BAB III

ANALISIS DAN PERANCANGAN SISTEM

Seperti yang telah dijelaskan pada bab sebelumnya, pada bab ini akan dijelaskan tentang perencanaan dan langkah pembuatan dari aplikasi, dari mulai langkah-langkah awal analisa sampai kepada perancangan sistem.

3.1. Analisis sistem

Analisis sistem merupakan kegiatan penguraian suatu sistem in*form*asi yang utuh dan nyata ke dalam bagian-bagian atau komponen-komponen komputer yang bertujuan untuk mengidentifikasi serta mengevaluasi masalah-masalah yang muncul, hambatan-hambatan yang akan mungkin terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga mengarah kepada solusi untuk perbaikan maupun pengembangan ke arah yang lebih baik dan sesuai dengan kebutuhan serta perkembangan teknologi. Adapun tahapan-tahapan yang ada pada analisa sistem ini meliputi : analisis masalah, analisis kebutuhan, lingkungan operasi, analisa basis data dan analisa prosedur sistem.

a. Analisis Masalah

Dalam menggunakan alat transportasi umum terkadang masyarakat memiliki beberapa permasalahan seperti ketidaktauan sebuah in*form*asi alat transportasi umum unuk jadwal keberangkatan, alat transportasi yang harus

digunakan untuk mencapai ke tempat tujuan, in*form*asi untuk pemesanan alat transportasi seperti taksi agar tidak kesulitan dalam mencari alat trasnportasi tersebut.

Permasalahan tersebut sebenarnya sudah memiliki solusi seperti jadwal keberangkatan pada kereta api yang tersedia pada website PT. Kereta Api Indonesia dan juga tersedia pada stasiun kereta api, in*form*asi untuk mencapai ke tempat tujuan dengan bus Transjakarta, in*form*asi pemesanan taksi yang hanya tersedia pada website penyedia jasa taksi dan buku *yellow pages* serta in*form*asi untuk mencapai ke tempat tujuan dengan patas dan angkutan kota yang hanya tersedia pada beberapa halte di jakarta

Solusi dari permasalahan tersebut masih memiliki kekurangan karena dengan solusi itu masyarakat tidak menghemat waktu dan agak rumit dalam mencari in*form*asi tersebut. Solusi yang baik adalah membangun sebuah sistem. pada sistem yang akan dibangun, posisi dari seseorang dan pencarian halte Trans Jakarta yang divisualisasikan pada peta digital, memberikan in*form*asi untuk mencapai ke tempat tujuan, menyediakan jadwal kereta api dan nomor telpon dari *provider* taksi, dan tidak mempersulit *user* dalam menggunakan aplikasi ini yang berada pada *smartphone* berbasis android.

Permasalahan yang dapat diidentifikasi pada sistem ini adalah data harus bersifat dinamis. Solusi untuk permasalahan-permasalahan tersebut adalah data-data harus tersimpan ke dalam *server* dan database *server*.

b. Analisis kebutuhan sistem

Kebutuhan sistem yang diperlukan dan yang digunakan dalam pembuatan sistem ini terdiri dari beberapa bagian, yakni : kebutuhan fungsional dan kebutuhan non fungsional.

3.1.1. Kebutuhan Non Fungsional

a. Spesifikasi Perangkat Lunak

Perangkat lunak digunakan dalam sebuah sistem merupakan perintah-perintah yang diberikan kepada perangkat keras agar bisa saling berinteraksi di antara keduanya. Perangkat lunak yang dibutuhkan untuk membangun aplikasi transportasi *guide* ini adalah sebagai berikut :

1) Sistem operasi Windows Vista Home Basic
2) J2SE SDK (*Java* 2 *Standart Edition Software Development Kit*) Versi 1.6.0
3) Eclipse versi 3.5
4) ADT (*Android Development Tools*)
5) AVD (*Android Virtual Device*)

Program ini dirancang dengan menggunakan Java Eclipse dengan menggunakan kelas-kelas standar untuk aplikasi GUI pada level tinggi dan level rendah. Jadi aplikasi ini dapat digunakan pada *smartphone* berbasis android dengan versi android 2.1 atau biasa dikenal dengan Froyo (*Frozen Yoghurt*). Oleh karena itu dalam pengujiannya, program ini menggunakn *emulator* android.

b. Spesifikasi Perangkat Keras

Komputer dan *smartphone* terdiri dari perangkat keras dan perangkat lunak yang saling berinteraksi. Perangkat lunak memberikan instruksi-instruksi kepada perangkat keras untuk melakukan suatu tugas tertentu, sehingga dapat menjalankan suatu sistem didalamnya.

Pada aplikasi transportasi *guide* ini, perangkat keras yang digunakan untuk mengembangkan aplikasi adalah sebagai berikut :

1) Intel Celeron M
2) RAM 2 GB
3) Hard disk 80 GB
4) Mouse dan Keyboard
5) *Smartphone* Android versi 2.1

Sedangkan perangkat keras yang digunakan untuk menguji langsung hasil dari aplikasi ini adalah *Smartphone* berbasis android 2.1.

c. Spesifikasi Pengguna (*user)*

Spesifikasi pengguna adalah kriteria yang harus dimiliki oleh pengguna agar tidak kesulitan dalam menjalankan aplikasi trnasportasi *guide* ini. Spesifikasi pengguna adalah sebagai berikut :

Tabel 3.1 Spesifikasi Pengguna

Syarat	Pengguna
Mengerti Operasional *Smartphone*	✓
Dapat Mengoperasikan Aplikasi pada *Smartphone*.	✓
Memiliki *Smartphone* Android 2.1	✓

d. Analisis Google Map Api

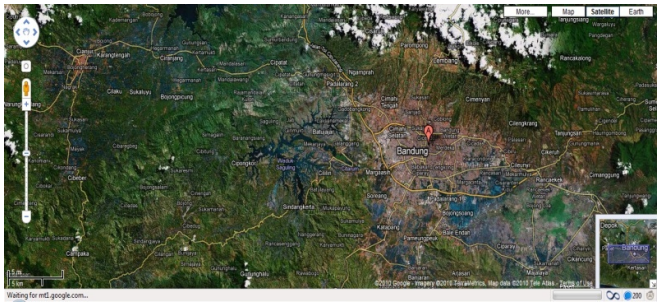
Google menyediakan layanan API (*Application Programming Interface)* memungkinkan aplikasi *client* untuk melihat, menyimpan dan memperbarui data peta dalam bentuk Data API Google *feed* dengan menggunakan model data fitur (letak, garis dan bentuk) dalam peta. Aplikasi ini diberi nama Google Maps API (GMaps API). Peta yang ditampilkan diambil dari layanan Google Maps. Ada tiga jenis tampilan yang bisa dipilih dari Google Maps, yaitu:

1. *Map* menampilkan peta dalam bentuk peta garis. Seperti pada gambar di bawah ini.



Gambar 3. 1 Penggambaran kota Bandung dalam bentuk *map*

2. *Sattelite* menampilkan peta dalam bentuk citra/foto satelit. Seperti pada gambar di bawah ini.



Gambar 3.2 Penggambaran kota Bandung dalam bentuk satelit

3. *Earth* merupakan gabungan dari *Map* dan *Satelite*, seperti pada gambar di bawah ini.



Gambar 3.3 Penggambaran kota Bandung dalam bentuk *Earth*

Menampilkan peta dan menentukan bagian peta yang ditampilkan. Peta yang diambil dari Google API akan menampilkan peta dengan titik koordinat (-6.861326 sampai 107.594862) serta memiliki *zoom level* 13.

e. Elemen-elemen yang terdapat pada Google Map Api

Elemen-elemen yang terdapat pada Google Map API adalah :

1. *Marker*

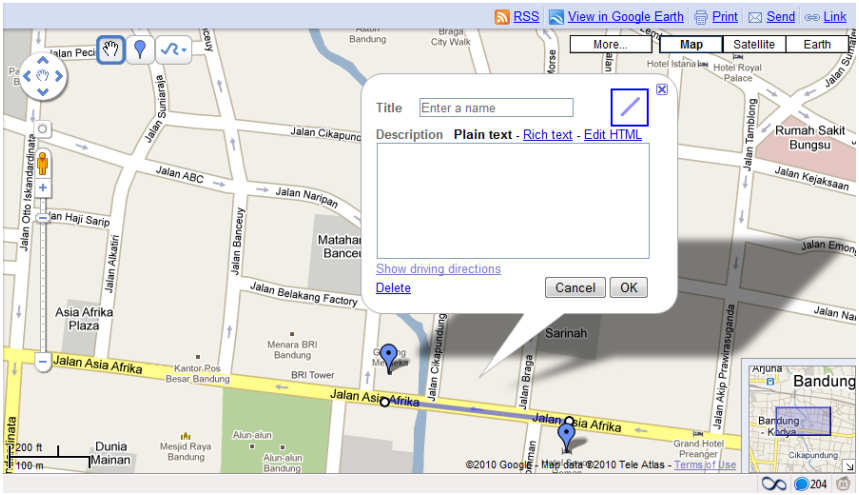
Marker adalah simbol yang menandakan suatu lokasi bangunan sejarah pada peta yang ditampilkan Google Maps.



Gambar 3.4 Penggunaan *marker*

2. *Polyline*

Polyline adalah *shape* yang digunakan untuk menandakan suatu jalur jalan atau area. *Polyline*, yaitu terdiri dari kumpulan titik koordinat.



Gambar 3.5 Penggunaan *polyline*

Objek model yang terdapat pada *Google Map API* adalah :

1. Inisialisasi *Map*

   inisialisasi di proses dengan menggunakan *method setCenter*(). *Method setCenter*() membutuhkan GetLatLng koordinat dan *zoom level*, dan method ini harus segera dikirim sebelum ada pengoperasian lain pada peta, termasuk seting atribut peta itu sendiri

2. *Loading Google Maps API* koneksi *script* yang kita buat ke*server* Google Map API dengan menggunakan *key* yang anda dapatkan pada saat anda mendaftar ke Google Maps API.

3. GMap2 - *Elementary Object*

   Class javascript yang membuat peta itu ada adalah *class* GMap2, *Object* dari *class* ini akan menyediakan sebuah peta di halaman web, *Variable map* akan berikan nilai sebuah object dari *class GMap2*. Fungsi *GMap2* adalah sebagai *constructo*r dan definisinya.

4. *Loading Map*

   Ketika halaman HTML di *render*, *document object model* (DOM) sudah bisa digunakan, dan semua gambar *external* dan *script* diterima oleh *object document*. Untuk memastikan peta kita dimuat sesudah halaman selesai dimuat oleh *browser*

5. *Latitude and Longitude*

   Objek *GLatLng* menentukan titik koordinat peta yang akan ditampilkan, parameternya terdiri dari lintang/*latitude* dan bujur/*longitude*.

6. Fungsi *Gdirections*

   Gdirection adalah *utility* yang menangani masalah pencarian rute optimal dari satu tempat ke tempat lain.

f. Analisis Perhitungan Jarak Lokasi Tujuan pada GPS

Dalam menentukan posisi suatu lokasi pada penelitian ini metode yang digunakan yaitu *Cell-ID (Cell Identification)*. Yaitu metode yang didasarkan pada daerah geografis yang tercakup oleh sebuah cell yang berhubungan dengan daerah cakupan dari sinyal GPS atau radio. Untuk itu cell yang dikenali dari *base station* merupan sebuah nilai koordinat berupa *Latitude* dan *Longitude*. Untuk mengukur jarak *mobile* dari *base station* tidak dapat diketahui dengan pasti, Oleh karena itu, untuk lebih meningkatkan lagi akurasi hasil pencarian, metode *Cell ID* dapat dikombinasikan dengan metode *Timing Advanced* (*TA*), dengan menggunakan TA ini, metode *Cell ID* akan ditambahkan sebuah fungsionalitas untuk menghitung *Round Trip Time* (*RTT*), yaitu waktu transmisi sebuah frame (dari *base station* ke handphone) dan waktu penerimaan sebuah frame (dari *mobile* ke *base station*). Dengan tambahan metode ini, jarak antara *mobile* dan base station dapat ditentukan dengan keakuratan 50 m.

Berikut ini salah satu rumus yang sering digunakan untuk menentukan jarak lokasi pada metode CELL ID dengan mengabaikan bentuk geografis baik bukit maupun lembah.

Jarak = Arc Cos(Cos(Lat1) x Cos(Lat2) x Cos(Long2-Long1) + **Sin(Lat1) x Sin(Lat 2)) x R**

Dimana

Lat1 = Nilai Latitude dari Lokasi Awal

Long1 = Nilai Longitude dari Lokasi Awal

Lat2 = Nilai Latitude dari Lokasi Tujuan

Long2 = Nilai Latitude dari Lokasi Tujuan

R = Jari-jari rata-rata Bumi (6378.2 KM)

Catatan : Nilai Lat dan Long dalam satuan radian.

Contoh :

Misalkan untuk mencari jarak antara Bandara Sultan Iskandar Muda di Banda Aceh ke Bandara Sentani di Jayapura Jika diketahui :

1. Latitude 1 (Banda Aceh) 5°31'1"N dan Longitude 1 (Banda Aceh) 95°25'12"E
2. Latitude 2 (Jayapura) 2°34'19"S dan Longitude 2 (Jayapura)140°30'41"E

Koordinat diatas kan masih dalam bentuk degree minute secon dirubah ke bentuk degree decimal

1. Latitude 1 (Banda Aceh) 5.516944 dan Longitude 1 (Banda Aceh) 95.42
2. Latitude 2 (Jayapura) -2.571944 dan Longitude 2 (Jayapura) 140.511389

Perhitungan :

Jarak = Arc Cos (Cos(5.516944 * π / 180) *

Cos(-2.571944 * π / 180) *

Cos((140.511389 * π / 180) – (95.42* π / 180)) +

Sin(5.516944 * π / 180) *

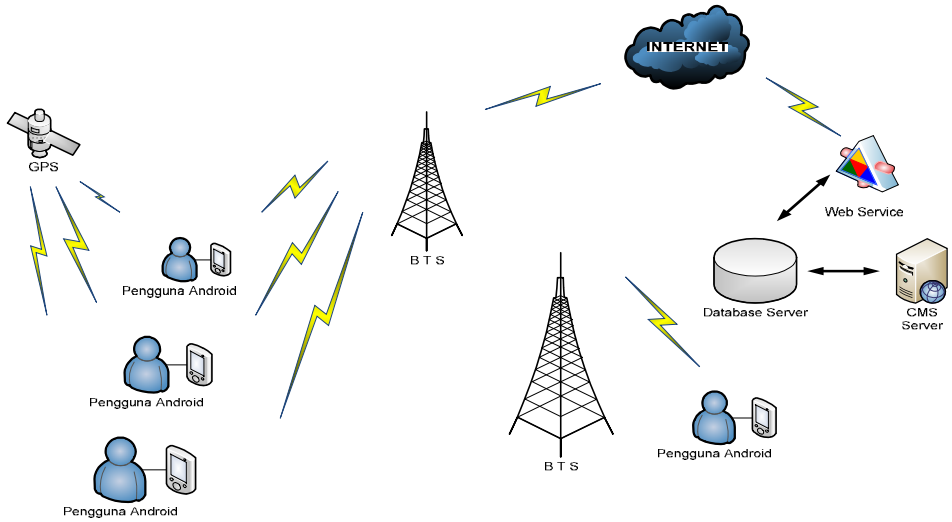
Sin(-2.571944 * π / 180)) * 6378.2

= Arc Cos (0.6976857) * 6.371

= 5093.849 KM

g. Analisis Arsitektur

Arsitektur fisik sistem mengacu kepada model arsitektur aplikasi *three tier*. Arsitektur fisik sistem terdiri dari tiga bagian utama yaitu *client*, *application server*, dan database *server*. Arsitektur sistem dalam pembangunan aplikasi Transportasi *Guide* ini ditunjukan oleh gambar 3.19 dibawah ini.

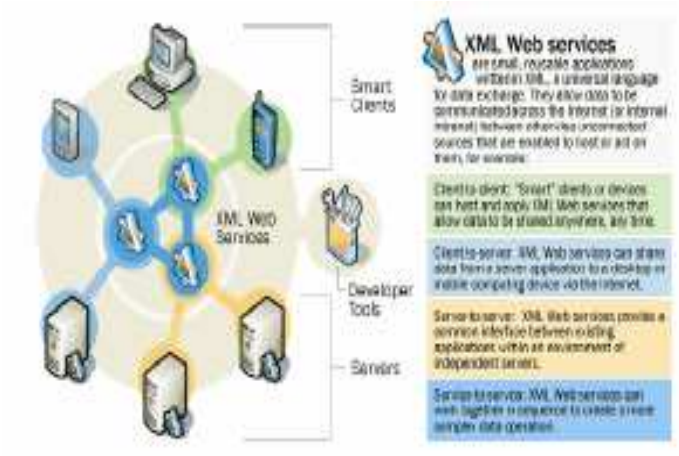


Gambar 3.6 Arsitektur Sistem

Pengguna android merupakan pengguna aplikasi pada *frontend* yaitu Aplikasi Transportasi *guide* yang sudah terinstal pada perangkat *mobile*. Bagian ini berfungsi sebagai antarmuka antara pengguna dengan sistem. Pengembangan antarmuka sistem dibangun dengan menggunakan IDE Eclipse dan SDK android dengan bahasa pemrogramam java.

Webservice merupakan aplikasi menengah (*middle application*) yang berfungsi sebagai penghubung antara aplikasi *frontend* dengan database *server* dalam proses penggunaan basis data. *Web Services* merupakan salah satu bentuk implementasi dari arsitektur model aplikasi N-Tier yang berorientasi layanan. Perbedaan *Web Services* dengan pendekatan N-Tier lainnya adalah dari segi infrastruktur dan dokumen yang digunakan sebagai *form*at pertukaran data. Dalam implementasinya, *Web Services* tidak mempunyai tampilan, karena *Web Services* termasuk dalam Business-Service tier. Artinya didalam *Web Services* hanya tersedia fungsi-fungsi yang nantinya dapat digunakan oleh aplikasi lainnya *Web Services* menggunakan XML sebagai *form*at dokumen dalam melakukan pertukaran datanya. Karena XML merupakan suatu *form*at dokumen yang berbasis teks, maka *Web Services* memungkinkan berlangsungnya komunikasi antar aplikasi yang berbeda dengan plat*form* yang berbeda pula. *Web Services* dapat diimplementasikan dalam berbagai jenis plat*form* dengan menggunakan bahasa pemrograman apa pun, dan bisa digunakan oleh berbagai aplikasi yang menggunakan bahasa pemrograman apapun dengan plat*form* apapun juga. Selama aplikasi tersebut dapat berkomunikasi dengan *Web Service* menggunakan protokol-protokol

komunikasi, Termasuk HTTP, XML, SOAP, UDDI (*Universal Description Discovery and Integration*), dan WSDL (*Web Services Description Language*). Pada Gambar 3.7 berikuft merupakan gambaran sederhana konsep *Web Services* serta keterhubungan antara *Web Services* dengan aplikasi Client maupun *Server*.



Gambar 3.7 Konsep Webservice

MS *server* merupakan aplikasi *backend* yang berfungsi sebagai pengelolaan data pada database *server* dalam proses penggunaan basis data. Pada bagian ini aplikasi dibangun dengan menggunakan bahasa pemrograman PHP.

Database *server* merupakan aplikasi *backend* yang berfungsi menyimpan data-data yang digunakan oleh layanan (Admin). Bagian ini di implementasikan menggunakan MySQL.

GPS pada arsitektur ini merupakan aktor lain yang bergerak didalam stasiun radio luar yang berfungsi untuk memberitahu keberadaan posisi pengguna baik dengan dengan koordinat *latitude* maupun *longitude*.

Akses koneksi jaringan yang digunakan penguna untuk berintegrasi dengan GPS atau pengguna ke BTS melalui perangkat mobile dapat menggunakan jaringan mobile GPRS/EDGE/UMTS.

3.1.2. Kebutuhan Fungsional

Dalam membangun aplikasi transportasi *guide* di Jakarta ini digunakan metode perancangan berorientasi objek. Pada dasarnya perancangan objek memberikan kemudahan dalam hal perawatan atau *maintenance* dan pengembangan sistem lebih lanjut.

Selain itu, perancangan aplikasi sistem ini menggunakan metode perancangan berorientasi objek dengan menggunakan UML (*Unfied Modelling Language*) untuk menentukan, visualisasi, konstruksi, dan mendokumentasikan in*form*asi yang digunakan atau dihasilkan dalam suatu proses pembuatan sistem.

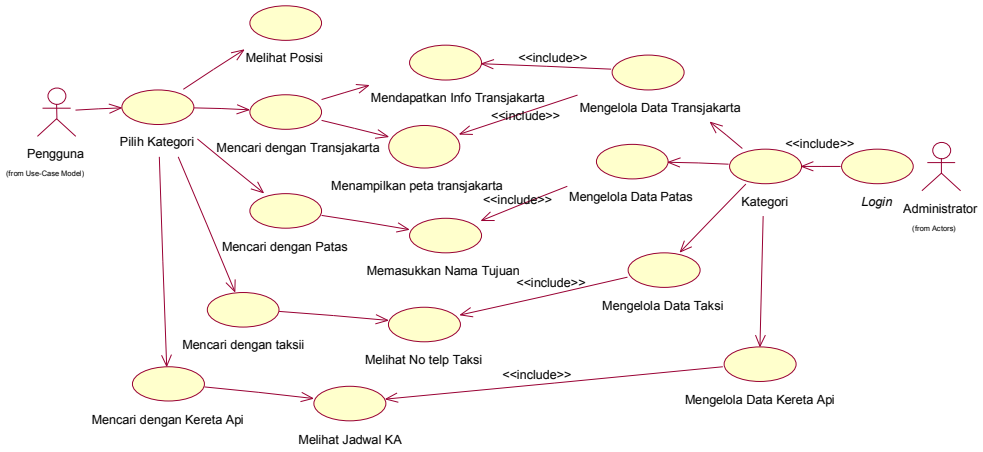
Berdasarkan analisis yang telah dilakukan dalam pembuatan aplikasi ini, maka dapat dilihat apa yang menjadi masukan (*input*), keluaran (*output*), metode yang digunakan, dan antar muka sistem yang dibuat, sehingga sistem yang dibuat sesuai yang diharapkan. Hasil perancangan berupa gambaran umum mengenai aplikasi secara keseluruhan yang akan menjelaskan proses implementasi dari sistem.

3.2. Perancangan UML

3.2.1. Use Case Diagram

Pemodelan yang akan digunakan dalam menggambarkan kebutuhan fungsional pada aplikasi yang dibangun yaitu dengan pemodelan Use Case

Diagram. Use case diagram pada aplikasi transoprtasi *guide* adalah sebagai berikut :



Gambar 3.8. Use Case diagram

3.2.1.1 Identifikasi Aktor

Identifikasi aktor adalah Aktor yang berperan dalam menjalankan sistem, aktor yang berperan dapat dilihat pada tabel dibawah ini :

Tabel 3.2. Identifikasi Aktor

No.	Aktor	Deskripsi
A-01	Pengguna	Merupakan aktor yang berperan dalam menggunakan aplikasi Transportasi *Guide* untuk mendapatkan in*form*asi ke tempat tujuan.
A-02	Admin	Merupakan aktor yang berperan dalam mengelola aplikasi Transportasi *Guide* melalui CMS berbasis Web.

Gambar 3.1. menjelaskan bahwa aktor berperan sebagai pengguna aplikasi (*frontend*) yang berinteraksi dengan perangkat bergerak atau *smartphone* berbasis android.

Sedangkan Gambar 3.2. memiliki aktor yang berperan sebagai admin yang bertugas mengelola aplikasi (*backend*) pada CMS (*Content Management System*) berbasis Web.

3.2.1.2 Identifikasi Use Case

Identifikasi *Use* Case adalah *Use Case* yang bekerja pada sistem aplikasi dan *Use Case* yang bekerja pada sistem adalah sebagai berikut :

Tabel 3.3 Identifikasi Use Case

No.	Nama *Use Case*	Deskripsi
UC-01	Pilih Kategori	Fungsionalitas yang dapat memilih kategori transportasi yang terdiri dari Transjakarta, patas, taksi, dan kereta api.
UC-02	Melihat posisi	Fungsionalitas ini dapat menggambarkan posisi pengguna pada peta dari Google Maps
UC-03	Mencari dengan Transjakarta	Fungsionalitas ini merupakan pilihan dari kategori untuk mencari tujuan menggunakan Transjakarta
UC-04	Mencari dengan patas	Fungsionalitas ini termasuk dalam salah satu pilihan kategori yang digunakan untuk mencari in*form*asi tempat tujuan dengan patas
UC-05	Mencari dengan taksi	Pada fungsionalitas ini, hanya mencari no telp penyedia jasa taksi di Jakarta
UC-06	Mencari jadwal Kereta Api	Fungsionalitas ini hanya mencari jadwal keberangkatan Kereta Api.
UC-07	Mendapatkan info transjakarta	Fungsionalitas ini dapat memberi in*form*asi mengenai data seluruh halte per koridor
UC-08	Menampilkan peta	Fungsionalitas ini berfungsi untuk

	transjakarta	menampilkan peta transjakarta
UC-09	Memasukkan Nama Tujuan	Fungsionalitas ini hanya memasukkan nama tujuan dengan menggunakan alat transportasi patas
UC-10	Melihat no telp taksi	Fungsionalitas ini hanya melihat daftar no telp taksi yang sudah tersedia
UC-11	Melihat jadwal Kereta Api	Fungsionalitas ini hanya menyediakan daftar jadwal keberangkatan KA yang sudah tersedia.
UC-12	*Login*	Fungsionalitas ini berguna hanya untuk admin, agar admin dapat mengelola data transportasi *guide*
UC-13	Pilih Kategori	Fungsionalitas yang dapat memilih kategori Transjakarta, patas, taksi, dan kereta api untuk dikelola datanya oleh admin
UC-14	Mengelola Data Transjakarta	Fungsionalitas ini berguna untuk mengelola data Transjakarta.
UC-15	Mengelola Data Patas	Fungsionalitas ini berguna untuk mengelola data patas
UC-16	Mengelola data taksi	Fungsionalitas ini berguna untuk mengelola data taksi yang berada diseluruh Jakarta.
UC-17	Mengelola data kereta api	Fungsionalitas ini berguna untuk mengelola data keberangkatan kereta api hanya dari dan ke Jakarta.

3.2.1.3 Skenario Use Case

Skenario Use Case adalah alur cerita atau proses-proses yang terjadi pada sistem antara aktor dengan Use Case. Skenario Use Case ditunjukkan pada tabel-tabel dibawah ini :

Tabel 3.4. Skenario Use Case Pilih Kategori

Identifikasi	
Nama *Use Case*	Pilih Kategori
Aktor	Pengguna
Tujuan	Memilih pilihan menu

Keadaan Awal	Sistem menampilkan menu pilihan
Sekenario Utama (UC01-A01)	
Aksi Aktor	**Reaksi sistem**
1. Memilih menu (where am I, Transjakarta, patas, taksi, kereta api) yang ditampilkan oleh sistem.	
	2.Jika memilih menu where am i maka akan menampilkan posisi pengguna pada peta
	3.jika memilih menu Transjakarta maka akan tampil menu pencarian halte Transjakarta dan peta Transjakarta untuk mencapai ketempat tujuan
	4.jika memilih menu patas maka akan menampilkan *input* nama tempat tujuan,
	5.jika memilih menu taksi maka akan menampilkan daftar no telp taksi
	6.jika memilih menu kereta api maka akan menampilkan jadwal keberangkatan kereta api.
Skenario Alternatif (UC01-A01) – Verifikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1.Menampilkan pesan bahwa proses gagal dilakukan (”Cek Koneksi Jaringan”)
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	Menampilkan in*form*asi kategori sesuai yang dipilih

Tabel 3.5. Skenario Use Case Melihat Posisi

Identifikasi	
Nama *Use Case*	Melihat Posisi
Aktor	Pengguna
Tujuan	Melihat posisi pengguna
Keadaan Awal	Sistem menampilkan menu pilih kategori

Sekenario Utama (UC02-A01)	
Aksi Aktor	**Reaksi sistem**
1. memilih menu where am i	
	2. menampilkan posisi pengguna dengan menggunakan peta dari Google Maps dan memanfaatkan teknologi GPS pada *smartphone*.
Skenario Alternatif (UC02-A01) – Verifikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1. Menampilkan pesan bahwa proses gagal dilakukan (”Cek Koneksi Jaringan”)
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	Menampilkan posisi pengguna

Tabel 3.6. Skenario Use Case Mencari Dengan Transjakarta

Identifikasi	
Nama *Use Case*	Mencari dengan Transjakarta, mendapatkan posisi halte Transjakarta, dan mencari tujuan dengan Transjakarta, mengelola data Transjakarta
Aktor	Pengguna
Tujuan	Mendapatkan in*form*asi ke tempat tujuan dgn Transjakarta
Keadaan Awal	Sistem menampilkan menu pilih kategori
Sekenario Utama (UC03-UC07-UC08-UC14-A01)	
Aksi Aktor	**Reaksi sistem**
1. memilih menu Transjakarta	
	2. jika memilih menu info transjakarta maka akan menampilkan seluruh data koridor dan seluruh data halte per koridor
	3.jika memilih menu menampilkan peta transjakarta maka akan menampilkan peta transjakarta
4.memilih menu info transjakarta	
	5. menampilkan seluruh data koridor dan seluruh data halte per koridor

6. memilih menu menampilkan peta transjakarta	
	7. menampilkan peta dari google maps yang berisi posisi pengguna dan posisi-posisi halte transjakarta. Dan juga memanfaatkan teknologi GPS
Skenario Alternatif (UC03-UC07-UC08-UC14-A01) – Verifikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1. Menampilkan pesan bahwa proses gagal dilakukan (”Cek Koneksi Jaringan”)
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	Menampilkan posisi halte Transjakarta atau menampilkan in*form*asi ke tempat tujuan

Tabel 3.7. Skenario Use Case Mencari Dengan Patas

Identifikasi	
Nama *Use Case*	Mencari dengan patas, memasukkan nama tujuan, mengelola data patas
Aktor	Pengguna
Tujuan	Mendapatkan in*form*asi ke tempat tujuan dgn patas
Keadaan Awal	Sistem menampilkan menu pilih kategori
Sekenario Utama (UC04-UC09-UC15-A01)	
Aksi Aktor	**Reaksi sistem**
1. memilih menu patas	
	2. menampilkan menu patas
3.mengisi *keyword* nama tempat tujuan ke kotak tujuan	
	4. kemudian mencarinya di *webservice*
	5.menampilkan in*form*asi sesuai dengan *keyword*
Skenario Alternatif (UC04-UC09-UC15-A01) – Verifikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1. Menampilkan pesan bahwa proses gagal dilakukan (”Cek Koneksi Jaringan”)

2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	Menampilkan in*form*asi sesuai dengan *keyword*

3.8. Skenario Use Case Mencari Dengan Taksi

Identifikasi	
Nama *Use Case*	Mencari dengan taksi, melihat no telp taksi, mengelola data taksi
Aktor	Pengguna
Tujuan	Mendapatkan in*form*asi no telp taksi
Keadaan Awal	Sistem menampilkan menu pilih kategori
Sekenario Utama (UC05-UC10-UC16-A01)	
Aksi Aktor	**Reaksi sistem**
1. memilih menu Taksi	
	2. menampilkan menu Taksi
	3.mencari seluruh data taksi pada mengelola data taksi di *webservice*
	4.menampilkan seluruh no telp taksi
5. memilih no telp untuk langsung di telpon	
Skenario Alternatif (UC05-UC10-UC16-A01) – Verifikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1. Menampilkan pesan bahwa proses gagal dilakukan (”Cek Koneksi Jaringan”)
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	Menampilkan list seluruh no telp.

Tabel 3.9. Skenario Use Case Mencari Jadwal Kereta Api

Identifikasi	
Nama *Use Case*	Mencari Jadwal Kereta Api, melihat jadwal Kereta Api, mengelola data Kereta Api
Aktor	Pengguna
Tujuan	Mendapatkan in*form*asi keberangkatan KA

Keadaan Awal	Sistem menampilkan menu pilih kategori
Sekenario Utama (UC06-UC11-UC17-A01)	
Aksi Aktor	**Reaksi sistem**
1. memilih menu Kereta Api	
	2. mencari jadwal keberangkatan kereta api di *server*
	3.menampilkan Jadwal Keberangkatan Kereta api
Skenario Alternatif (UC06-UC11-UC17-A01) – Autentikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1. Menampilkan pesan bahwa proses gagal dilakukan (”Cek Koneksi Jaringan”)
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	Menampilkan jadwal keberangkatan KA

Tabel 3.10. Skenario Use Case *Login*

Identifikasi	
Nama *Use Case*	*Login*
Aktor	Admin
Tujuan	Mengelola data
Keadaan Awal	menampilkan halaman *login*
Sekenario Utama (UC12-A02)	
Aksi Aktor	**Reaksi sistem**
1. memasukkan *user*name dan password	
	2. memeriksa kesamaan antara *user*name dan password.
	3.jika password dan *user*name benar maka akan menampilkan pilih kategori
Skenario Alternatif (UC12-A02) – Autentikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1. Jika password dan *user*name salah maka menampilkan Pesan ”*user*name dan Password Salah”
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.

Kondisi Akhir	Menampilkan pilih kategori

Tabel 3.11. Skenario Use Case Pilih Kategori Admin

Identifikasi	
Nama *Use Case*	Pilih Kategori
Aktor	Admin
Tujuan	Mengelola data
Keadaan Awal	menampilkan halaman pilih kategori
Sekenario Utama (UC12-A02)	
Aksi Aktor	**Reaksi sistem**
1. memilih data yang akan dikelola	
	2. jika memilih Transjakarta maka akan menampilkan halaman mengelola data Transjakarta
	3. jika memilih patas maka akan menampilkan halaman mengelola data patas
	4. jika memilih taksi maka akan menampilkan halaman mengelola data taksi
	5. jika memilih Kereta Api maka akan menampilkan halaman mengelola data Kereta Api
Skenario Alternatif (UC12-A02) – Verifikasi Gagal	
Aksi Aktor	**Reaksi sistem**
Kondisi Akhir	Memilih kategori

Tabel 3.12. Skenario Use Case Mengelola Data Transjakarta

Identifikasi	
Nama *Use Case*	Mengelola Data Transjakarta
Aktor	Admin
Tujuan	Mengelola data Transjakarta
Keadaan Awal	Menampilkan halaman pilih kategori
Sekenario Utama (UC14-A02)	

Aksi Aktor	Reaksi sistem
1.memilih mengelola data Transjakarta	
	2.menampilkan halaman mengelola data Transjakarta
2.menambah atau menghapus data Transjakarta	
	3.menambah atau menghapus data Transjakarta di database
Skenario Alternatif (UC14-A02) – Verifikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1.Menampilkan pesan ”Field Tidak Boleh Kosong”
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	data Transjakarta berhasil di *update*

Tabel 3.13. Skenario Use Case Mengelola Data Patas

Identifikasi	
Nama *Use Case*	Mengelola Data Patas
Aktor	Admin
Tujuan	Mengelola data Patas
Keadaan Awal	Menampilkan halaman pilih kategori
Sekenario Utama (UC15-A02)	
Aksi Aktor	**Reaksi sistem**
1.memilih mengelola data patas	
	2.menampilkan halaman mengelola data patas
3.menambah, merubah atau menghapus data patas	
	3.menyimpan ke database patas
Skenario Alternatif (UC15-A02) – Verifikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1.Menampilkan pesan ”Field Tidak Boleh Kosong"
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	Data patas berhasil *update*

Tabel 3.14. Skenario Use Case Mengelola Data Taksi

Identifikasi	
Nama *Use Case*	Mengelola Data Taksi
Aktor	Admin
Tujuan	Mengelola data taksi
Keadaan Awal	menampilkan halaman pilih kategori
Sekenario Utama (UC17-A02)	
Aksi Aktor	**Reaksi sistem**
1.memilih mengelola data taksi	
	2.menampilkan halaman mengelola data taksi
3.menambah, merubah, atau menghapus data taksi	
	4.menyimpan ke database taksi
Skenario Alternatif (UC17-A02) – Verifikasi Gagal	
Aksi Aktor	**Reaksi sistem**
	1.Menampilkan pesan ”Field Tidak Boleh Kosong"
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	Data taksi berhasil *update*

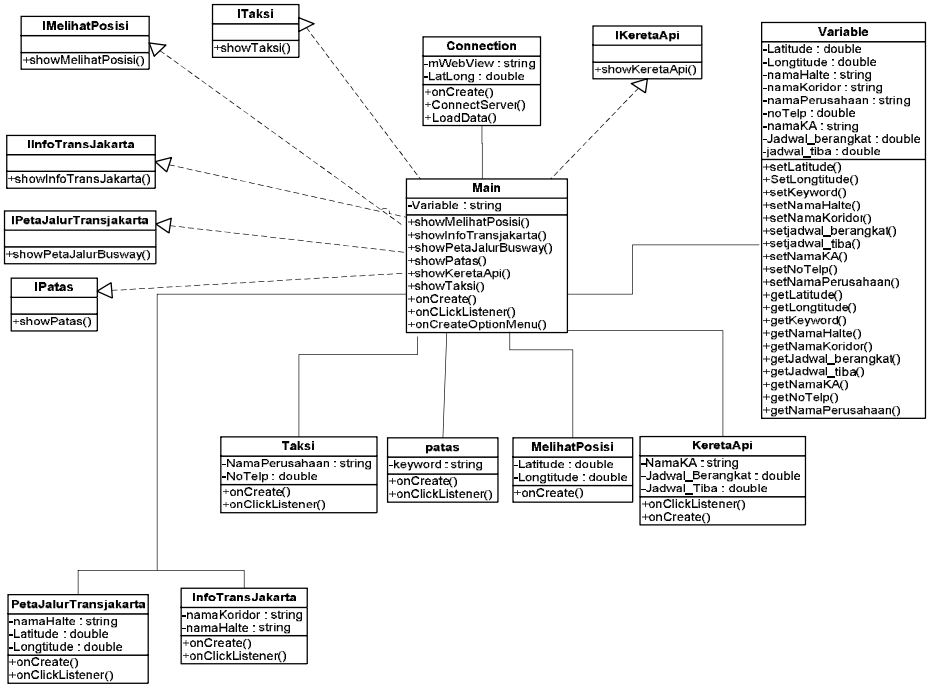
Tabel 3.15. Skenario Use Case Mengelola Data KA

Identifikasi	
Nama *Use Case*	Mengelola Data KA
Aktor	Admin
Tujuan	Mengelola data KA
Keadaan Awal	menampilkan halaman pilih kategori
Sekenario Utama (UA07-A02)	
Aksi Aktor	**Reaksi sistem**
1.memilih mengelola data kereta api	
	2.menampilkan halaman mengelola data kereta api
3.menambah,merubah atau menghapu data Kereta api	
	4.menyimpan ke database Kereta Api
Skenario Alternatif (UA07-A02) – Verifikasi Gagal	

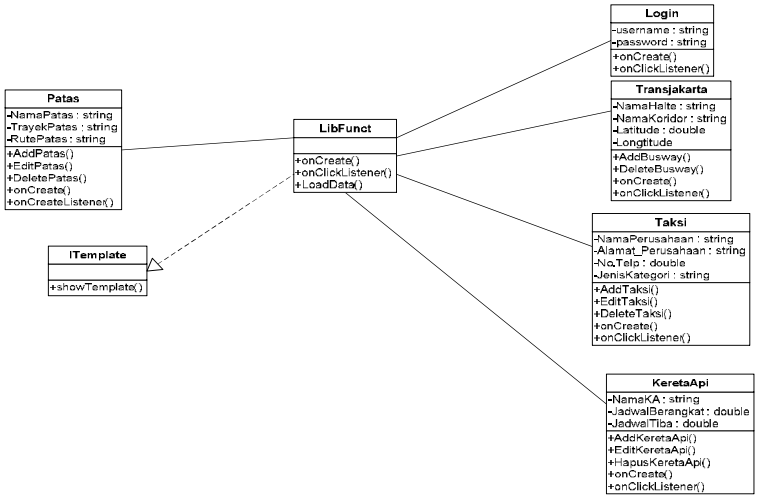
Aksi Aktor	Reaksi sistem
	1. Menampilkan pesan ”Field Tidak Boleh Kosong "
2. Menekan tombol ”OK”	
	3. Sistem akan *terminate*.
Kondisi Akhir	Data kereta api berhasil *update*

3.2.2. Class Diagram

Class adalah spesifikasi objek, yang memiliki atribut/property dan layanan/fungsional (metode/fungsi). Pada tahap ini class diagram dibagi menjadi dua yaitu untuk *frontend* dan *backend* :



Gambar 3.9 class diagram aplikasi *Frontend*



Gambar 3.10 Class Diagram pada Aplikasi *Backend*

Tabel 3.16 Deskripsi Class Diagram

Class	Jenis Kelas	Deskripsi
Main	Control	*Class* utama yang berisikan method-method yang berhubungan dengan class lain dan berfungsi sebagai *controller*
ShowMelihatPosisi	Interface	*Class* menu yang berisikan method-method untuk menampilkan posisi pengguna pada peta
InfoTrans	Control	*Class* yang berisikan method-method yang berhubungan dengan menampilkan in*form*asi Transjakarta

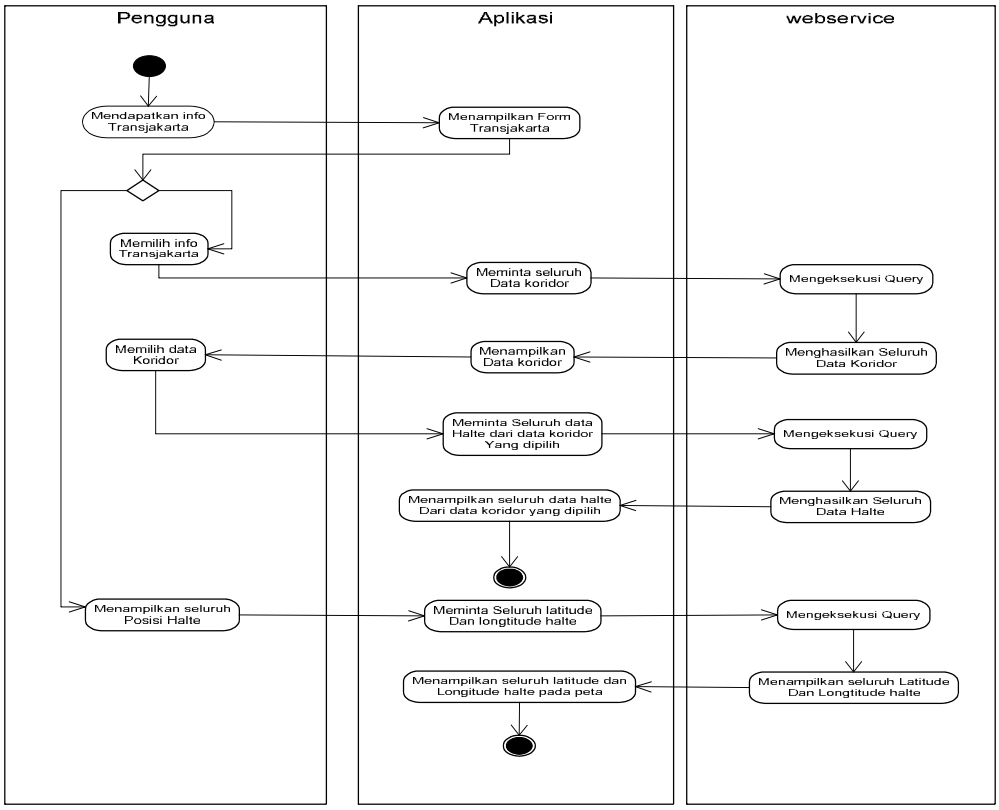
PetaJalurTransjakarta	Control	*Class* yang berisikan method-method yang berhubungan dengan menampilkan peta jalur Transjakarta
Taksi	Control	*Class* yang berisikan method-method yang berhubungan dengan dengan menampilkan data taksi
Patas	Control	*Class* yang berisikan method-method yang digunakan untuk mencari nama tempat tujuan dan berhubungan juga dengan menampilkan data hasil pencarian
KeretaApi	Control	*Class* yang berisikan methode-method yang berhubungan dengan menampilkan data kereta api
MelihatPosisi	Control	*Class* yang berisikan method-method yang berhubungan dengan menampilkan posisi pengguna pada peta
ShowInfoTransJakarta	Interface	Class yang berisikan method-method untuk menampilkan data koridor dan data halte per koridor
ShowPetaJalurTransjakarta	Interface	*Class* yang berisikan method-method untuk menampilkan posisi dari halte Transjakarta pada peta
ShowPatas	Interface	*Class* yang berisikan method-method untuk menampilkan pencarian dan hasil pencarian dari nama tempat tujuan.
showTaksi	Interface	*Class* yang berisikan method-method untuk menampilkan data taksi berupa List

ShowKeretaApi	Interface	*Class* yang berisikan method-method untuk menampilkan data kereta api
Connection	Control	*Class* yang berisikan method-method yang menangani koneksi ke *server*
Variable	Entity	*Class* yang berisikan method-method yang berhubungan dengan variabel-variabel yang akan digunakan dalam *query* ke *server*
Template	Interface	*Class* yang berisikan *form* untuk menampilkan template
LibraryFunct	Control	*Class* yang berisikan method-method untuk mengontrol *class* entity.
Login	Entity	*Class* yang berisikan data admin
Transjakarta	Entity	*Class* yang berisikan data halte Transjakarta
Patas	Entity	*Class* yang berisikan data patas
Taksi	Entity	*Class* yang berisikan data Taksi
KeretaApi	Entity	*Class* yang berisikan data Kereta Api

3.2.3. Activity Diagram

Activity diagram adalah teknik untuk mendeskripsikan logika procedural, proses bisnis dan aliran kerja dalam banyak kasus. Activity diagram mempunyai peran seperti halnya *flowchart*, akan tetapi perbedannya dengan *flowchart* adalah activity diagram bisa mendukung perilaku parallel sedangkan *flowchart* tidak bisa.

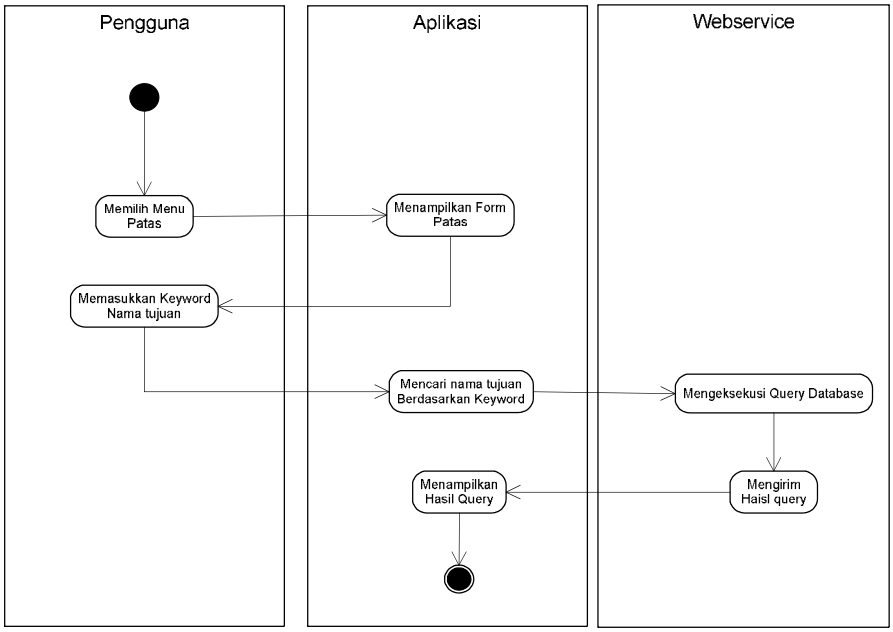
1. Activity diagram Pencarian Tujuan dengan Transjakarta



Gambar 3.11 Activity Diagram Pencarian Tujuan dengan Transjakarta

Gambar activity diagram diatas menjelaskan tentang aktivitas yang terjadi pada menu Transjakarta yang dilakukan oleh pengguna, pada awalnya pengguna memilih menu Transjakarta pada menu pilih kategori kemudian sistem menampilkan *form* Transjakarta yang selanjutnya akan menampilkan 2 menu pilihan, jika pengguna memilih mendapatkan info transjakarta maka diaplikasi akan meminta seluruh data koridor ke *webservice* yang kemudian dari *webservice* akan mengeksekusi *Query* ke database setelah itu hasil *Query* akan di kirim ke aplikasi untuk di tampilkan data koridor kemudian pengguna memilih koridor berapa yang akan dipilih yang kemudian di aplikasi akan meminta seluruh data

halte dari data koridor yang dipilih setelah itu di *webservice* akan kembali di eksekusi *Query* database agar menghasilkan seluruh data halte yang kemudian akan ditampilkan di aplikasi data halte dari data koridor yang dipilih Sedangkan jika pengguna memilih menu yang kedua yaitu mencari tujuan dengan peta jalur Transjakarta, jika pengguna memilih mendapatkan posisi halte Transjakarta maka diteruskan dengan meminta posisi halte Transjakarta pada *server* kemudian permintaan diproses di *webservice* sehingga menghasilkan posisi halte Transjakarta yang kemudian hasil tersebut siap untuk dikirim ke aplikasi dan pada aplikasi akan menampilkan posisi halte untuk pengguna. Activity Diagram Pencarian Tujuan dengan Patas

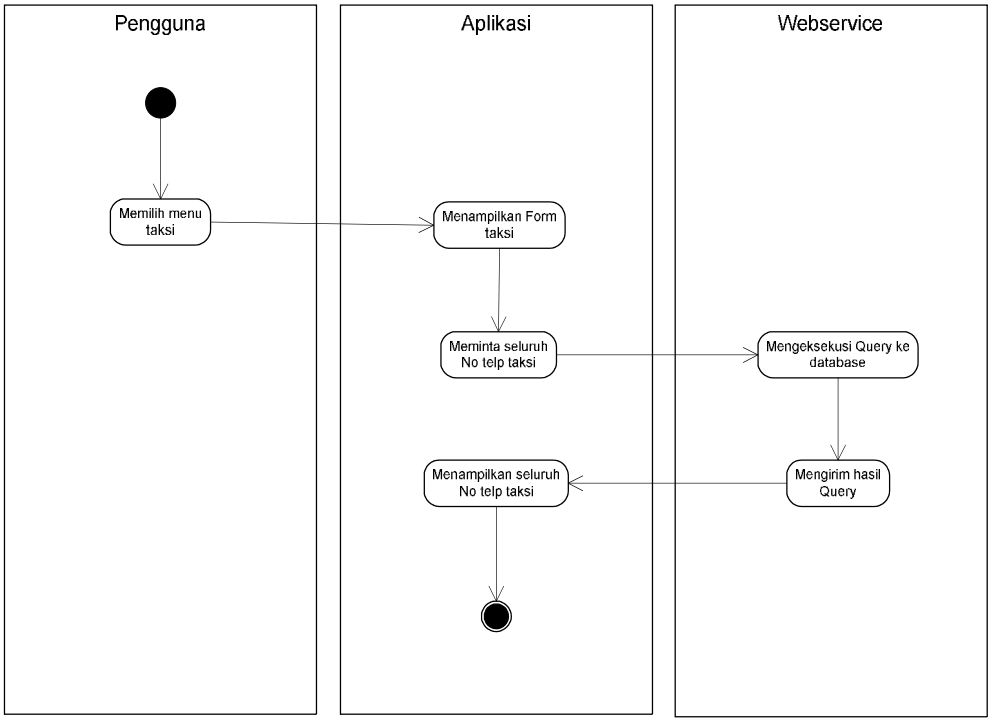


Gambar 3.12 Activity Diagram Pencarian Tujuan dengan Patas

Activity diagram diatas menjelaskan tentang proses pencarian yang terjadi pada patas, dimana proses yang terjadi adalah pengguna memilih menu patas yang

tersedia pada menu pilih kategori, kemudian menampilkan *form* patas selanjutnya pengguna harus memasukkan *keyword* berdasarkan nama tempat tujuan yang dicari selanjutnya mencari nama tempat tujuan ke *webservice*, kemudian pencarian itu diproses dengan mengggunakan *query* ke database selanjutnya mengirim hasil *query* ke aplikasi dan menampilkan hasil pencarian untuk pengguna.

2. Activity Diagram Menampilkan Nomor Telpon Taksi

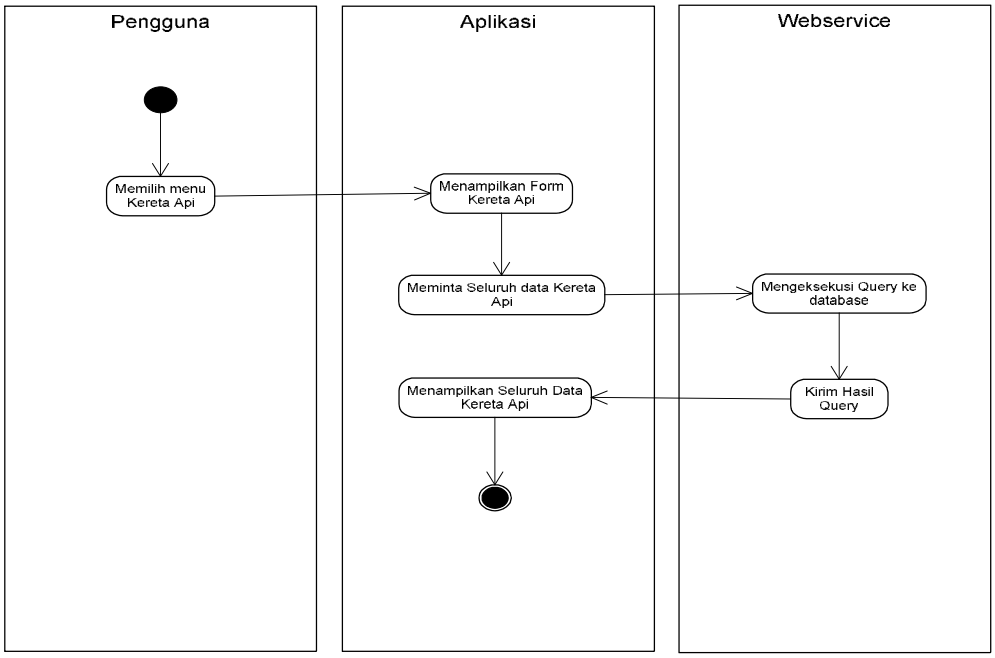


Gambar 3.13 Activity Diagram Menampilkan Nomor Telpon Taksi

Activity diagram diatas mendeskripsikan tentang proses yang terjadi pada menu taksi. Awal proses adalah pengguna memilih menu taksi yang berada pada menu pilih kategori setelah itu menampilkan *form* taksi, pada saat proses

menampilkan *form* taksi secara bersamaan juga terjadi proses meminta seluruh nomor telpon taksi pada *webservice* yang kemudian diproses di *server* dengan menggunakan *query* ke database setelah itu hasil *query* dikirim kembali untuk di tampilkan di *form* taksi kepada pengguna.

3. Activity Diagram Menampilkan Jadwal Kereta Api

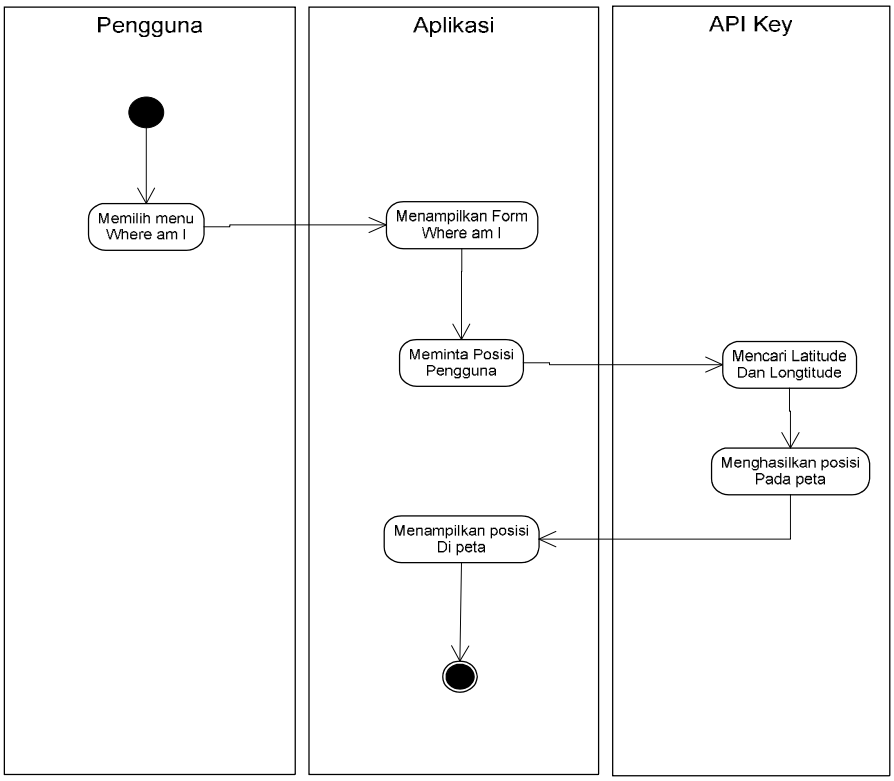


Gambar 3.14 Activity Diagram Menampilkan Jadwal Kereta api

Gambar tersebut menjelaskan tentang aktivitas yang terjadi pada kereta api, dimana awal dari proses aktifitas tersebut adalah pengguna mmeilih menu kereta api yang berada pada menu pilih kategori selanjutnya menampilkan *form* kereta api, dimana pada saat proses untuk menampilkan *form* kereta api terjadi juga proses untuk meminta seluruh jadwal keberangkatan kereta api ke *webservice*, dimana *webservice* menerima proses tersebut kemudian mencari

permintaan dengan menggunakan *query* ke database selanjutnya *query* tersebut dikirim ke aplikasi *frontend* dan hasil *query* tersebut ditampilkan di *form* kereta api untuk pengguna.

4. Activity Diagram Menampilkan Posisi Pengguna

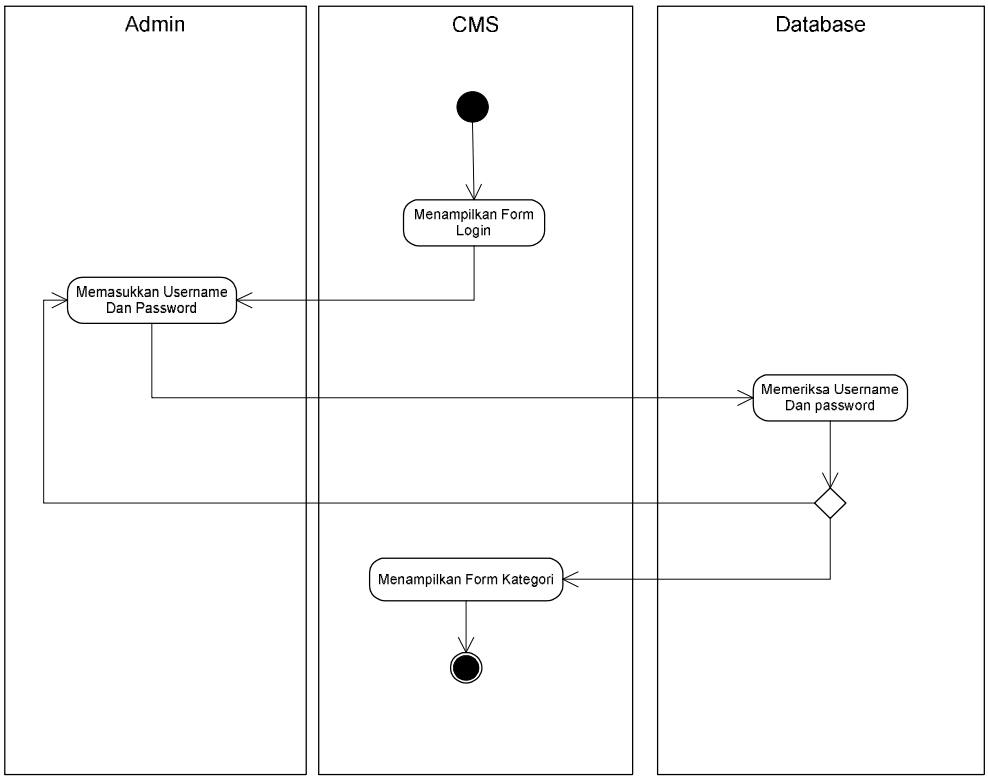


Gambar 3.15 Activity Diagram Menampilkan Posisi Pengguna

Menampilkan posisi pengguna di peta pada *smartphone* dimana aktivitas yang terjadi yang digambarkan oleh activity diagram adalah pengguna memilih menu where am I yang tersedia di menu pilih kategori, kemudia menampilkan *form* where am I dimana pada saat menampilkan *form* ini, terjadi proses yang bersamaan juga dengan mencari keberadaan pengguna dengan menggunakan koneksi GPS lalu GPS mencari titik *latitude* dan *longtitude* dimana titik *latitude*

dan *longtitude* ini berfungsi untuk memberikan sebuah tanda di peta, dalam aplikasi ini peta yang digunakan adalah peta dari google yang bernama google maps dan kemudian peta tersebut ditampilkan di aplikasi untuk pengguna.

5. Activity Diagram Proses *Login*

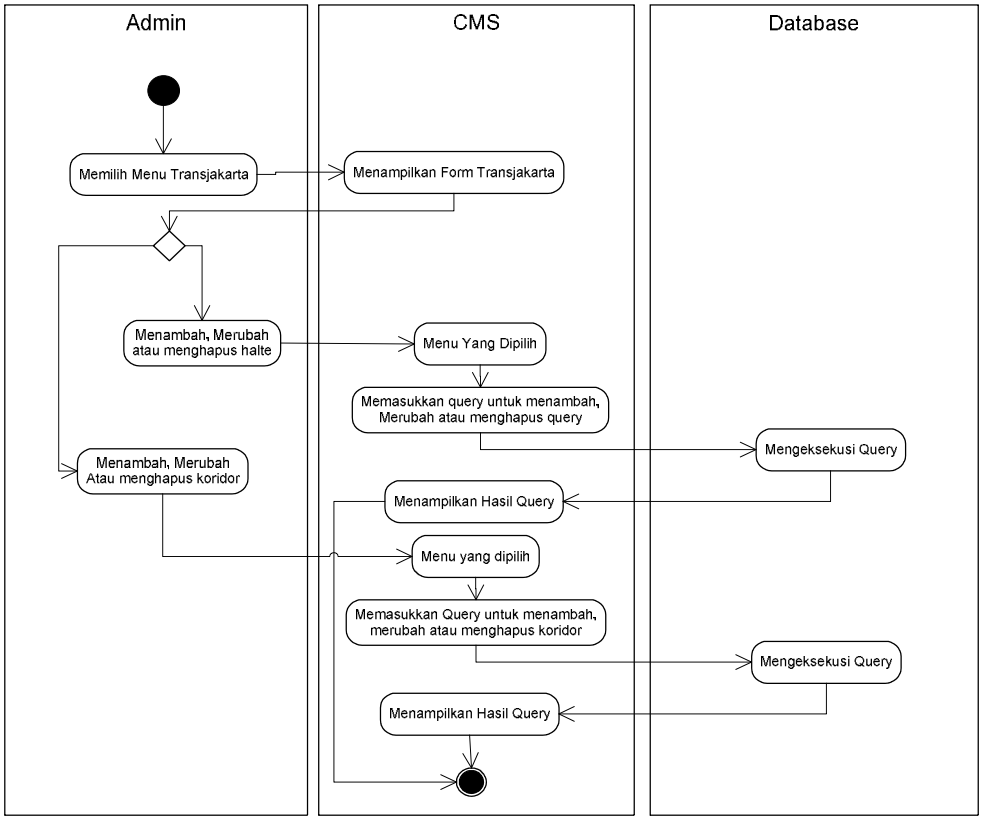


Gambar 3.16 Activity Diagram Proses *Login*

Activity Diagram mendeskripsikan tentang proses *login* yan gterjadi pada aplikasi *Backend*, proses yang terjadi adalah admin memasukkan *Username* dan *Password* yang kemudian akan di periksa di database , jika *Username* dan *Password* salah maka akan kembali pada proses memasukkan *Username* dan

Password dan jika *username* dan *password* benar maka akan menampilkan *form* kategori

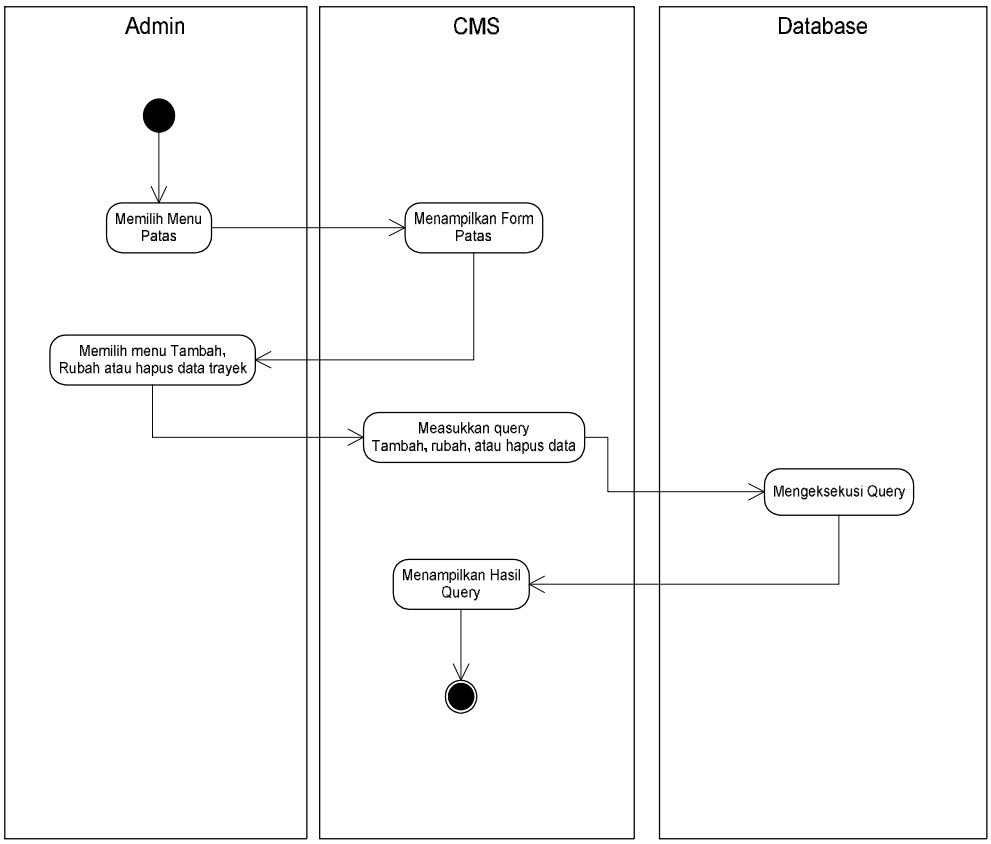
6. Activity Diagram Mengelola Data Transjakarta



Gambar 3.17 Activity Diagram Mengelola Data Transjakarta

Gambar diatas menjelaskan tentang Activity Diagram yang menjelaskan tentang pengelolaan data Transjakarta pada aplikasi *Backend*, terdapat 2 menu di dalamnya yaitu mengelola data halte dan mengelola data koridor, dimana hanya ada *query* menambah, merubah atau menghapus data di setiap masing-masing menu yang kemudian *query* tersebut akan di *query* di database dan ditampilkan di CMS

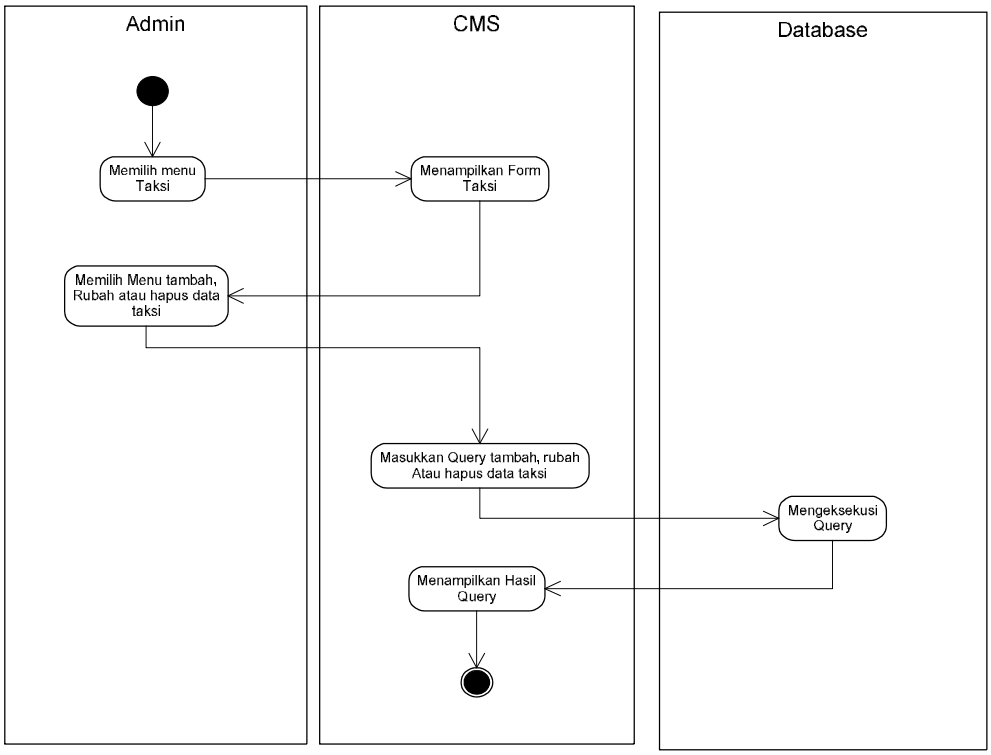
7. Activity Diagram Mengelola Data Patas



Gambar 3.18 Activity Diagram Mengelola Data Patas

Mengelola data patas pada aplikasi *Backend* dimana activity diagram yang terjadi adalah admin memilih menu patas yang kemudian menampilkan *form* patas yang kemudian admin mengelola data, setelah itu CMS akan mengirim *query* dari pengelolaan data yang dilakukan oleh admin ke database kemudian eksekusi *query* terjadi pada database dan hasil dari *query* akan dikirim kembali ke CMS

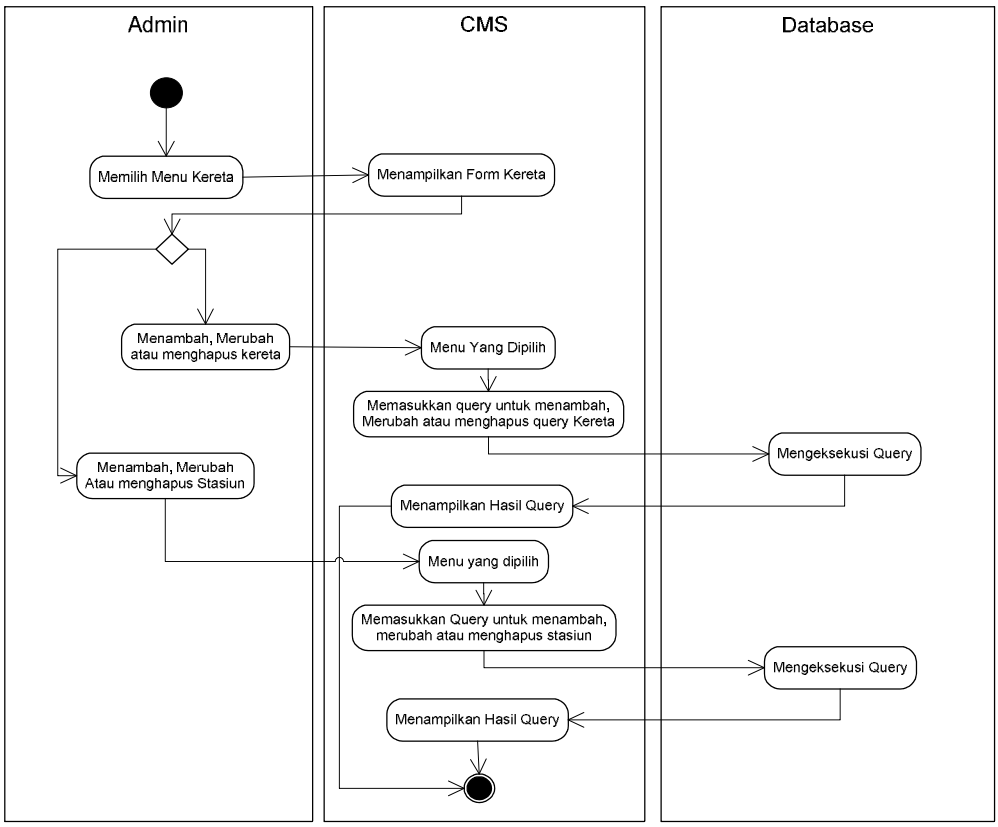
8. Activity Diagram Mengelola data Taksi



Gambar 3.19 Activity Diagram Mengelola Data Taksi

Activity diagram yang terjadi pada pengelolaan data taksi adalah admin memilih menu taksi yang kemudian akan menampilkan *form* taksi setelah itu pengelolaan data pada taksi terjadi yang dilakukan oleh admin yang kemudian pengelolaan data tersebut berubah menjadi *query* di CMS dan proses pengeksekusi *query* terjadi di database yang kemudian hasil *query* akan di tampilkan di CMS

9. Activity Diagram Mengelola Data Kereta Api

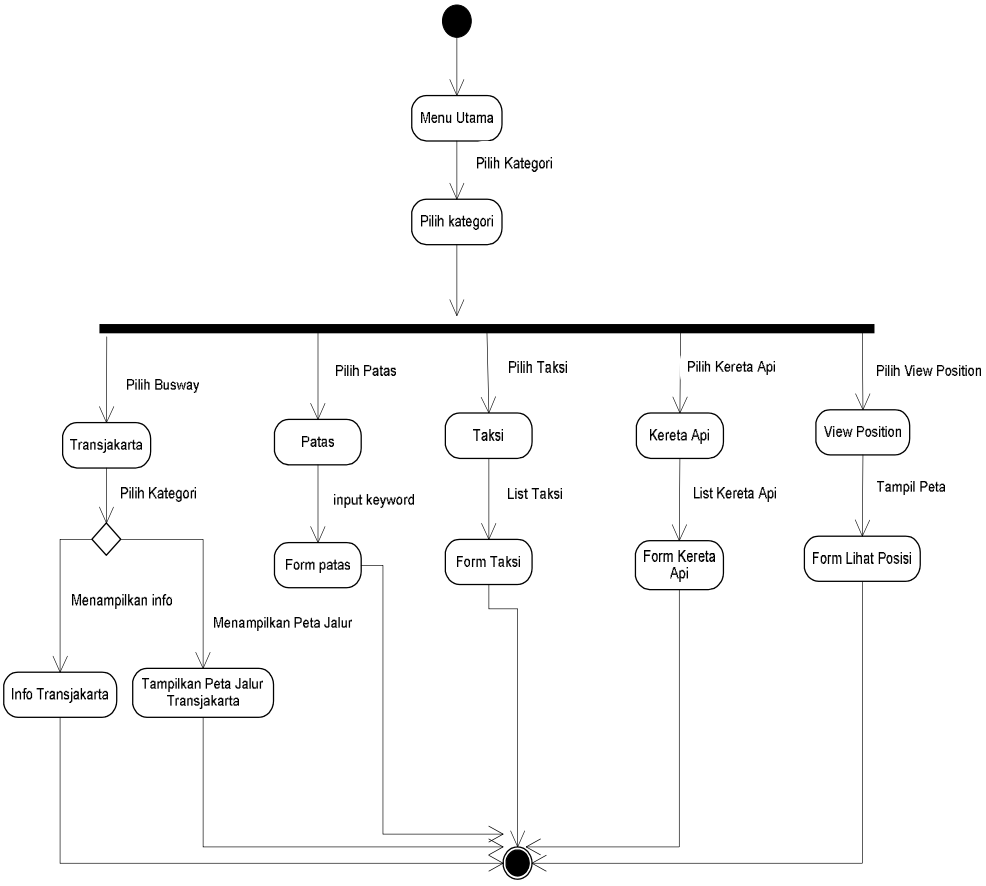


Gambar 3.20 Activity Diagram Mengelola Data Kereta Api

Activity Diagram yang terjadi pada pengelolaan data kereta api yang dilakukan oleh admin adalah admin memilih menu kereta yang kemudian menampilkan *form* kereta dan terdapat 2 pilihan untuk pengelolahan data yaitu pengolahan data kereta dan pengolahan data stasiun, kedua pilihan pengolahan data tersebut akan berubah jadi *query* pada CMS, setelah itu pengeksekusian *query* terjadi pada database yang kemudian akan menampilkan hasil *query* di CMS.

3.2.4. Statechart Diagram

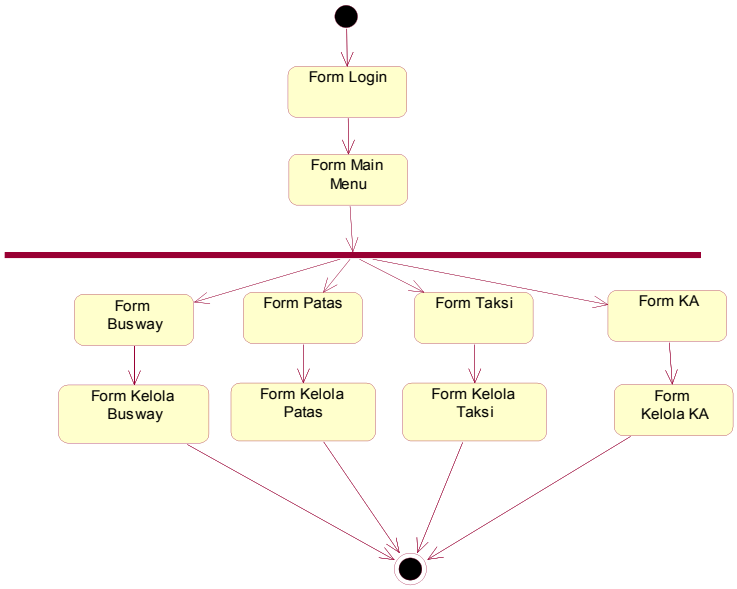
Diagram ini menggambarkan transisis dari perubahan keadaan suatu objek, akibat dari stimulus/*input* yang diterimanya, statechart diagram menelusuri individu-individu obyek melalui keseluruhan daur hidupnya, menspesifikasikan semua urutan yang mungkin dari pesan-pesan yang akan diterima obyek tersebut, bersama-sama dengan tanggapan atas pesan-pesan tersebut.



Gambar 3.21 State Chart Diagram Aplikasi *Frontend*

Gambar state chart diagram diatas mendeskripsikan tentang keadaan awal dari suatu state yang dimulai dari menu utama yang kemudian di menu utama

tersebut terdapat lima kategori yang terdiri dari Transjakarta, Patas, Taksi, Kereta Api dan view position. Pengguna dapat memilih salah satu dari lima kategori tersebut. Selanjutnya jika pengguna memilih Transjakarta maka akan menampilkan *form* Transjakarta dan didalam *form* Transjakarta terdapat dua kategori yaitu tampilkan posisi halte Transjakarta dan tampilkan peta jalur Transjakarta jika memilih tampilkan posisi halte Transjakarta maka akan menampilkan posisi halte Transjakarta yang terdekat dengan pengguna di peta dan jika memilih kategori tampilkan peta jalur Transjakarta maka akan menampilkan peta jalur Transjakarta sesuai dengan tujuan pengguna, selanjutnya jika pengguna memilih patas maka akan menampilkan *form* patas dan yang harus di lakukan oleh pengguna adalah memasukkan *keyword* sesuai dengan nama tempat tujuan kemudian akan menampilkan hasil berdasarkan *keyword* sesuai dengan nama tempat tujuan, lalu jika pengguna memilih taksi akan menampilkan *form* taksi dimana di *form* ini akan menampilkan *list* yang hanya berisi nama perusahaan dan nomor telpon perusahaan yang akan dihubungi, kemudian kalau pengguna memilih kereta api maka akan menampilkan *form* kereta api di *form* ini hanya menampilkan nama kereta api beserta jadwal keberangkatan dan jadwal tiba kereta api, kemudian menu yang terakhir adalah menu where am I aktifitas yang dilakukan di *form* ini adalah menampilkan posisi pengguna pada peta yang menggunakan koneksi ke GPS.

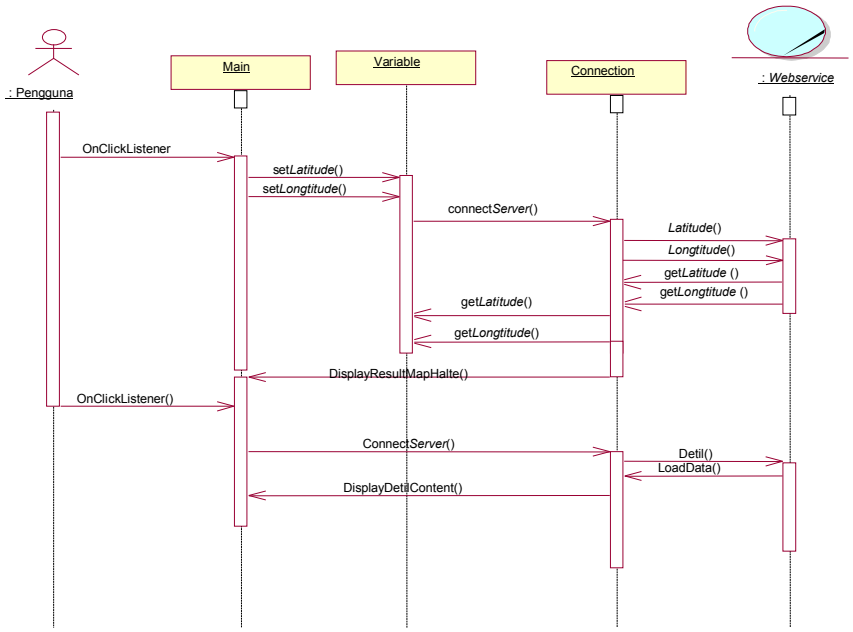


Gambar 3.22 State Chart Diagram Aplikasi *Backend*

3.2.5. Sequence diagram

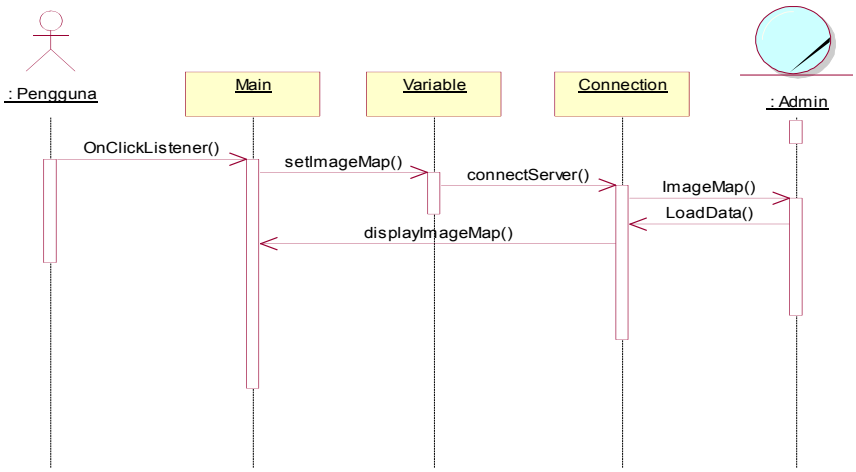
Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Diagram ini menunjukkan sejumlah contoh obyek dan message yang diletakkan diantara obyek-obyek ini didalam use case. Biasa digunakan untuk menggambarkan scenario atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan output tertentu. Diawali dari sebuah aktivitas tertentu, kemudian berproses mengikuti urutan tertentu yang bisa terlihat melalui *message* antar objeknya.

3.2.5.1. Sequence diagram pencarian halte Transjakarta



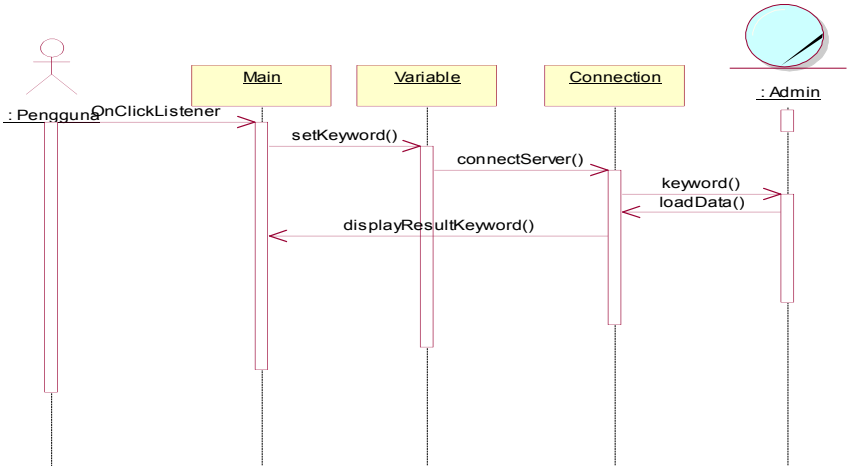
Gambar 3.23 Sequence Diagram Pencarian Halte Transjakarta

3.2.5.2. Sequence Diagram Menampilkan Peta Jalur Transjakarta



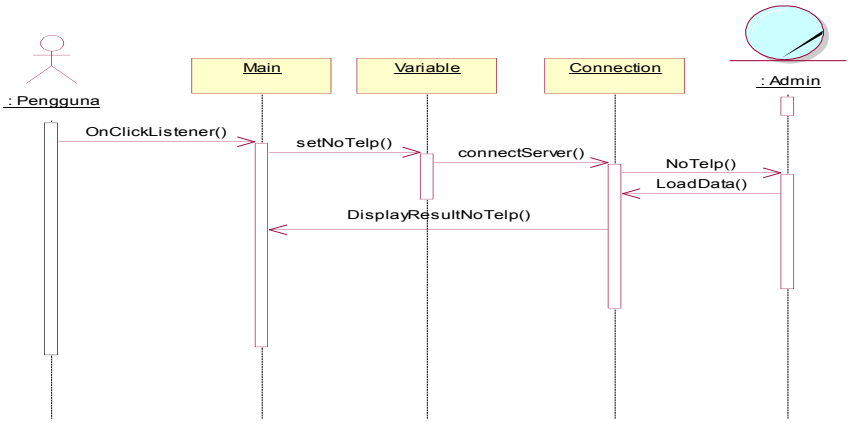
Gambar 3.24 Sequence Diagram Menampilkan Peta Jalur Transjakarta

3.2.5.3. Sequence Diagram Pencarian tujuan dengan Patas



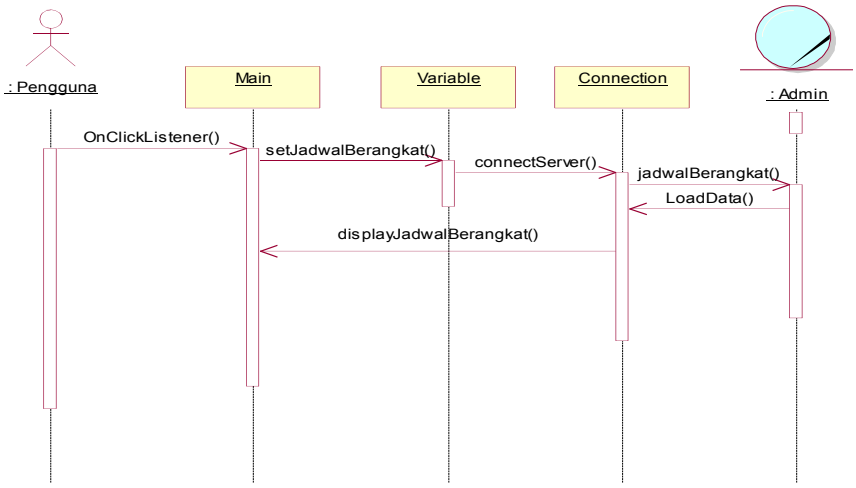
Gambar 3.25 Sequence Diagram Pencarian Tujuan Dengan Patas

3.2.5.4. Sequence Diagram Menampilkan Nomor Telpon Taksi



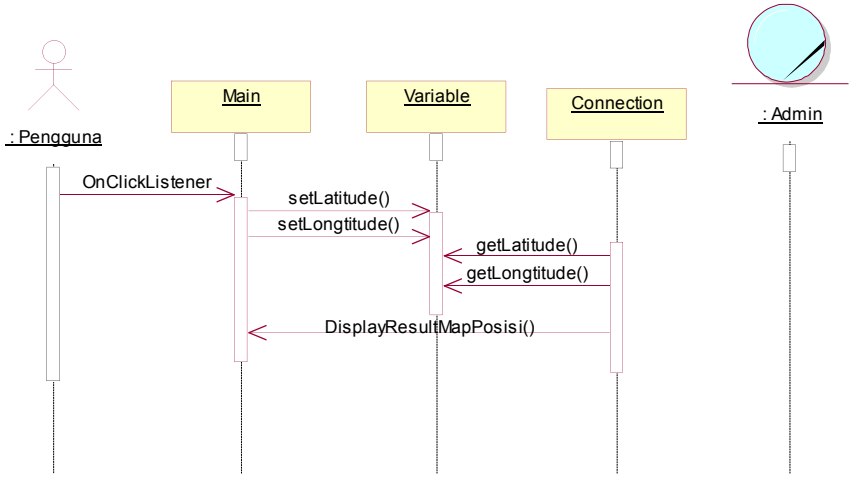
Gambar 3.26 Sequence Diagram menampilkan nomor telpon Taksi

3.2.5.5. Sequence Diagram Menampilkan jadwal kereta api



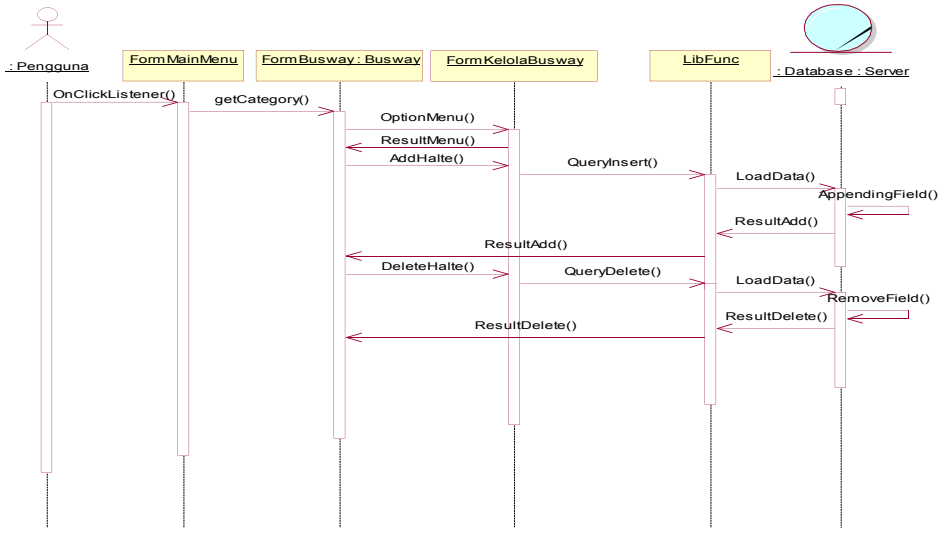
Gambar 3.27 Sequence Diagram Menampilkan Jadwal Kereta Api

3.2.5.6. Sequence Diagram *View Position*



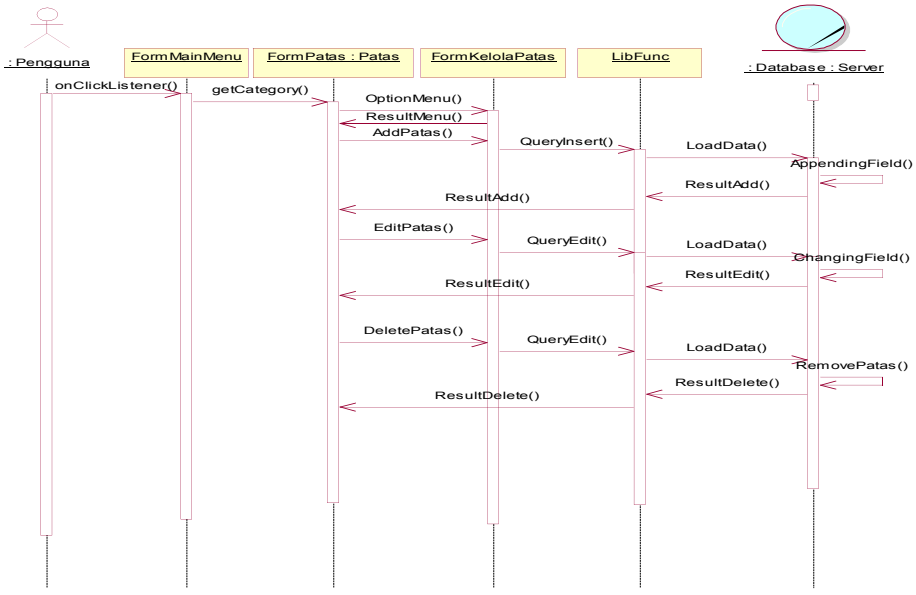
Gambar 3.28 Sequence Diagram *View Position*

3.2.5.7. Sequence Diagram Mengelola Data Transjakarta



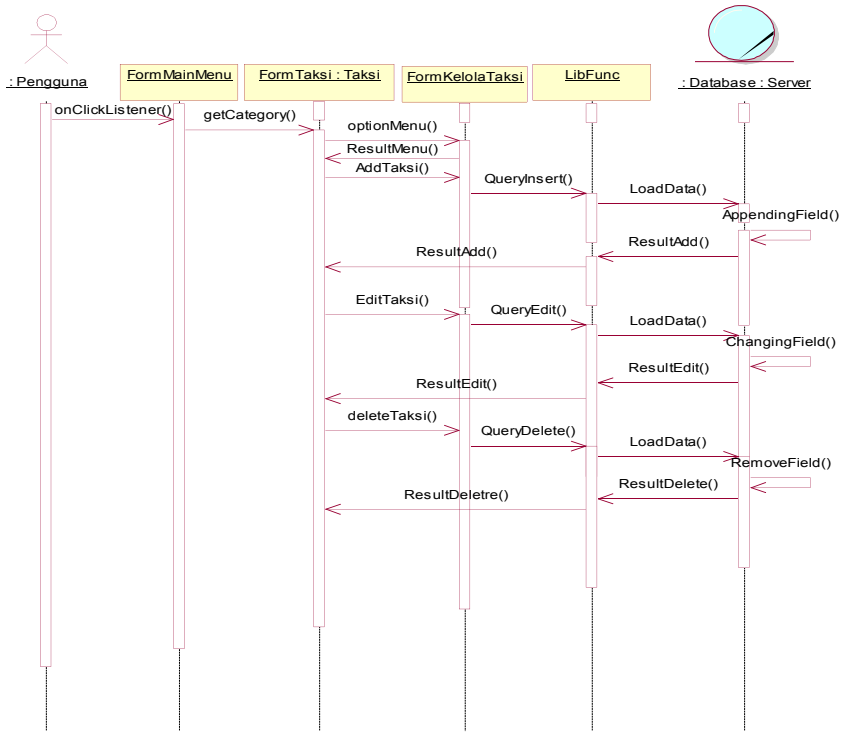
Gambar 3.29 Sequence Diagram pada mengelola data Transjakarta

3.2.5.8. Sequence Diagram Mengelola Data Patas



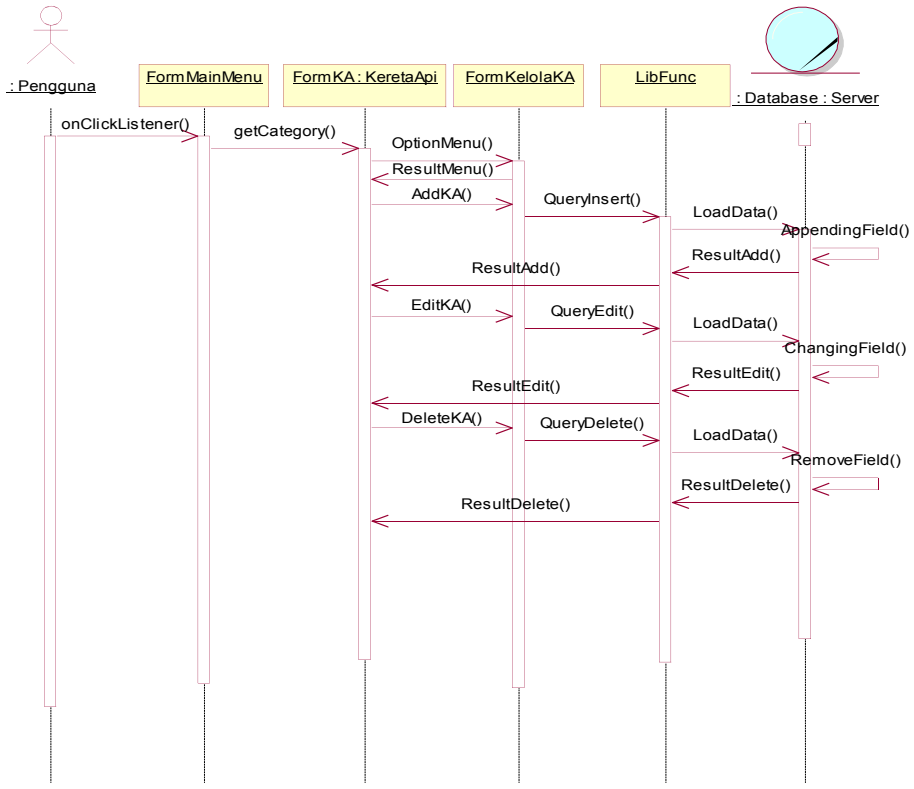
Gambar 3.30 Sequence Diagram pada Mengelola Data Patas

3.2.5.9. Sequence Diagram pada mengelola data taksi



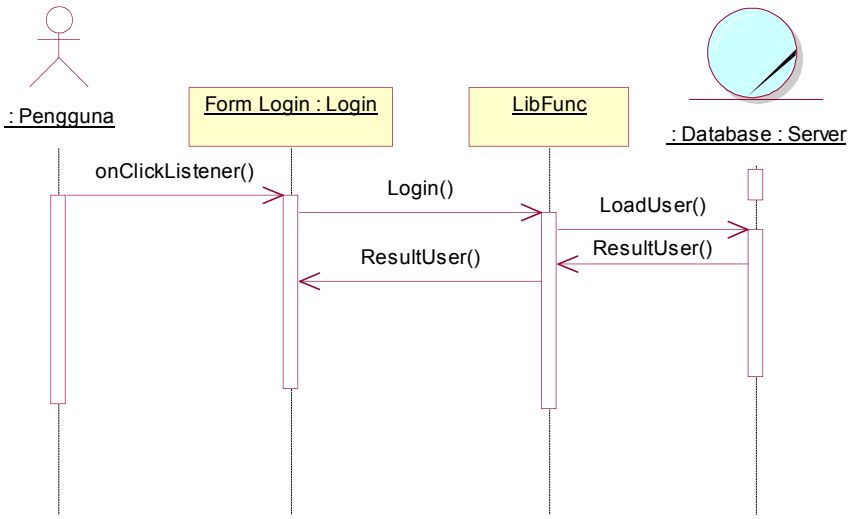
Gambar 3.31 Sequence Diagram pada Mengelola Data Taksi

3.2.5.10. Sequence Diagram pada Mengelola Data Kereta Api



Gambar 3.32 Sequence Diagram pada Mengelola Data Kereta Api

3.2.5.11. Sequence Diagram proses *Login*

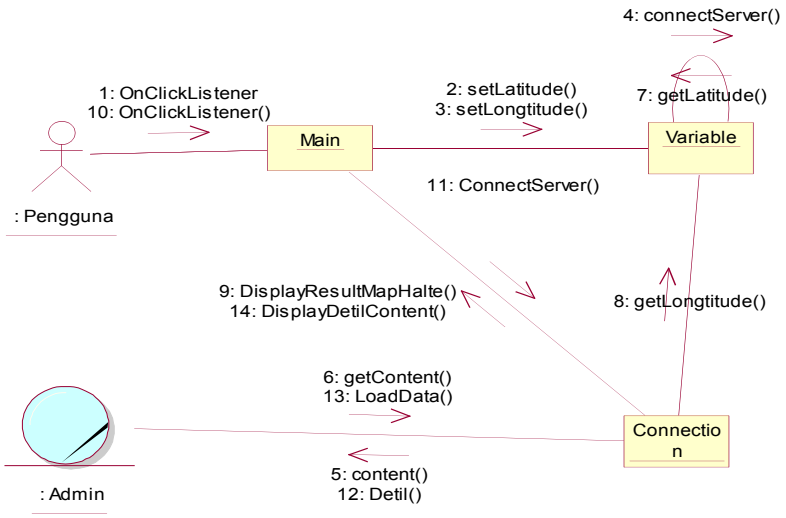


Gambar 3.33 Sequence Diagram Proses *Login*

3.2.6. Collaboration Diagram

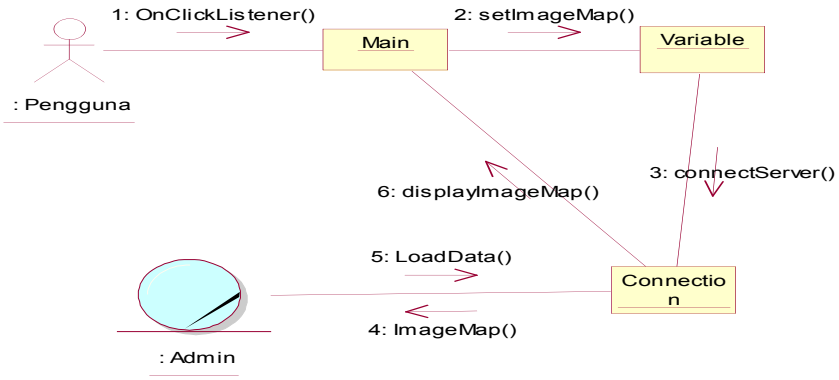
Collaboration diagram adalah perluasan dari obyek diagram. (obyek diagram menunjukkan obyek-obyek dan hubungannya satu dengan yang lain). Collaboration diagram menunjukkan message-message obyek yang dikirimkan satu sama lain. Antara collaboration diagram dan sequence diagram bisa saling mengisi. Dengan demikian pada collaboration diagram kita bisa tambah nomor urut pada tabel sebuah message untuk menunjukkan urutan in*form*asi.

3.2.6.1. Collaboration Diagram Mencari Halte Transjakarta



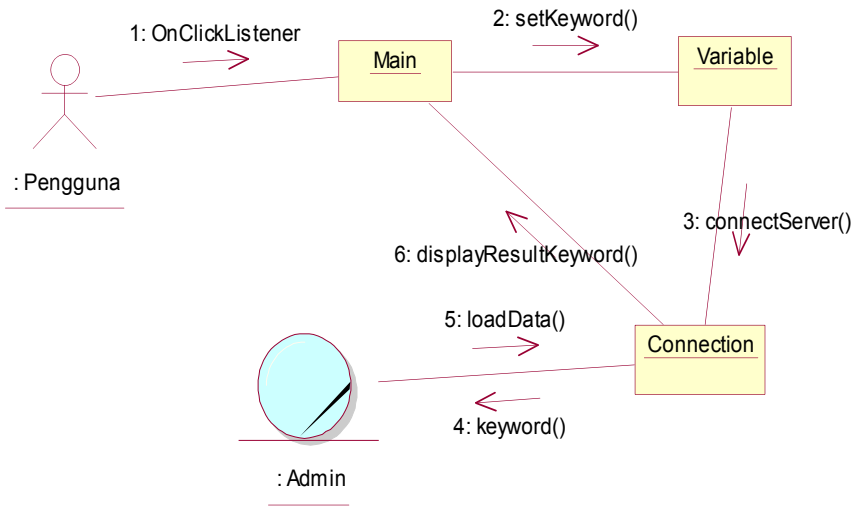
Gambar 3.34 Collaboration Diagram Pencarian Halte Transjakarta

3.2.6.2. Collaboration Diagram Menampilkan Peta Jalur Transjakarta



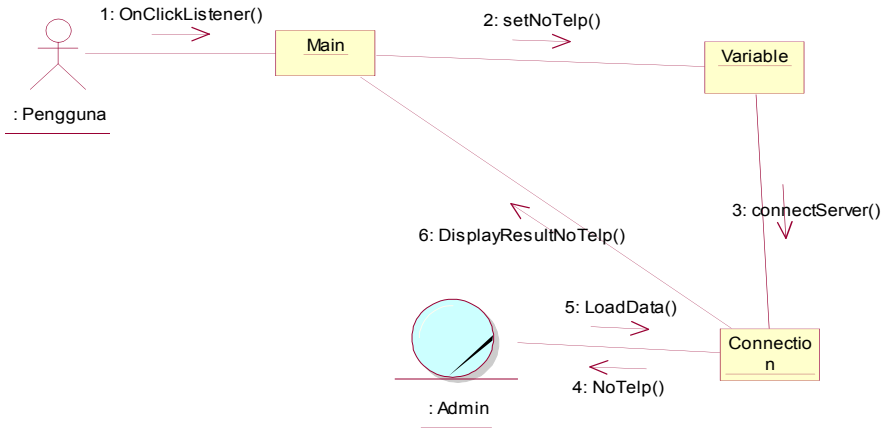
Gambar 3.35 Collaboration Diagram Menampilkan Peta Jalur Transjakarta

3.2.6.3. Collaboration Diagram Pencarian Tujuan dengan Patas



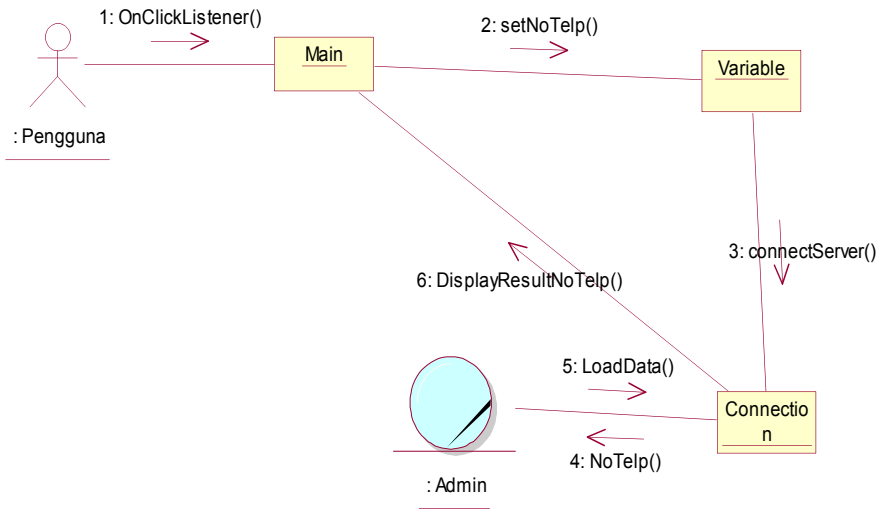
Gambar 3.36 Collaboration Diagram Pencarian tujuan dengan Patas

3.2.6.4. Collaboration Diagram Menampilkan Nomor Telpon Taksi



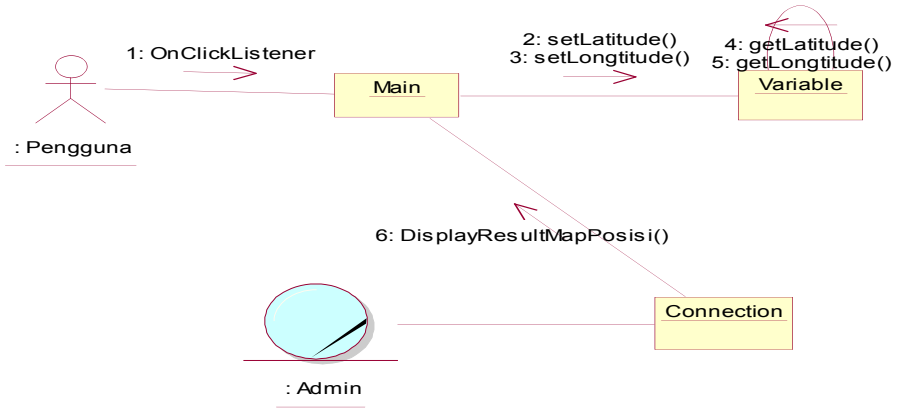
Gambar 3.37 Collaboration Diagram Menampilkan Nomor Telpon Taksi

3.2.6.5. Collaboration Diagram Menampilkan Jadwal Kereta Api



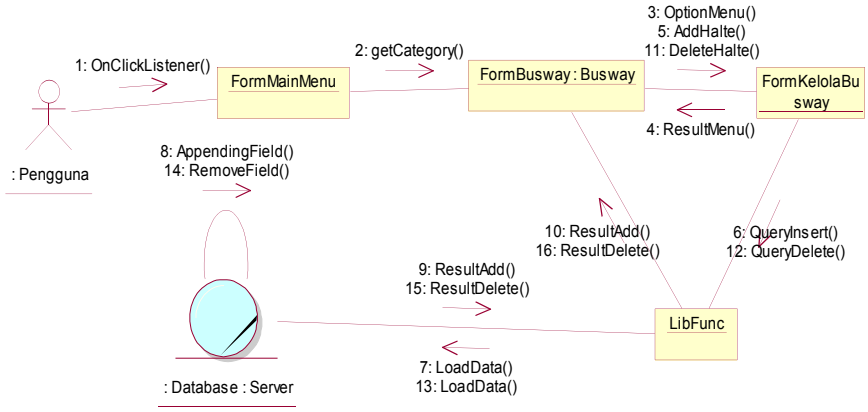
Gambar 3.38 Collaboration Diagram Menampilkan Jadwal Kereta Api

3.2.6.6. Collaboration Diagram *View Position*



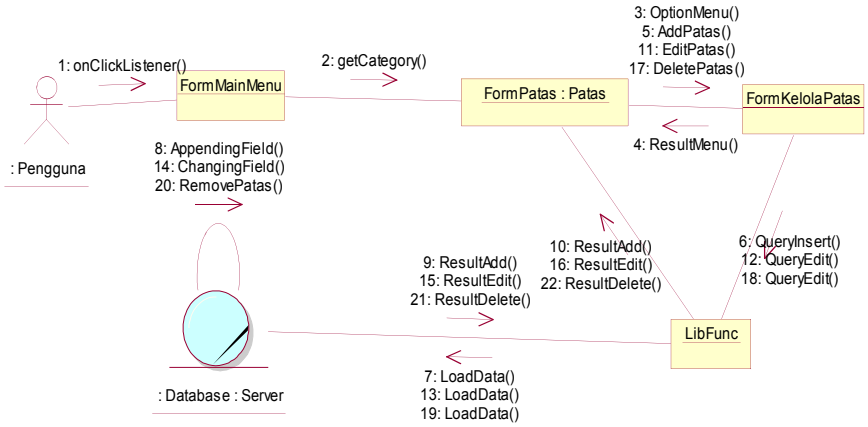
Gambar 3.39 Collaboration Diagram *View Position*

3.2.6.7. Collaboration Diagram Mengelola Data Transjakarta



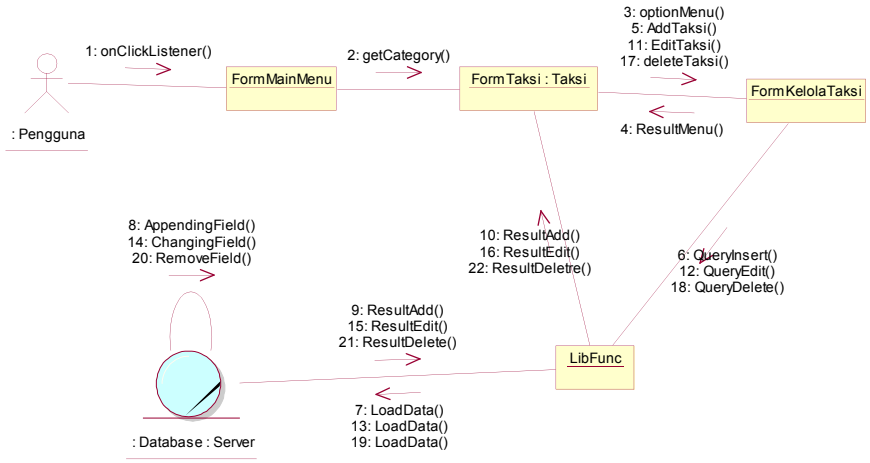
Gambar 3.40 Collaboration Diagram pada Mengelola Data Transjakarta

3.2.6.8. Collaboration Diagram pada Mengelola Data Patas



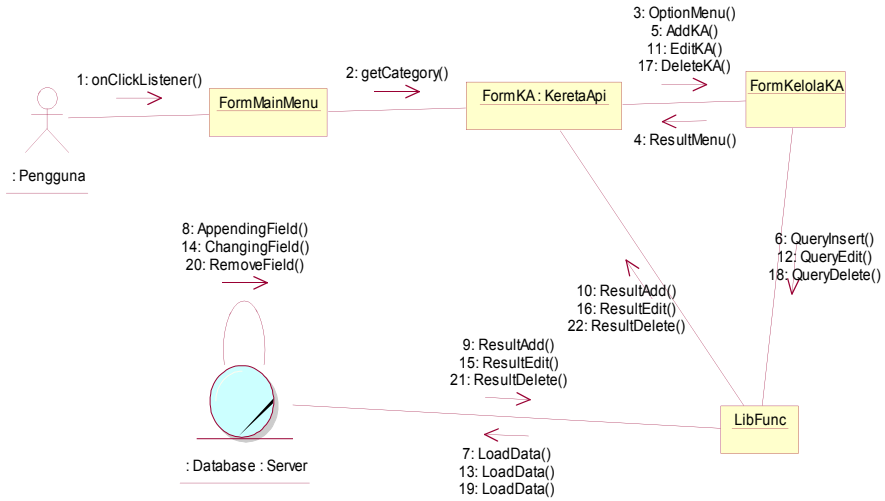
Gambar 3.41 Collaboration Diagram pada mengelola Data Patas

3.2.6.9. Collaboration Diagram pada mengelola data taksi



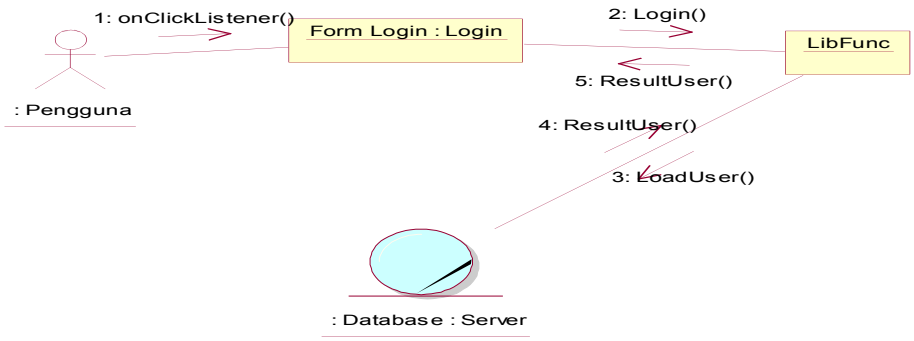
Gambar 3.42 Collaboration Diagram pada mengelola data taksi

3.2.6.10. Collaboration Diagram pada mengelola data kereta api



Gambar 3.43 Collaboration Diagram pada mengelola data kereta api

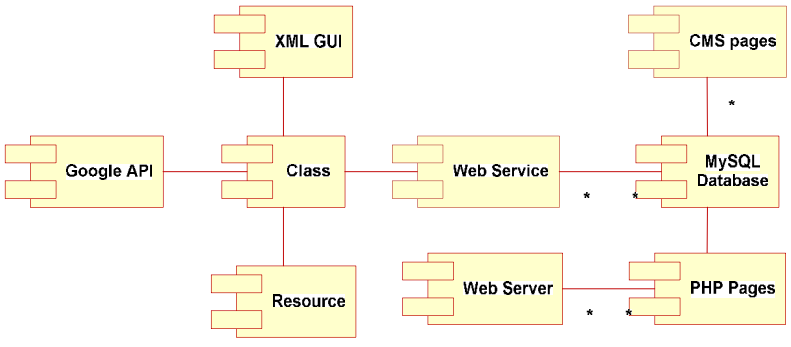
3.2.6.11. Collaboration Diagram proses *Login*



Gambar 3.44 Collaboration Diagram Proses *Login*

3.2.7. Component Diagram

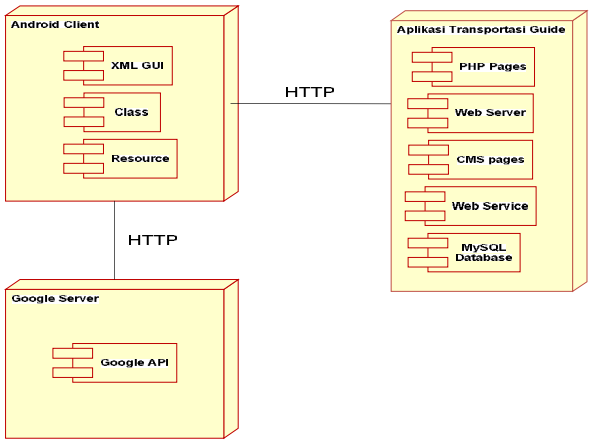
Component software adalah bagian fisik dari sebuah sistem, karena menetap di computer, bukan di benak para analis. Komponen bisa berupa tabel, file data, file exe, dan lain-lain. Diagram ini menggambarkan struktur dan hubungan antara component peranti lunak, termasuk ketergantungan (*dependency*). Diataranhya modul berisi kode, baik berisi source kode, *binary*, *library*, *executable*. Compoenent diagram ditunjukkan pada gambar berikut



Gambar 3.45 Component Diagram

3.2.8. Deployment Diagram

Deployment diagram menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian software yang berjalan pada bagian-bagian hardware. Deployment diagram menggambarkan detail bagaimana komponen dibentuk dan didistribusikan (deploy) dalam infrastruktur sistem. Dimana komponen akan terletak pada mesin, *server* atau peranti keras. Bagaimana jaringan pada lokasi tersebut, misalnya *server*, client dan hal-hal lain yang bersifat fisik.



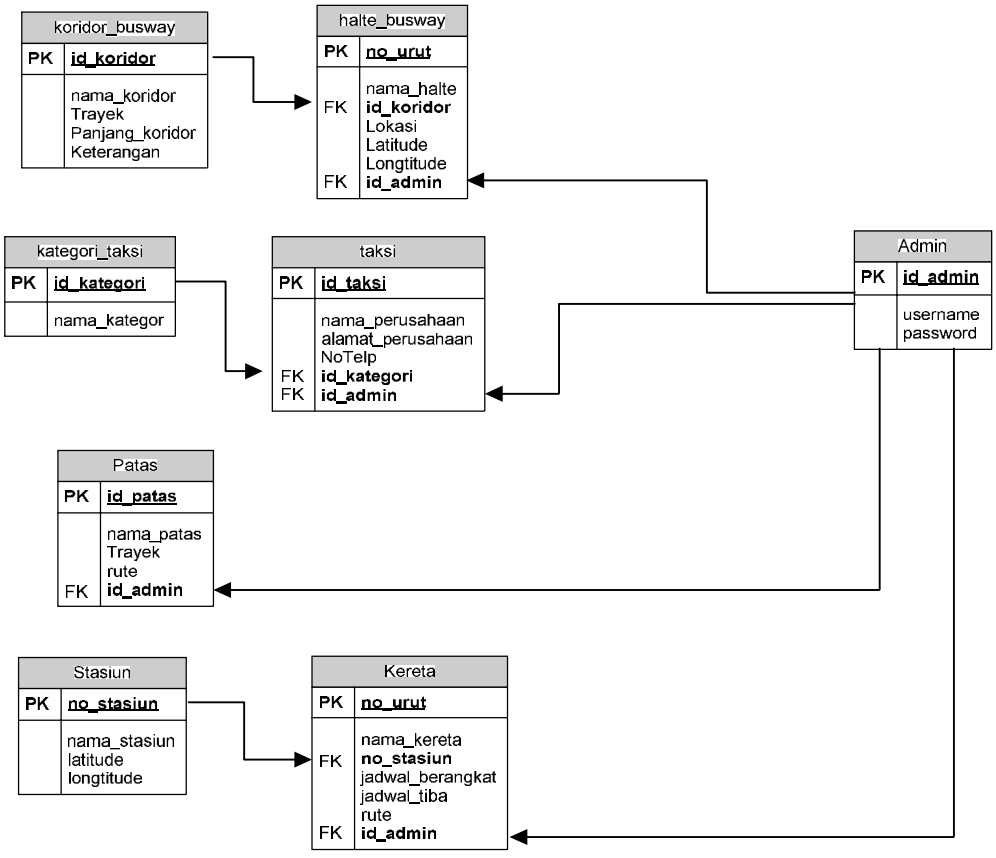
Gambar 3.46 Deployment Diagram

3.3. Perancangan Sistem

Perancangan sistem bertujuan untuk menspesifikasikan aspek-aspek teknik yang menjadi solusi dalam perencanaan. Pada tahap ini perancanagn akan didefinisikan secara detail untuk mengatasi masalah-masalah yang lebih teknis, berkaitan dengan kegiatan implementasi seperti perancangan arsitektur sistem, perancangan basis data, dan perancangan antarmuka.

3.3.1. Skema Relasi

Relasi antar tabel merupakan gabungan antar *file* yang mempunyai kunci utama yang sama, sehingga *file-file* tersebut menjadi satu kesatuan yang dihubungkan oleh *field* kunci (*Primary Key*). Pada proses ini elemen-elemen data dikelompokkan menjadi satu *file database* beserta *entitas* dan hubungannya. Skema relasi ini dapat dilihat pada gambar dibawah ini.



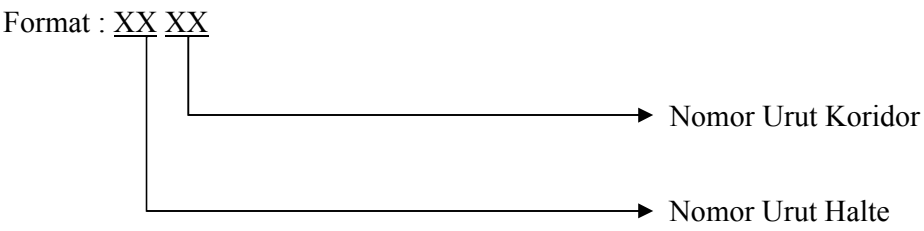
Gambar 3.47 Skema Relasi Aplikasi Transportasi *Guide*

3.3.2. Kodi fikasi

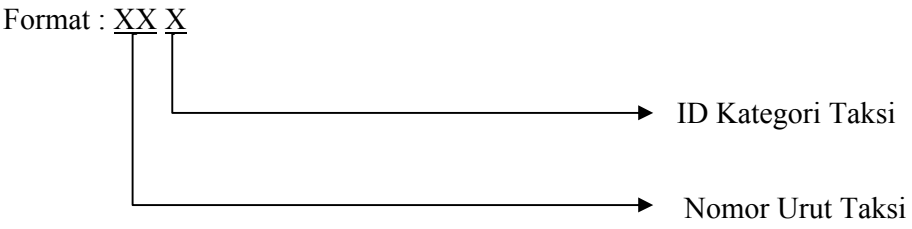
Pengkodean dibuat untuk mengidentifikasi suatu objek secara lebih singkat. Dengan dibuat pengkodean, kesalahan dalam mengidentifikasi objek dapat dikurangi dan berguna untuk mengelompokkan data. Tujuan pengkodean adalah untuk mempermudah proses pencarian data guna penyajian informasi.

Adapun pengkodean dalam aplikasi transportasi guide pada aplikasi *backend* yang digunakan adalah sebagai berikut :

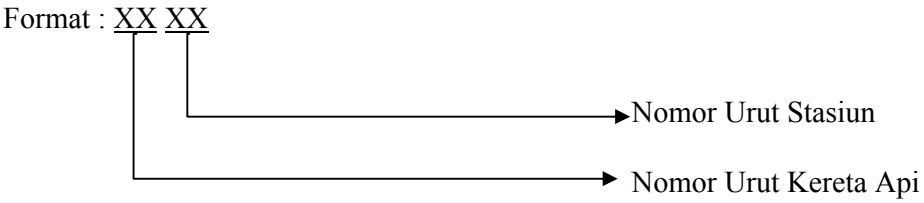
a. Kode halte Transjakarta



b. Kode taksi

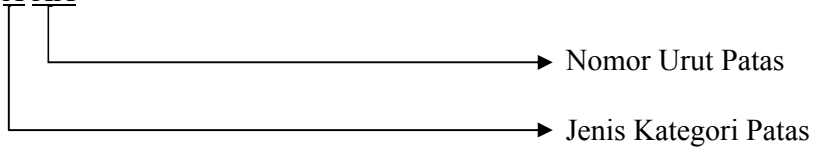


c. Kode Kereta Api



d. Kode Patas

Format : X XX



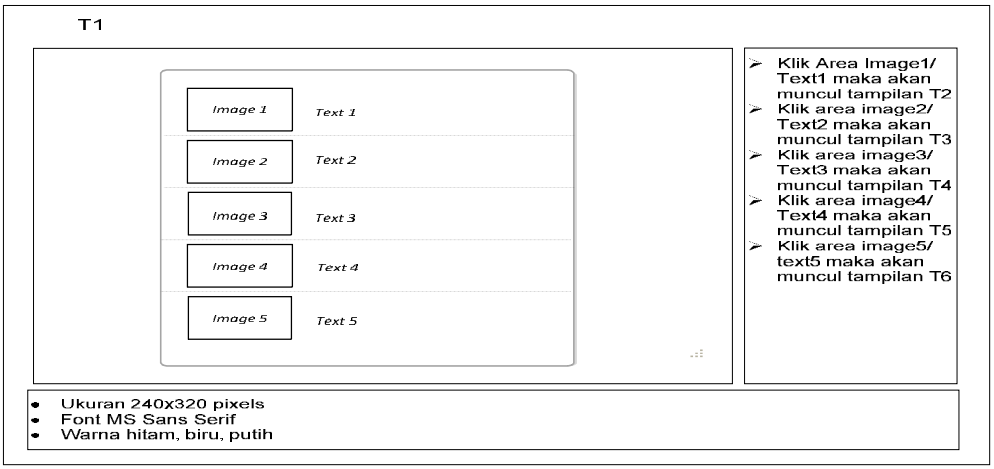
3.3.3. Perancangan Grafis Antar Muka

Tahap perancangan desain bertujuan untuk mencari bentuk yang optimal dari aplikasi yang akan dibangun dengan pertimbangan factor-faktor permasalahan dan kebutuhan yang ada pada sistem seperti yang telah ditetapkan pada tahap analisis. Dalam tahap ini upaya yang dilakukan yaitu dengan cara mengkombinasikan penggunaan teknologi perangkat keras dan perangkat lunak yang tepat sehingga diperoleh yang optimal dan mudah diimplementasikan.

Perancangan grafis dan antarmuka terdiri dari dua bagian utama yaitu disain grafis dan antarmuka *frontend application* dan *backend application*.

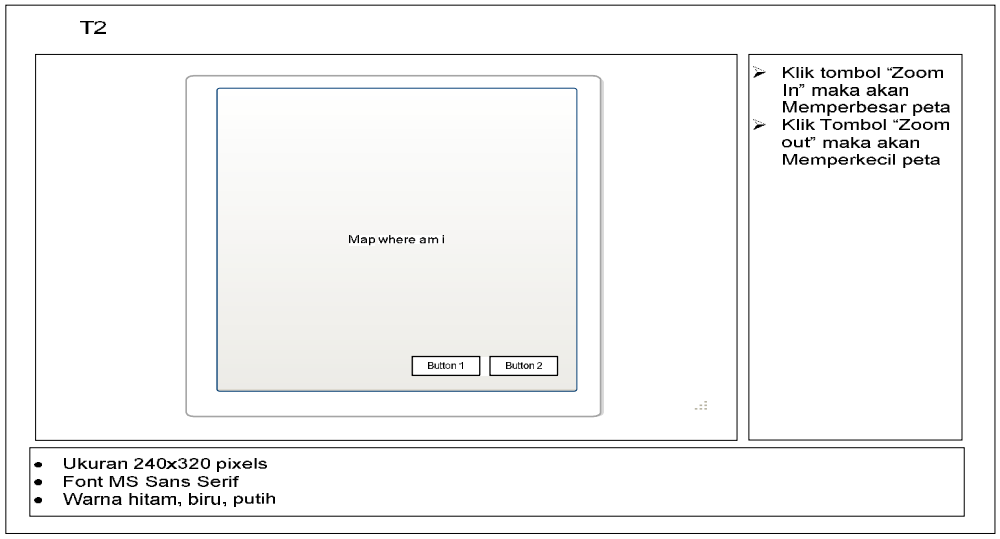
3.3.4.1. Perancangan Antar Muka *Frontend Application*

3.3.4.1.1. Perancangan Antar Muka Menu Utama



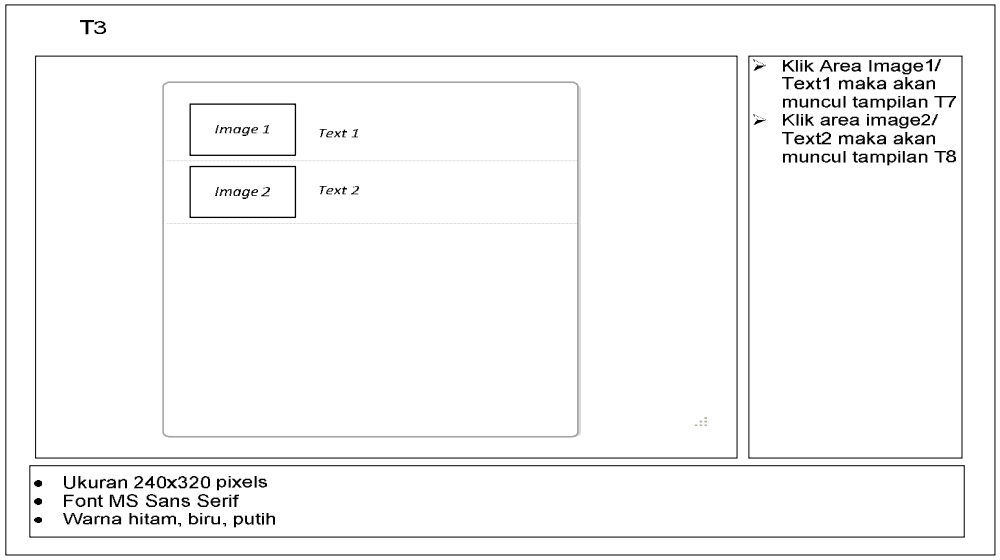
Gambar 3.48 Perancangan Antar Muka Menu Utama

3.3.4.1.2. Perancangan antar muka menu where am i



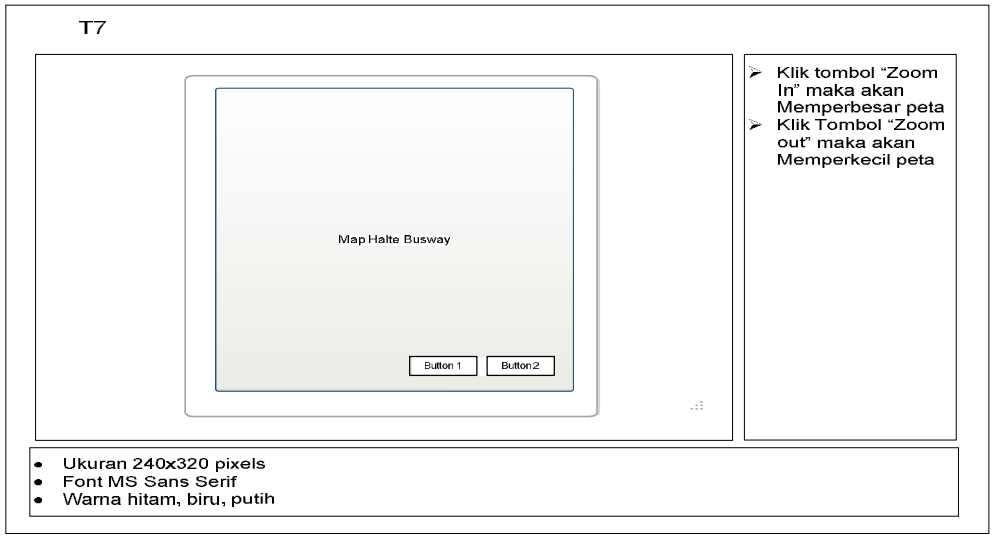
Gambar 3.49 Perancangan antar muka menu where am i

3.3.4.1.3. Perancangan antar muka menu Transjakarta



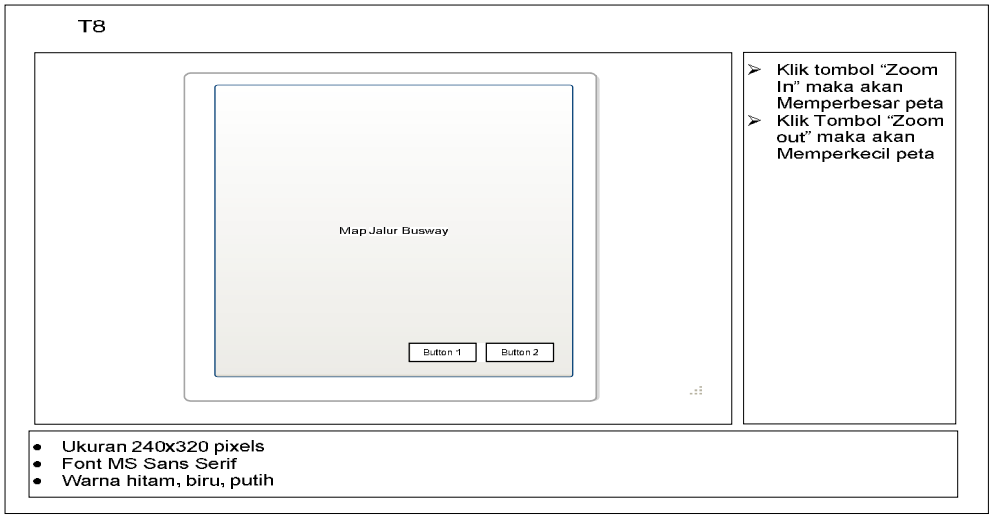
Gambar 3.50 Perancangan antar muka menu Transjakarta

3.3.4.1.4. Perancangan antar muka menu halte Transjakarta



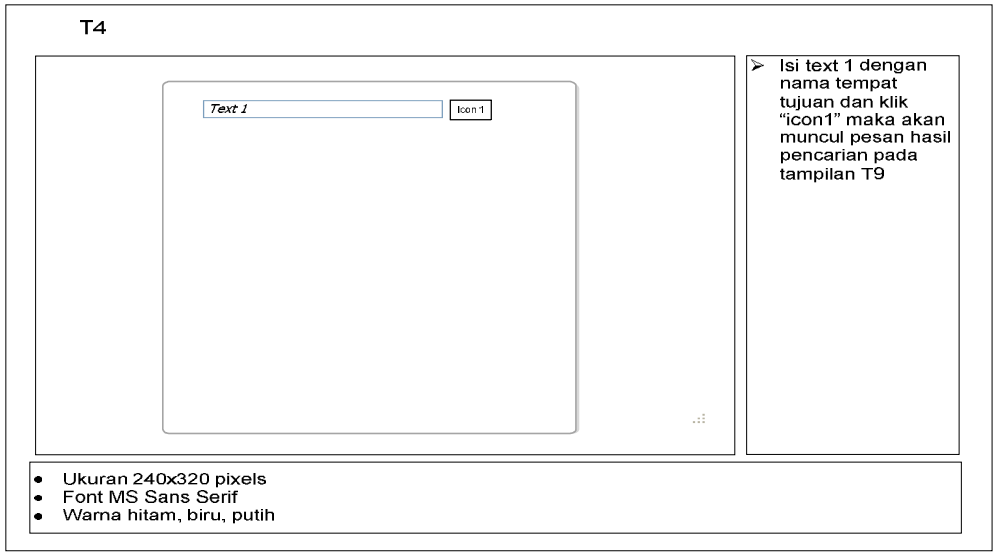
Gambar 3.51 Perancangan antar muka menu halte Transjakarta

3.3.4.1.5. Perancangan antar muka menu jalur Transjakarta



Gambar 3.52 Perancangan antar muka menu jalur Transjakarta

3.3.4.1.6. Perancangan antar muka menu patas



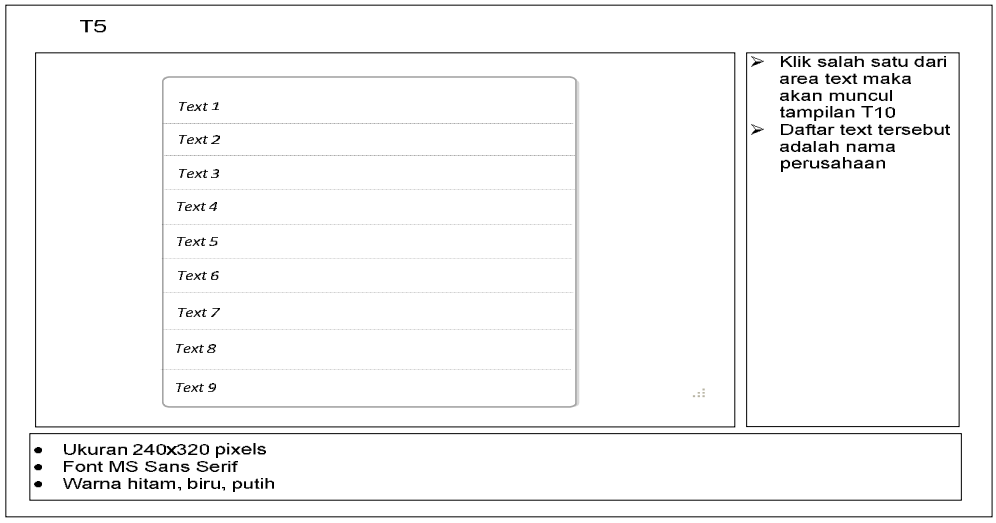
Gambar 3.53 Perancangan antar muka menu patas

3.3.4.1.7. Perancangan antar muka hasil pencarian patas



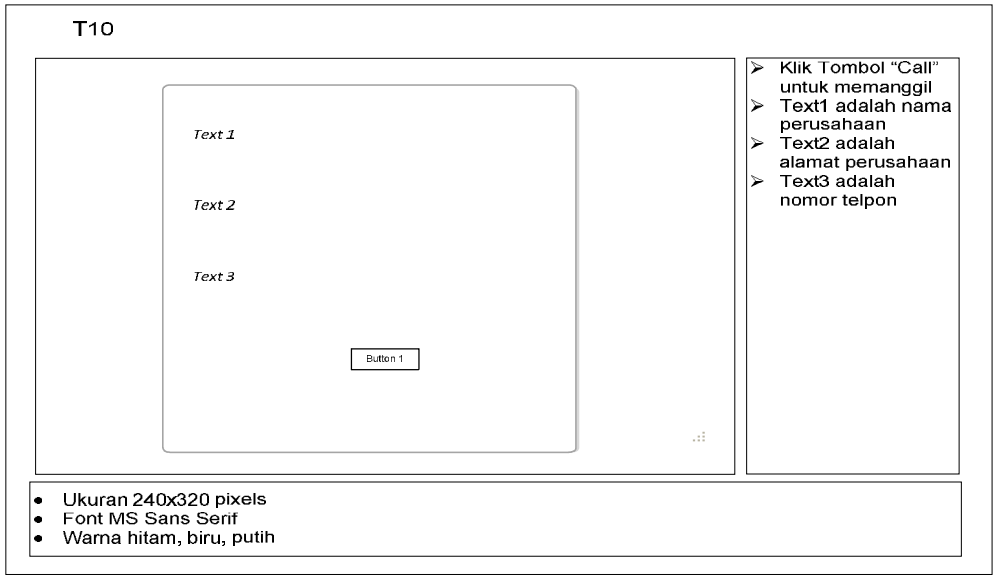
Gambar 3.54 Perancangan antar muka hasil pencarian patas

3.3.4.1.8. Perancangan antar muka menu taksi



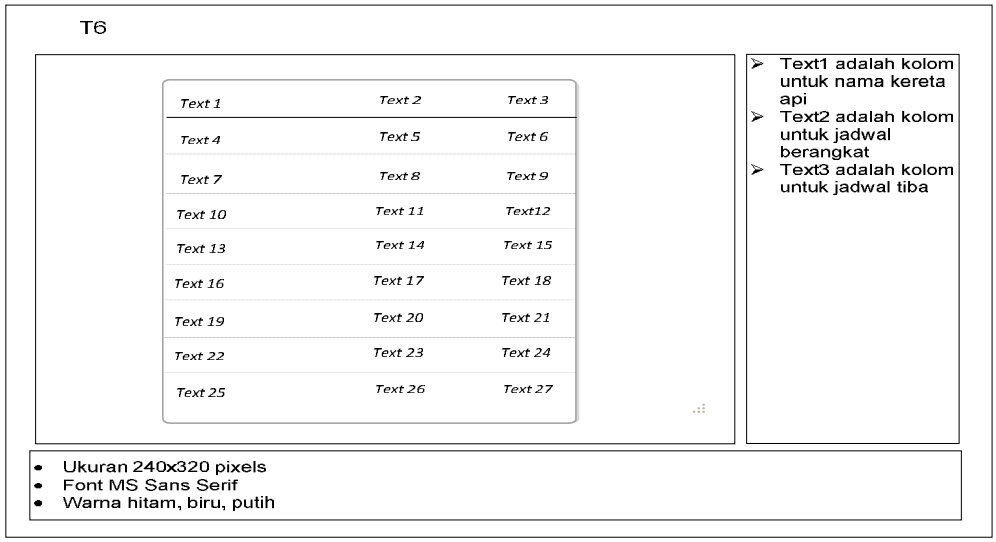
Gambar 3.55 Perancangan antar muka menu taksi

3.3.4.1.9. Perancangan antar muka detail taksi



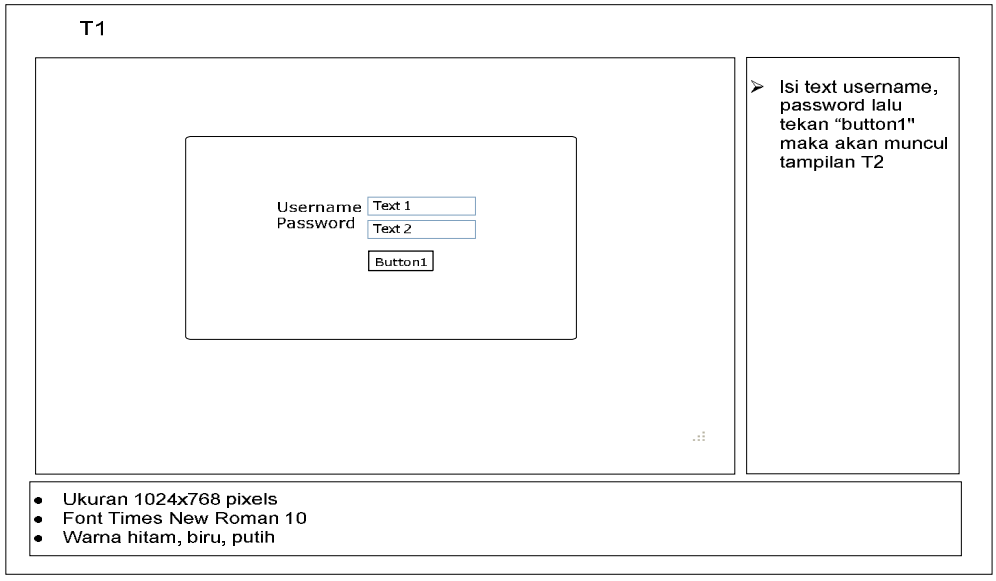
Gambar 3.56 Perancangan antar muka detail taksi

3.3.4.1.10. Perancangan antar muka menu kereta api



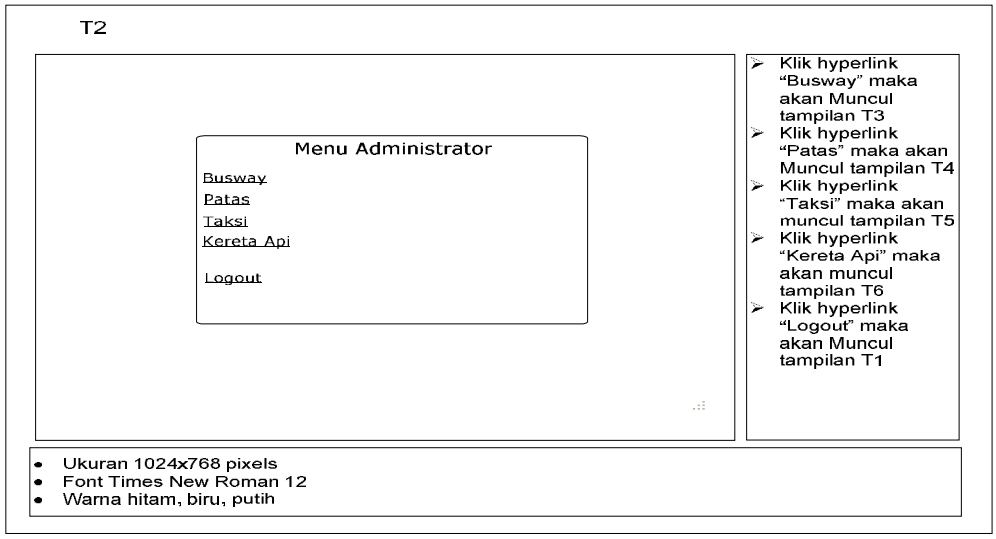
Gambar 3.57 Perancangan antar muka menu kereta api

3.3.4.1.11. Perancangan Antar Muka Halaman *Login* untuk Admin



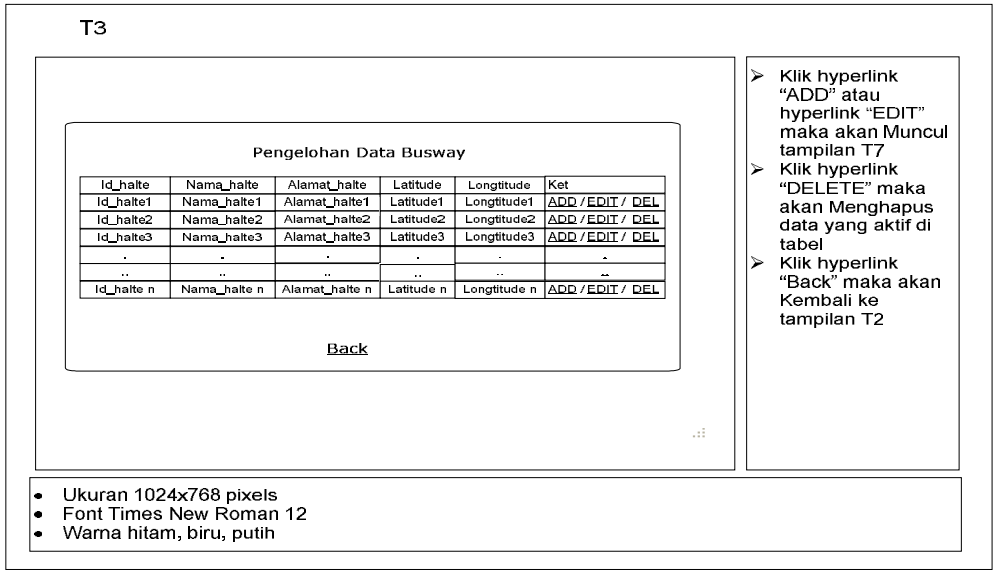
Gambar 3.58 Perancangan Antar Muka Menu *Login*

3.3.4.1.12. Perancangan Antar Muka Halaman Pilih Kategori



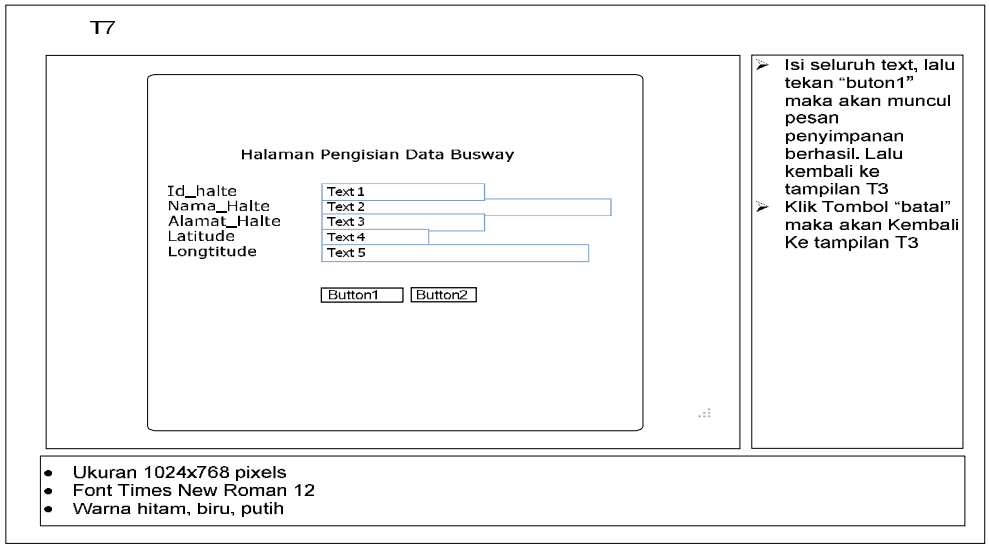
Gambar 3.59 Perancangan Antar Muka Menu Pilih Kategori

3.3.4.1.13. Perancangan Antar Muka Halaman Data Transjakarta



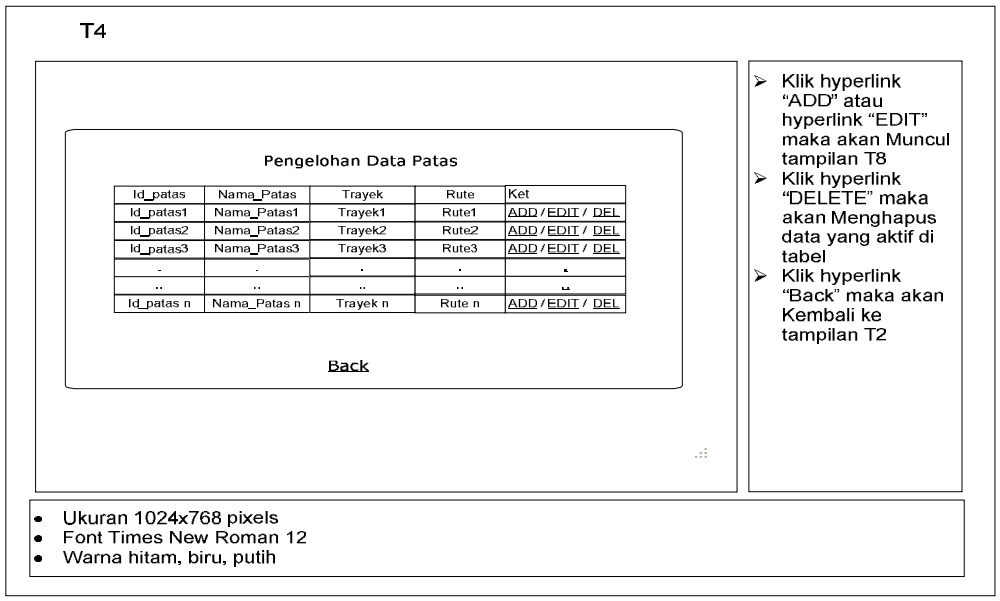
Gambar 3.60 Perancangan Antar Muka Halaman Data Transjakarta

3.3.4.1.14. Perancangan Antar Muka Halaman Kelola Data Transjakarta



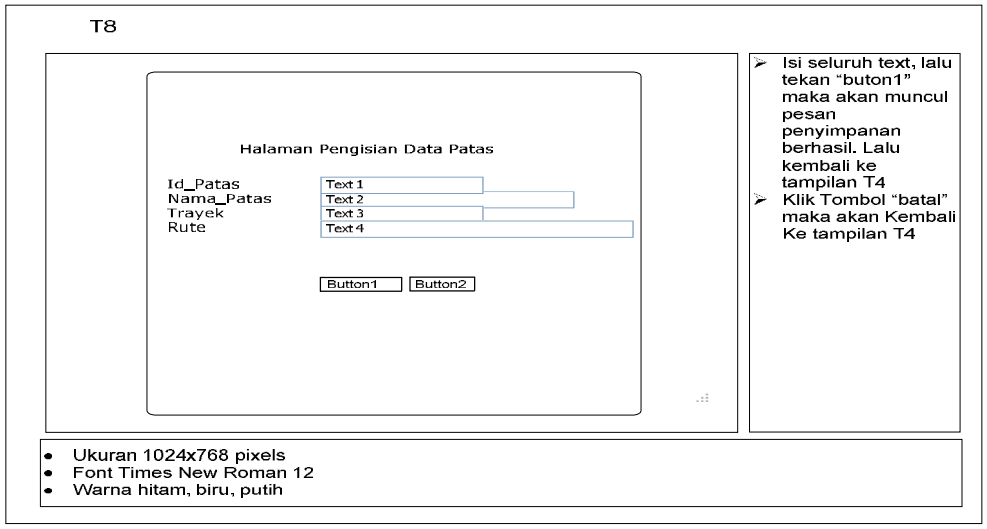
Gambar 3.61 Perancangan Antar Muka Halaman Kelola Data Transjakarta

3.3.4.1.15. Perancangan Antar Muka Halaman Data Patas



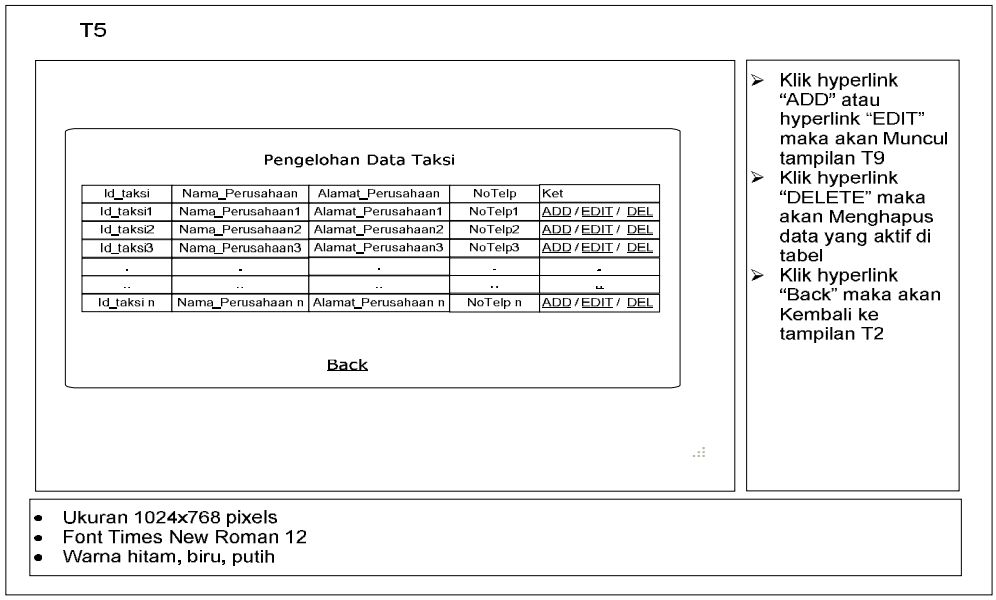
Gambar 3.62 Perancangan Antar Muka Halaman Data Patas

3.3.4.1.16. Perancangan Antar Muka Halaman Kelola Data Patas



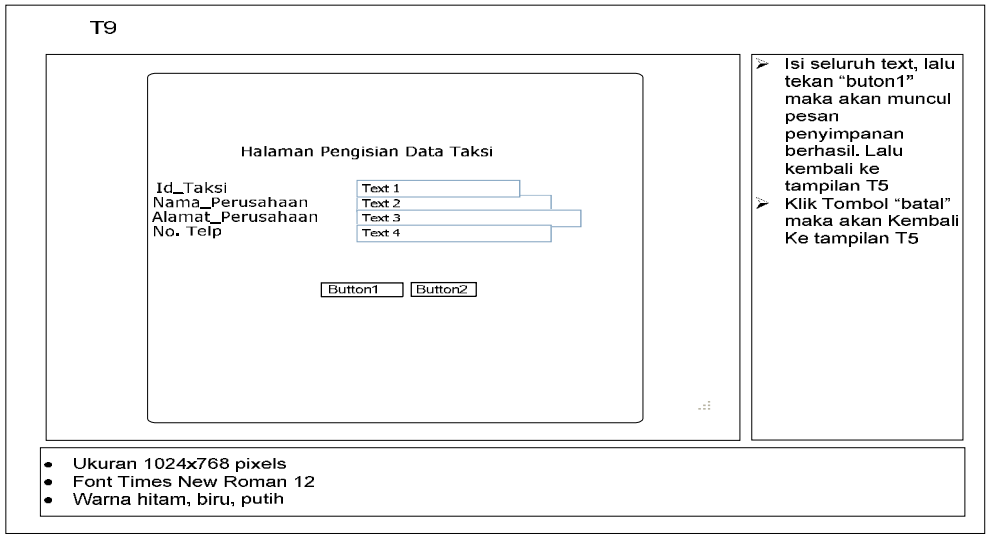
Gambar 3.63 Perancangan Antar Muka Halaman Kelola Data Patas

3.3.4.1.17. Perancangan Antar Muka Halaman Data Taksi



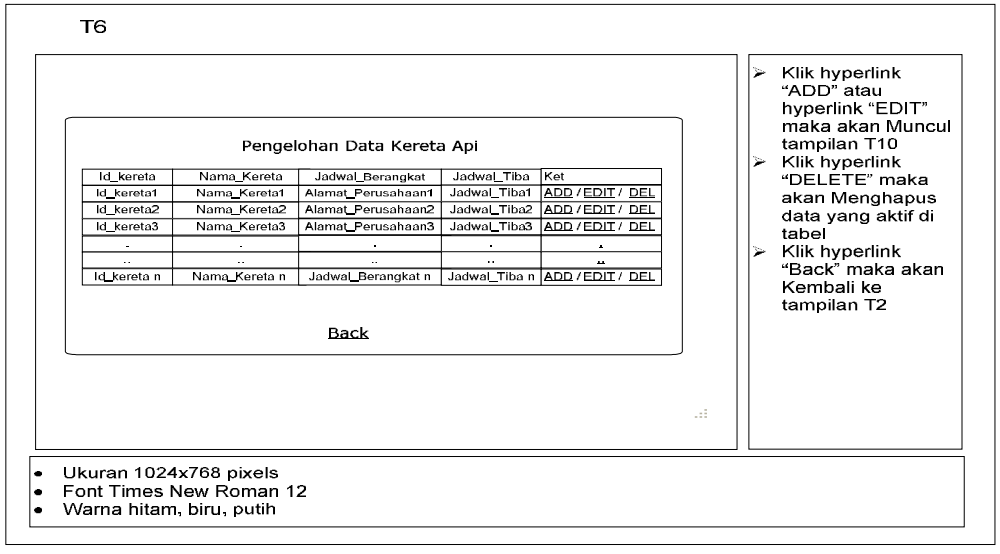
Gambar 3.64 Perancangan Antar Muka Halaman Data Taksi

3.3.4.1.18. Perancangan Antar Muka Halaman Kelola Data Taksi



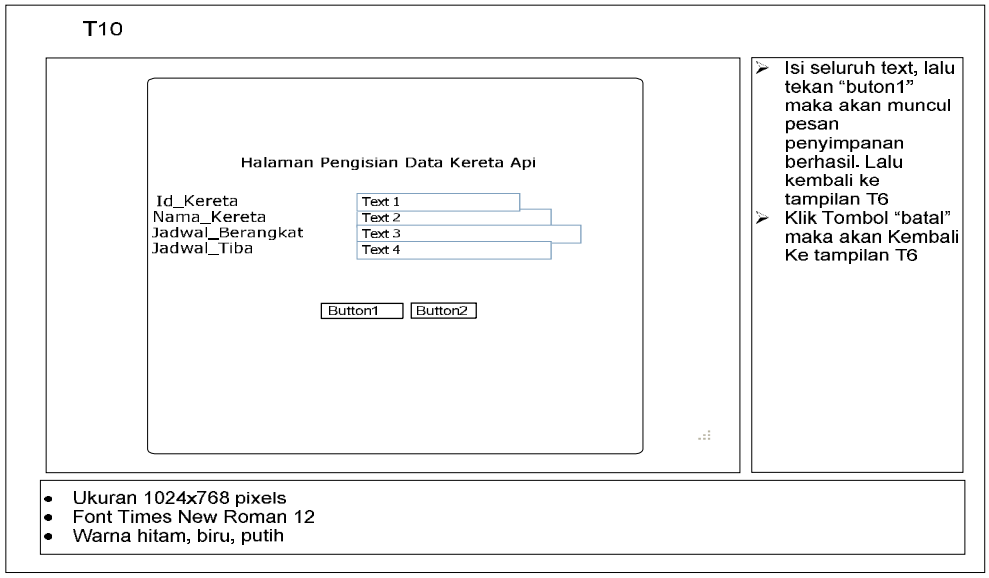
Gambar 3.65 Perancangan Antar Muka Halaman Kelola Data Taksi

3.3.4.1.19. Perancangan Antar Muka Halaman Data Kereta Api



Gambar 3.66 Perancangan Antar Muka Halaman Data Kereta Api

3.3.4.1.20. Perancangan Antar Muka Halaman Kelola Data Kereta Api



Gambar 3.67 Perancangan Antar Muka Halaman K elola Data Kereta Api