∞		∞	∞	∞	∞	∞	∞			2.1_I	∞
6	**J**			2.3_F	∞	∞		∞	∞	∞	∞	∞	∞			2.1_I	3.65_J
7	**C**			2.3_F	4.1_C	8.3_C		3.5_C	3.53_C	∞	∞	∞	∞				3.65_J
8	**G1**				4.1_C	8.3_C		3.5_C	3.53_C	∞	∞	∞	∞				3.65_J
9	**G2**				4.1_C	8.3_C			3.53_C	∞	∞	∞	∞				3.65_J
10	**K**				4.1_C	8.3_C				∞	∞	∞	5.25_K				3.65_J
11	**D**				4.1_C	8.3_C				∞	∞	∞	5.25_K				
12	**G6**					8.3_C				∞	∞	∞	5.25_K				
13	**E**					8.3_C				9.3_E	∞	10_E					
14	**G3**									9.3_E	10.5_{G3}	10_E					
15	**G5**										10.5_{G3}	10_E					
16	**G4**										10.5_{G3}						

Dari hasil penelusuran lokasi gereja terdekat berdasarkan perhitungan algoritma *dijkstra* pada tabel 3.1 maka diperoleh urutan daftar gereja dari yang terdekat hingga seterusnya. Daftar gereja dan jalur terdekatnya adalah.

Terdekat 1= G1 dengan jalur melewati *node* : A-H-I-F-C-G1 = 3.5 km
Terdekat 2= G2 dengan jalur melewati *node* : A-H-I-F-C-G2 = 3.35 km
Terdekat 3= G6 dengan jalur melewati *node* : A-H-I-J-K-G6 = 5.25 km
Terdekat 4= G3 dengan jalur melewati *node* : A-H-I-F-C-E-G3 = 9.3 km
Terdekat 5= G5 dengan jalur melewati *node* : A-H-I-F-C-E-G5 = 10 km
Terdekat 6= G4 dengan jalur melewati *node* : A-H-I-F-G3-G4 = 10.5 km

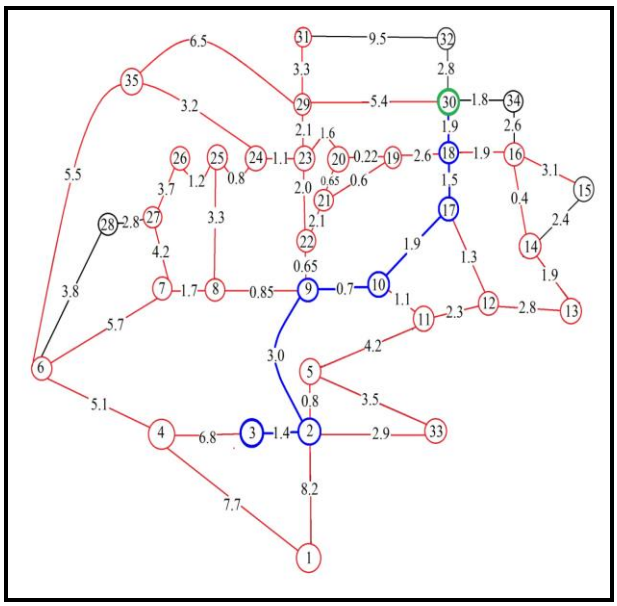
Karena aplikasi ini hanya menelusuri gereja terdekat dengan maksimum jarak 5 km pada menu cari gereja. Maka yang ditampilkan adalah G1 dan G2 saja.

III.2.2. Pencarian jalur terdekat pada menu daftar gereja.

Pencarian pada menu daftar gereja yaitu, aplikasi akan mencari jalur terdekat suatu gereja yang dipilih oleh pengguna dari daftar gereja Tahap pertama adalah menginisialisasi *node* awal dan jarak antar *node.*

Gambar 3. 3 Jalur jalan pada peta yang akan dicari

Dapat dilihat pada gambar 3.3 di atas, bahwa jalur menuju gereja ada lebih dari satu, dengan begitu algoritma *dijkstra* akan berperan untuk menghitung dan menentukan jalur terdekat ke objek tempat tujuan.



Gambar 3. 4 Gambaran jalur jalan pada graf

Gambar 3.4 adalah gambaran jalur jalan pada peta dalam bentuk graf untuk menentukan jalur terdekat ke objek tujuan seperti yang dijelaskan pada gambar 3.3. Satuan arak pada jalur adalah kilometer (km), graf terdiri dari 35 *node* dan 52 sisi (*edge*). *Node* 3 adalah *node* awal dan *node* 30 adalah *node* lokasi tujuan. Perhitungan dapat dilihat pada tabel 3.2

(dist[node], prev[node] interaction.

Initialization

g=0 (*Node* awal)

a, b, c, d, e, f, h, i, j, k, l = ∞

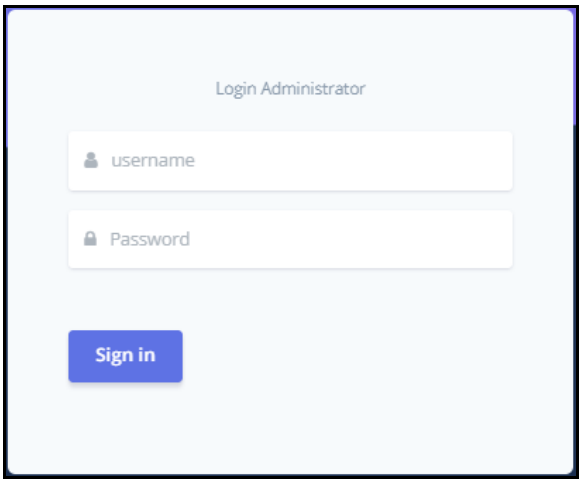
Tabel 3. 2 Urutan perhitungan algoritma *dijkstra*

G	v	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	3	∞	1.4_{3}	0_{3}	6.8_{3}	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	2	9.6_{2}	1.4_{3}		6.8_{3}	2.2_{2}	∞	∞	∞	4.4_{2}	∞	∞	∞	∞	∞	∞	∞	∞	∞
3	5	9.6_{2}			6.8_{3}	2.2_{2}	∞	∞	∞	4.4_{2}	∞	6.4_{5}	∞	∞	∞	∞	∞	∞	∞
4	33	9.6_{2}			6.8_{3}		∞	∞	∞	4.4_{2}	∞	6.4_{5}	∞	∞	∞	∞	∞	∞	∞
5	9	9.6_{2}			6.8_{3}		∞	∞	5.25_{9}	4.4_{2}	5.1_{9}	6.4_{5}	∞	∞	∞	∞	∞	∞	∞
6	22	9.6_{2}			6.8_{3}		∞	∞	5.25_{9}		5.1_{9}	6.4_{5}	∞	∞	∞	∞	∞	∞	∞
7	10	9.6_{2}			6.8_{3}		∞	∞	5.25_{9}		5.1_{9}	6.2_{10}	∞	∞	∞	∞	∞	7_{10}	∞
8	8	9.6_{2}			6.8_{3}		∞	6.95_{8}	5.25_{9}			6.2_{10}	∞	∞	∞	∞	∞	7_{10}	∞
9	4	9.6_{2}			6.8_{3}		11.9_{4}	6.95_{8}				6.2_{10}	∞	∞	∞	∞	∞	7_{10}	∞
10	11	9.6_{2}					11.9_{4}	6.95_{8}				6.2_{10}	8.5_{11}	∞	∞	∞	∞	7_{10}	∞
11	7	9.6_{2}					11.9_{4}	6.95_{8}					8.5_{11}	∞	∞	∞	∞	7_{10}	∞
12	23	9.6_{2}					11.9_{4}						8.5_{11}	∞	∞	∞	∞	7_{10}	∞
13	17	9.6_{2}					11.9_{4}						8.3_{17}	∞	∞	∞	∞	7_{10}	8.5_{17}
14	21	9.6_{2}					11.9_{4}						8.3_{17}	∞	∞	∞	∞		8.5_{17}
15	19	9.6_{2}					11.9_{4}						8.3_{17}	∞	∞	∞	∞		8.5_{17}
16	20	9.6_{2}					11.9_{4}						8.3_{17}	∞	∞	∞	∞		8.5_{17}
17	12	9.6_{2}					11.9_{4}						8.3_{17}	11.1_{12}	∞	∞	∞		8.5_{17}
18	18	9.6_{2}					11.9_{4}							11.1_{12}	∞	∞	10.4_{18}		8.5_{17}
19	24	9.6_{2}					11.9_{4}							11.1_{12}	∞	∞	10.4_{18}		
20	25	9.6_{2}					11.9_{4}							11.1_{12}	∞	∞	10.4_{18}		
21	1	9.6_{2}					11.9_{4}							11.1_{12}	∞	∞	10.4_{18}		
22	29						11.9_{4}							11.1_{12}	∞	∞	10.4_{18}		
23	26						11.9_{4}							11.1_{12}	∞	∞	10.4_{18}		
24	16						11.9_{4}							11.1_{12}	10.8_{16}	13.5_{16}	10.4_{18}		
25																			

19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4.3_{2}	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4.3_{2}	∞	∞
∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	4.3_{2}	∞	∞
∞	∞	∞	5.05_{9}	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞		∞	∞
∞	∞	7.15_{22}	5.05_{9}	7.05_{22}	∞	∞	∞	∞	∞	∞	∞	∞	∞		∞	∞
∞	∞	7.15_{22}		7.05_{22}	∞	∞	∞	∞	∞	∞	∞	∞	∞		∞	∞
∞	∞	7.15_{22}		7.05_{22}	∞	8.55_{8}	∞	∞	∞	∞	∞	∞	∞		∞	∞
∞	∞	7.15_{22}		7.05_{22}	∞	8.55_{8}	∞	∞	∞	∞	∞	∞	∞		∞	∞
∞	∞	7.15_{22}		7.05_{22}	∞	8.55_{8}	∞	∞	∞	∞	∞	∞	∞		∞	∞
∞	∞	7.15_{22}		7.05_{22}	∞	8.55_{8}	∞	11.15_{7}	∞	∞	∞	∞	∞		∞	∞
∞	$8.65._{23}$	7.15_{22}		7.05_{22}	8.15_{23}	8.55_{8}	∞	11.15_{7}	∞	9.15_{23}	∞	∞	∞		∞	∞
∞	$8.65._{23}$	7.15_{22}			8.15_{23}	8.55_{8}	∞	11.15_{7}	∞	9.15_{23}	∞	∞	∞		∞	∞
7.75_{21}	7.8_{21}	7.15_{22}			8.15_{23}	8.55_{8}	∞	11.15_{7}	∞	9.15_{23}	∞	∞	∞		∞	∞
7.75_{21}	7.8_{21}				8.15_{23}	8.55_{8}	∞	11.15_{7}	∞	9.15_{23}	∞	∞	∞		∞	∞
	7.8_{21}				8.15_{23}	8.55_{8}	∞	11.15_{7}	∞	9.15_{23}	∞	∞	∞		∞	∞
					8.15_{23}	8.55_{8}	∞	11.15_{7}	∞	9.15_{23}	∞	∞	∞		∞	∞
					8.15_{23}	8.55_{8}	∞	11.15_{7}	∞	9.15_{23}	10.4_{18}	∞	∞		∞v	∞
					8.15_{23}	8.55_{8}	∞	11.15_{7}	∞	9.15_{23}	10.4_{18}	∞	∞		∞	12.35_{23}
						8.55_{8}	9.75_{25}	11.15_{7}	∞	9.15_{23}	10.4_{18}	∞	∞		∞	12.35_{23}
							9.75_{25}	11.15_{7}	∞	9.15_{23}	10.4_{18}	∞	∞		∞	12.35_{23}
							9.75_{25}	11.15_{7}	∞	9.15_{23}	10.4_{18}	12.45_{29}	∞		∞	12.35_{23}
							9.75_{25}	11.15_{7}	∞		10.4_{18}	12.45_{29}	∞		∞	12.35_{23}
								11.15_{7}	∞		10.4_{18}	12.45_{29}	∞		13_{16}	12.35_{23}

											10.4_{18}					

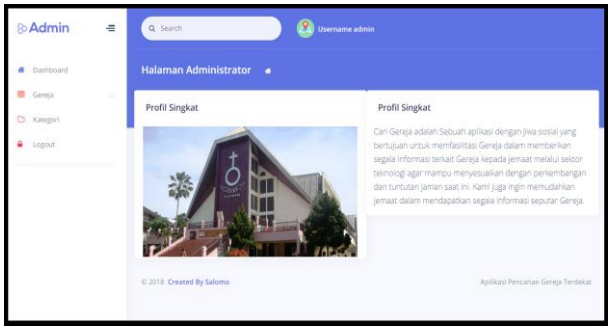
Algoritma *dijkstra* memulai penelusuran dari *node* 3 menuju ke *node* 2 dengan nilai bobot sebesar 1.4_3, dan *node* 4 dengan nilai bobot 6.8_3. Untuk menentukan *node* yang akan dipilih berikutnya maka dilakukan pencarian nilai bobot terkecil dari *node* yang belum pernah terkunjungi sebelumnya yaitu *node* 2. Setelah *node* yang akan dilalui selanjutnya telah ditemukan, kemudian akan dilakukan penjumlahan nilai dari semua *node* sebelumnya dalam kasus ini dari *node* 3 akan menuju *node* 2 karena nilainnya lebih kecil yaitu 1.4_3 lalu ke *node* 5 dengan nilai bobot 0.8 menjadi *node* 3-2-5 dengan total nilai bobot 2.2_2. Namun *node-node* ini belum tentu jarak terdekat karena *dijkstra* akan menghitung kembali *node-node* yang belum terkunjungi. Jika node yang belum terkunjungi adalah nilai yang lebih kecil maka node tersebut akan diberi label permanen dan akan melakukan penelusuran kembali. Tahap ini akan terus diulang hingga node 30 terkunjungi.



Gambar 3. 5 Halaman *login* admin

Pada halaman *login* admin terdapat *form* input *username* dan *password. Login admin* dilakukan agara *admin* dapat mengelola data yang ada pada aplikasi pencarian gereja terdekat kota Palembang. *Admin* harus mengisi *username* dan *password* secara benar agar dapat masuk ke halaman *admin*, jika salah maka *admin* gagal melakukan *login.*

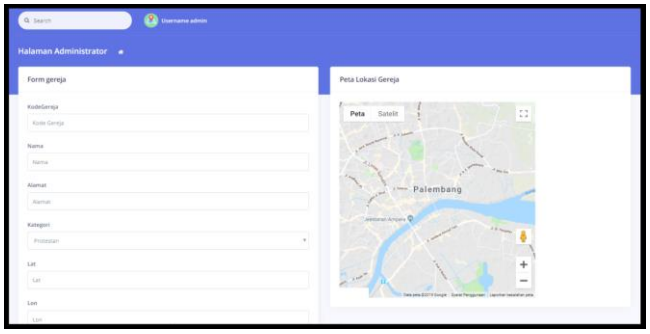
1. Halaman Utama *Dashboard Admin*



Gambar 3. 6 Halaman Utama Dashboard Admin

Halaman utama *dashboard admin* merupakan sebuah halaman yang berisi berkaitan mengenai tentang dan penjelasan aplikasi penelusuran gereja di kota Palembang.

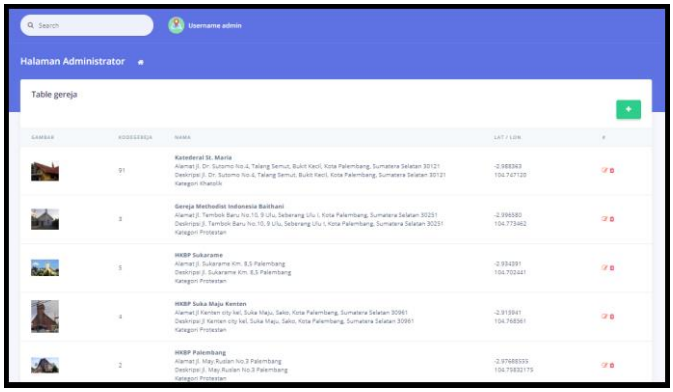
2. Halaman *input* data gereja



Gambar 3. 7 Halaman *input* gereja

Pada halaman *input* data gereja ini admin dapat melakukan *input* kode gereja, nama gereja, alamat gereja, *latitude*, *longitude*, deskripsi, gambar dan memilih kategori gereja (Katolik atau Protestan).

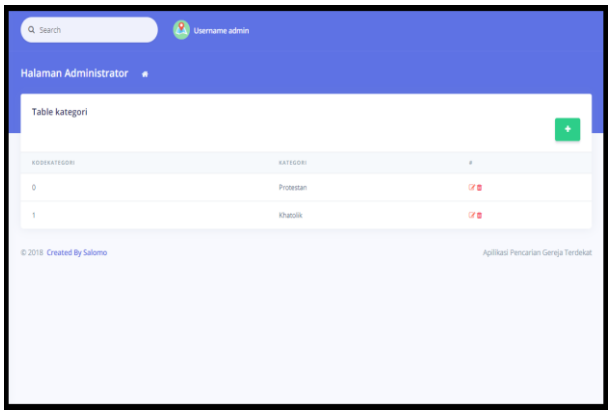
3. Halaman tabel gereja



Gambar 3. 8 Halaman tabel gereja

Halaman tabel gereja merupakan sebuah halaman dari seorang *admin* yang telah melakukan *input* data gereja dan kemudian data-data tersebut akan disimpan dalam tabel gereja. Di halaman ini juga *admin* dapat melakukan tambah, *update*, *edit* dan hapus data gereja.

4. Halaman tabel kategori gereja

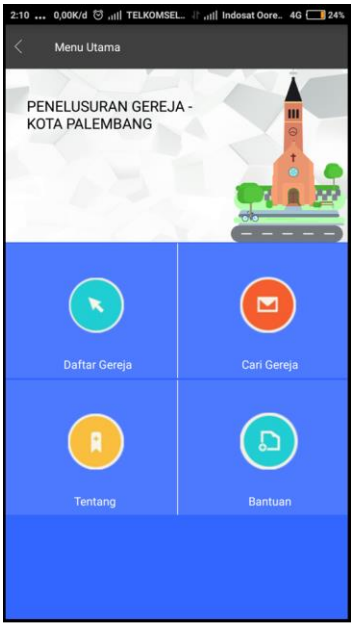


Gambar 3. 9 Halaman tabel kategori gereja

Halaman tabel kategori merupakan sebuah halaman dari seorang *admin* yang dapat mengelompokkan setiap gereja agar sesuai dengan kategori gereja tersebut. Di halaman ini juga admin dapat melakukan perubahan kategori seperti tambah, *update*, *edit*, dan hapus.

Interface* Halaman *User

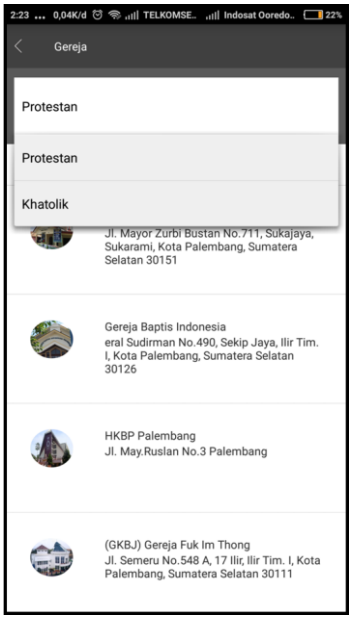
1. Halaman utama aplikasi *user*



Gambar 3. 10 Halaman utama aplikasi *user*

Halaman utama ini adalah halaman utama dari aplikasi pencarian gereja terdekat di kota Palembang. Di halaman utama ini terdapat empat menu, mulai dari menu daftar gereja, cari gereja, tentang, dan bantuan.

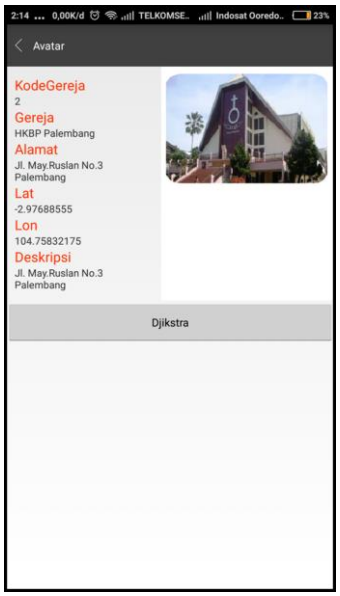
2. Halaman daftar gereja



Gambar 3. 11 Halaman daftar gereja

Pada halaman ini aplikasi akan menampilkan semua daftar gereja yang ada di kota Palembang setelah *admin* melakukan *input* data gereja pada halaman *input* data gereja di dalam aplikasi *admin*. Pada halaman ini *user* juga dapat memilih gereja sesuai kategori gereja yang telah ditentukan oleh *admin.*

3. Halaman detail gereja



Gambar 3. 12 Halaman detail gereja

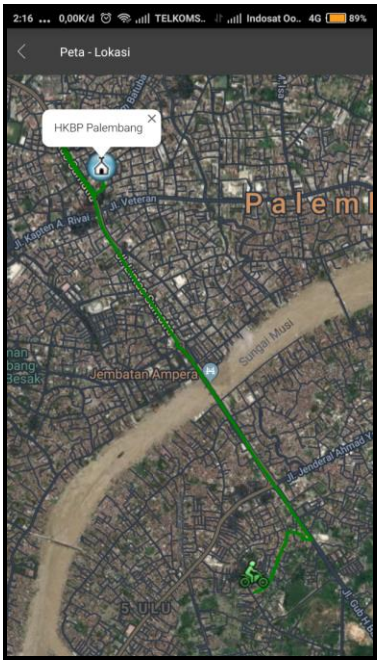
Halaman detail gereja adalah saat *user* memilih salah satu gereja pada halaman daftar gereja. Halaman ini, aplikiasi akan menampilkan detail gereja seperti kode, nama, alamat, *latitude*, *longitude* dan deskripsi suatu gereja.

4. Halaman nilai hasil *dijkstra*.

Gambar 3. 13 Halaman nilai hasil *dijkstra*

Halaman ini menampilkan nilai setiap jalur dan waktu tempuh dari posisi awal pengguna ke posisi tujuan. Pada halaman ini juga terdapat *button dijkstra* yang akan mengarahkan *user* ke halaman rute terdekat pada peta.

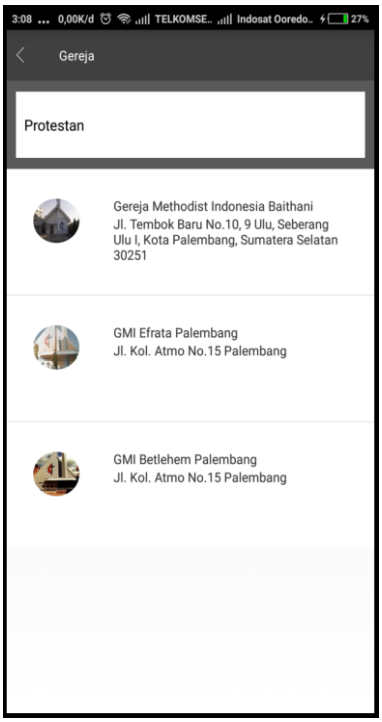
5. Halaman jalur terdekat pada peta



Gambar 3. 14 Halaman jalur terdekat pada peta

Pada halaman jalur terdekat pada peta adalah saat pengguna memilih *dijkstra* di halaman detail gereja sebelumnya. Halaman ini menampilkan jalur terdekat warna biru dari posisi *user* menuju lokasi gereja yang telah pilih sebelumnya.

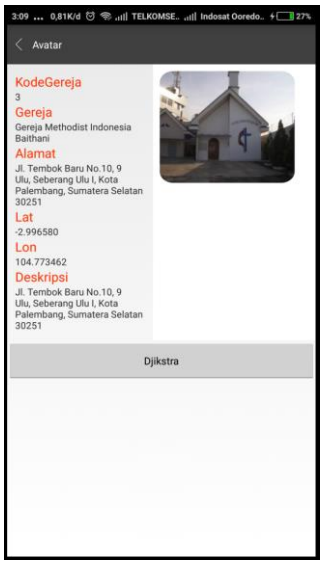
6. Halaman cari gereja



Gambar 3. 15 Halaman cari gereja

Halaman cari gereja adalah dimana pada saat *user* memilih menu cari gereja pada halaman utama dan aplikasi akan menampilkan daftar gereja terdekat dari posisi *user*. Pada halaman ini *user* dapat memilh salah satu gereja yang ingin diketahui jalur terdekatnya.

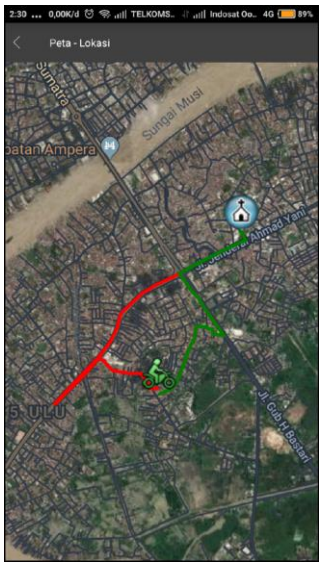
7. Halaman detail gereja



Gambar 3. 16 Halaman detail gereja

Sama halnya dengan Gambar 3.7 sebelumnya. Hanya saja pada halaman detail gereja ini adalah saat *user* memilih salah satu gereja terdekat pada halaman cari gereja sebelumnya. Pada halaman ini, aplikiasi akan menampilkan detail gereja seperti kode, nama, alamat, *latitude, longitude* dan deskripsi suatu gereja.

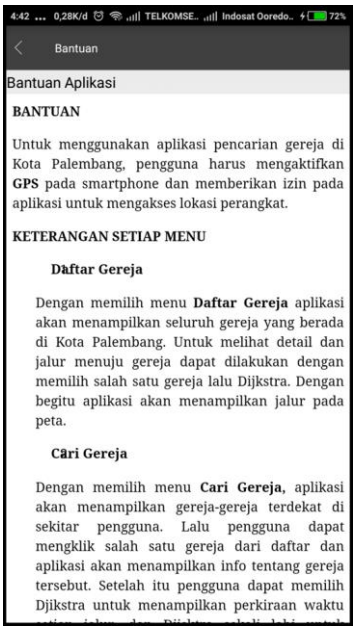
8. Halaman jalur terdekat pada peta



Gambar 3. 17 Halaman jalur terdekat pada peta

Pada halaman jalur terdekat pada peta adalah saat pengguna memilih *dijkstra* di halaman detail gereja sebelumnya. Halaman ini menampilkan jalur terdekat dari posisi *user* menuju lokasi gereja yang telah pilih sebelumnya.

9. Halaman bantuan



Gambar 3. 18 Halaman bantuan aplikasi

Halaman ini menampilkan bantuan dan cara mengoperasikan aplikasi penelusuran objek lokasi gereja kota Palembang.

10. Halaman tentang

Gambar 3. 19 Halaman tentang aplikasi

Halaman ini menampilkan tentang aplikasi, yakni alasan dibuatnya aplikasi, metode yang digunakan untuk merancang, dan algoritma yang dipakai.

IV. KESIMPULAN DAN SARAN

IV.1. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan pada aplikasi penelusuran objek lokasi dan jalur terdekat gereja di kota Palembang menggunakan algoritma *dijkstra* berbasis *android* (Studi kasus: Kota Palembang), maka dapat disimpulkan sebagai berikut:

1. Menghasilkan sebuah aplikasi *android mobile* yaitu aplikasi penelusuran gereja dan jalur terdekatnya di kota Palembang, untuk membantu khususnya pengguna yang beragama nasrani.
2. Dihasilkan sebuah aplikasi penelusuran gereja di kota Palembang yang dimana mencakup 2 (dua) jenis kategori gereja yaitu protestan dan katolik.
3. Aplikasi yang dihasilkan ini mampu melakukan penelusuran gereja dan menampilkan jalur terdekat menuju gereja tersebut.
4. Sistem yang dihasilkan ini mampu melakukan pengelolaan data setiap gereja kota Palembang. Pengelolaan data tersebut meliputi input, update, hapus data gereja.
5. Dengan adanya aplikasi penelusuran gereja ini, *user* dapat mengetahui, nama, jumlah, lokasi bahkan jalur terdekat menuju gereja yang berada di kota Palembang.

IV.2. Saran

Aplikasi penelusuran gereja berbasis *android mobile* ini masih sangat jauh dari kata sempurna dan masih banyak sekali kekurangannya. Oleh karena itu, perlu dilakukan pengembangan dan penyempurnaan lebih lanjut. Adapun saran agar aplikasi penelusuran gereja ini dapat berjalan secara optimal dan terlihat lebih menarik yaitu dengan sebagai berikut ini:

1. Sistem perangkat lunak ini bisa dikembangkan lagi dengan menambahkan *navigator* agar perjalanan pada rute lebih *real time*.
2. Aplikasi yang dibangun dapat dikembangkan lagi agar aplikasi tetap dapat digunakan meski tidak terkoneksi dengan internet (*Offline*)
3. Aplikasi yang dibangun dapat dikembangkan dengan menambahkan layanan *chat* dan membangun sistem bagi pihak setiap gereja agar *user* dapat berinteraksi langsung dengan pihak (pengurus) gereja.

V. DAFTAR PUSTAKA

___, Java*. http://www.techopedia.com/3927/java*. *Diakses tanggal 20 Februari 2018.*

___, Pengertian *Android Development Tools (ADT).* *https://www.google.co.id/amp/s/haidibarasa.wordpress.com/2013/07/06/pengertian-android-development-tools-adt/amp/. Diakses 20 Februari 2018.*

https://www.google.co.id/amp/s/haidibarasa.wordpress.com/2013/07/06/pengertian-android-development-tools-adt/amp/. Pengertian *Android Development Tools (ADT). diakses 20 Februari 2018.*

Kemenag. 2016, Oktober. Data Pemeluk Agama 2016. *https://sumsel.kemenag.go.id/artikel/view/39692/data-pemeluk-agama-2016. diakses pada 06 Februari 2018.*

Sedgewick, Robert dan Kevin Wayne. 2018. *Algorithms. Shortest paths. Fourth Edition.https://algs4.cs.princeton.edu/44sp/. diakses tanggal 16 Februari 2018.*