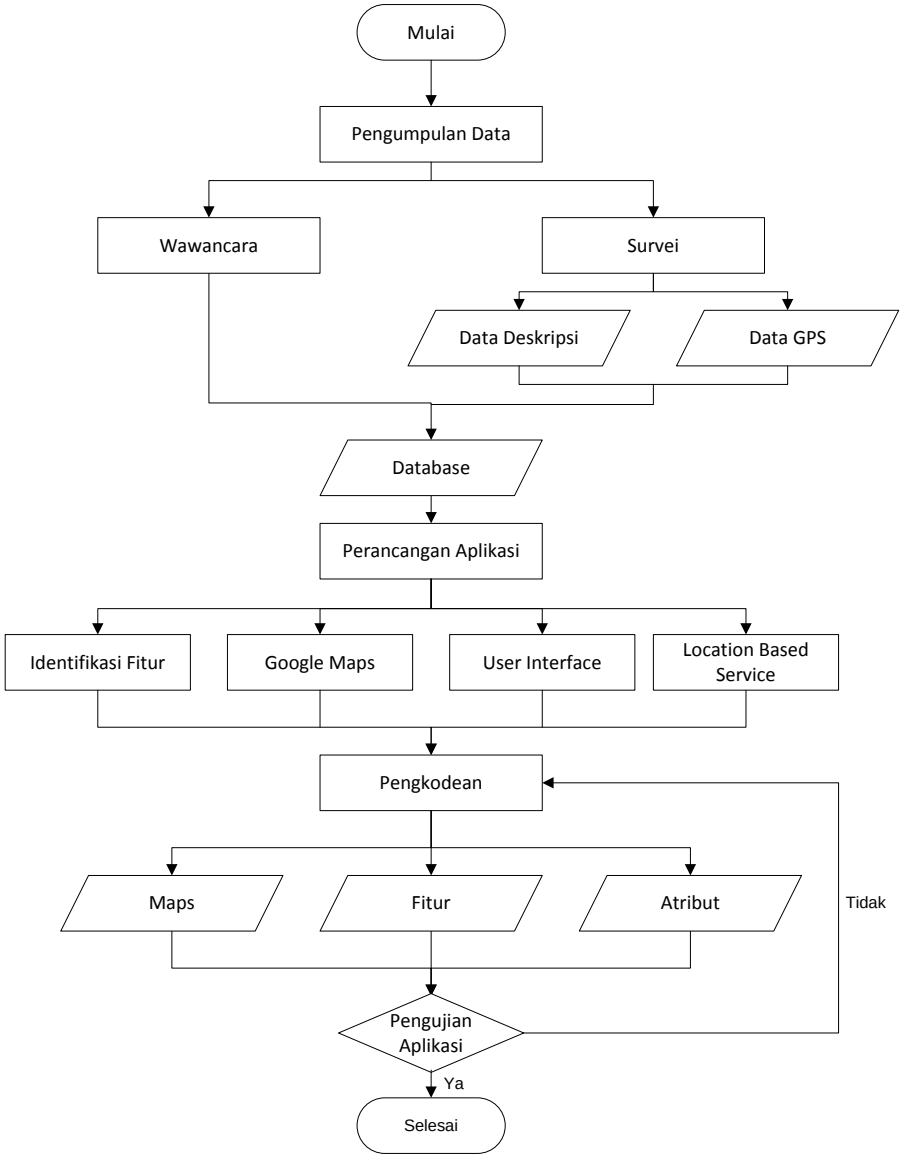
BAB III
METODOLOGI PENELITIAN

Pada bab ini akan dijelaskan mengenai metodologi yang digunakan dalam pelaksanaan penelitian ini, dimana hasil akhirnya adalah berupa aplikasi persebaran objek wisata di Kota Semarang yang bernama GeoTourism pada *smartphone* Android. Berikut adalah diagram alir dari penelitian ini.



Gambar 3.1 Diagram Alir

III.1 Alat dan Bahan

III.1.1 Peralatan

Peralatan yang digunakan dalam penelitian ini di spesifikasikan dalam *hardware* dan *software*, yaitu sebagai berikut :

1. Perangkat keras atau *hardware* yang terdiri dari :
   a. Perangkat laptop dengan spesifikasi *Processor Intel* (R) *Core* (TM) i3-2310M CPU @ 210 Ghz (4 CPUs), *Harddisk* 350 GB, RAM 2.00 GB.
   b. GPS *Handheld/Mobile*
   c. Kamera
   d. *Smartphone* Android (Lenovo A800 *Ice Cream Sandwich 4.0.4 version*)
2. Perangkat lunak atau *software*, yang terdiri dari :
   a. *App Inventor* (*version* v.134), digunakan sebagai media pembuatan aplikasi.
   b. *Java Development Kit* (JDK), agar komputer dapat membaca bahasa pemrograman Java.
   c. *Google App Engine*, digunakan untuk membuka *App Inventor* pada *browser.*
   d. *Python* 2.7, digunakan untuk membaca bahasa pemrograman *visual block.*
   e. *Google Chrome*, digunakan sebagai media aplikasi *App Inventor* secara *online* maupun *offline.*
   f. *Microsoft Office Visio 2007*, digunakan untuk perancangan sistem dan atau diagram aplikasi.
   g. *Photoshop CS5*, digunakan untuk pembuatan tampilan *user interface* dan atau *editing* atribut foto.
   h. *VideoPad Professional*, digunakan untuk *editing* data atribut video.

III.1.2 Data Penelitian

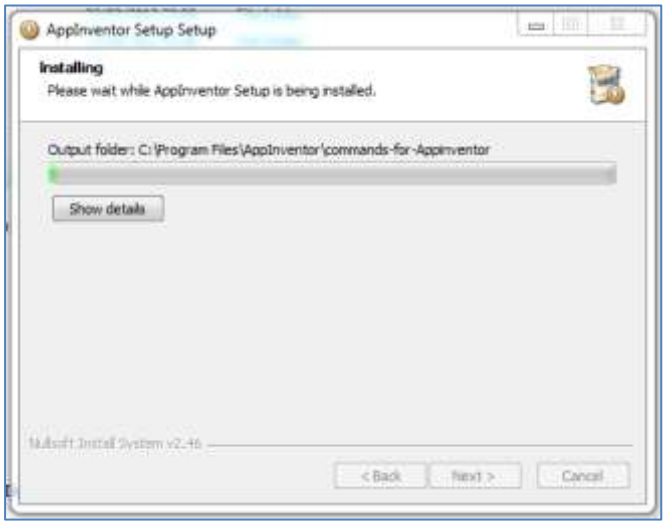
Data-data yang diperlukan dalam pelaksanaan penelitian ini adalah sebagai berikut :

1. Data koordinat GPS tiap lokasi objek wisata, diperoleh melalui pengukuran GPS.
2. Data daftar objek wisata unggulan Kota Semarang, diperoleh dari Dinas Kebudayaan dan Pariwisata Kota Semarang.
3. Data atribut, diperoleh dari Dinas Kebudayaan dan Pariwisata Kota Semarang serta survey lapangan.

III.2 Instalasi Program

Langkah awal dalam pelaksanaan penelitian ini adalah dengan mempersiapkan *software-software* yang dibutuhkan, baik itu *software* utama maupun *software* pendukung, seperti yang telah disebutkan diatas antara lain :

1. Instalasi *Java Development Kit* (JDK). Android adalah aplikasi *open source* yang dikembangkan dengan basis bahasa pemrograman Java, sehingga agar dapat membaca bahasa pemrograman yang akan dibuat, maka pada komputer harus terinstal program Java.
2. Instalasi *App Inventor* (*version* v.134). *App Inventor* akan digunakan dalam pembuatan aplikasi dimana nantinya dapat di jalankan secara *online* melalui *browser* atau dijalankan secara *offline* dengan *buildserver localhost* melalui *browser*. App Inventor merupakan aplikasi yang bersifat *free* dan terintegrasi dengan Google sehingga dapat digunakan oleh setiap *developer* dengan cara mendaftar menggunakan akun *Gmail*. *Project* yang dibuat akan tersimpan secara otomatis jika menggunakan *App Inventor* secara *online* melalui *browser*.



Gambar 3.2 Instalasi *App Inventor*

3. Instalasi *Python*. *Software* ini digunakan untuk pembacaan bahasa pemrograman yang digunakan saat pembuatan aplikasi, dalam hal ini adalah bahasa pemrograman *visual block* dan juga dapat digunakan untuk pembuatan *tinywebdb*. Karena *Python* dapat membaca multi paradigma pemrograman interpretatif multiguna dimana tidak dibatasi pada satu jenis bahasa pemrograman.

Gambar 3.3 Instalasi *Software Python*

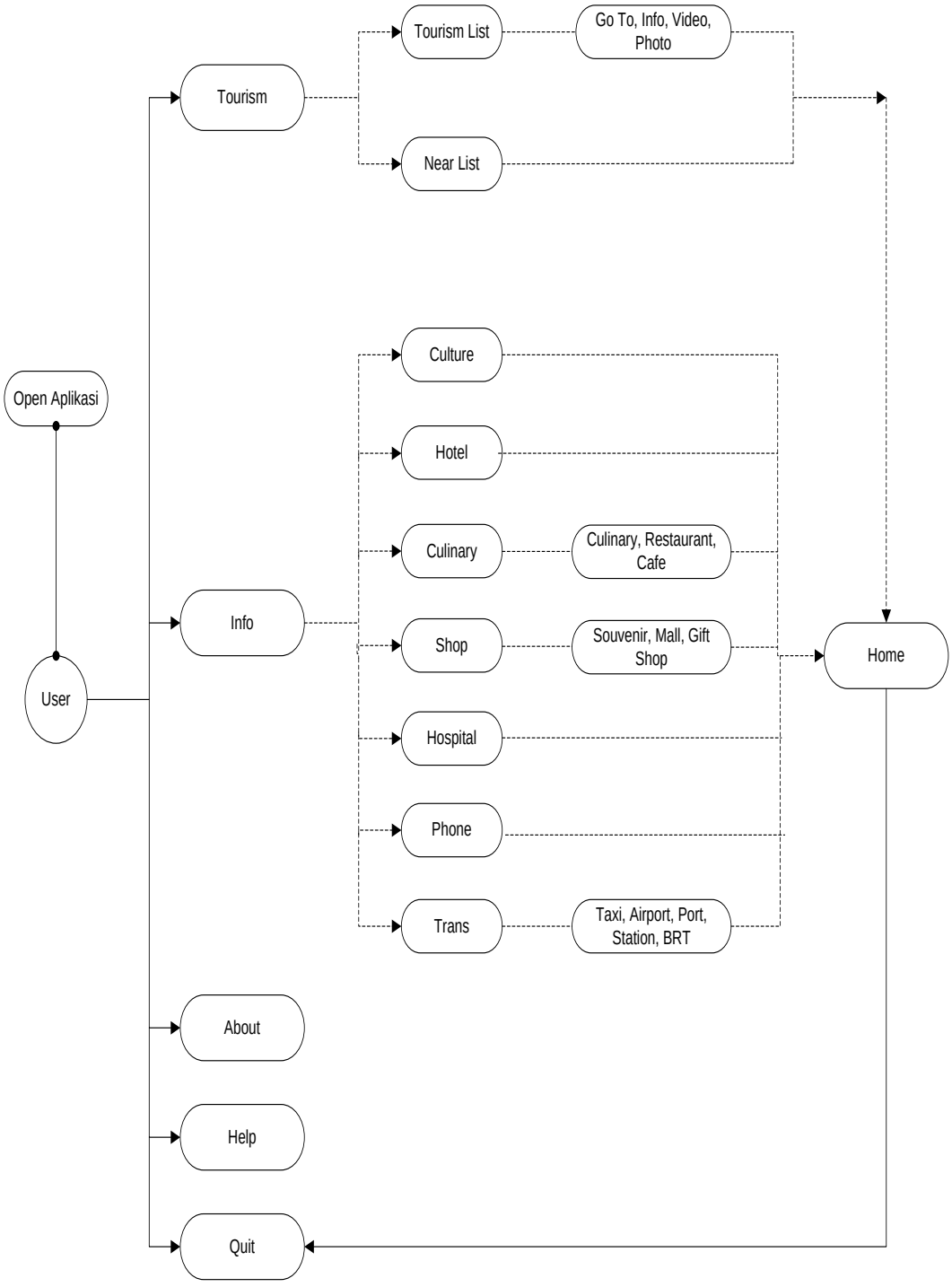
4. Instalasi *Google App Engine*. Digunakan sebagai pendukung saat menjalankan *App Inventor* secara *online* atau *offline*. Dimana kita dapat mengupload paket *project* yang ada ke *App Inventor* kemudian melakukan proses pembuatan aplikasi.

III.3 Perancangan Program

Pada tahap ini adalah melakukan perancangan program dimana nantinya akan menentukan hasil akhir dari aplikasi yang telah dibuat, yaitu meliputi perancangan sistem, *user interface,* dan *activity* yang akan dijalankan oleh *user.*

III.3.1 Perancangan Sistem Aplikasi

Pada perancangan sistem aplikasi yaitu melakukan perencanaan mengenai sistem aktivitas yang ada pada aplikasi dan akan dibuat menggunakan metode UML (*Unified Modelling language*). UML adalah bahasa standar yang digunakan untuk menjelaskan dan menvisualisasikan artifak dari proses analisis dan desain berorientasi objek. UML memungkinan *developer* melakukan pemodelan secara visual, yaitu penekanan pada penggambaran. Pemodelan visual membantu untuk menangkap struktur dan perilaku dari objek, mempermudah penggambaran interaksi antara elemen dalam sistem, dan mempertahankan konsistensi antara desain dan implementasi dalam pemrograman. Berikut adalah gambaran perancangan sistem pada aplikasi GeoTourism dengan metode UML.



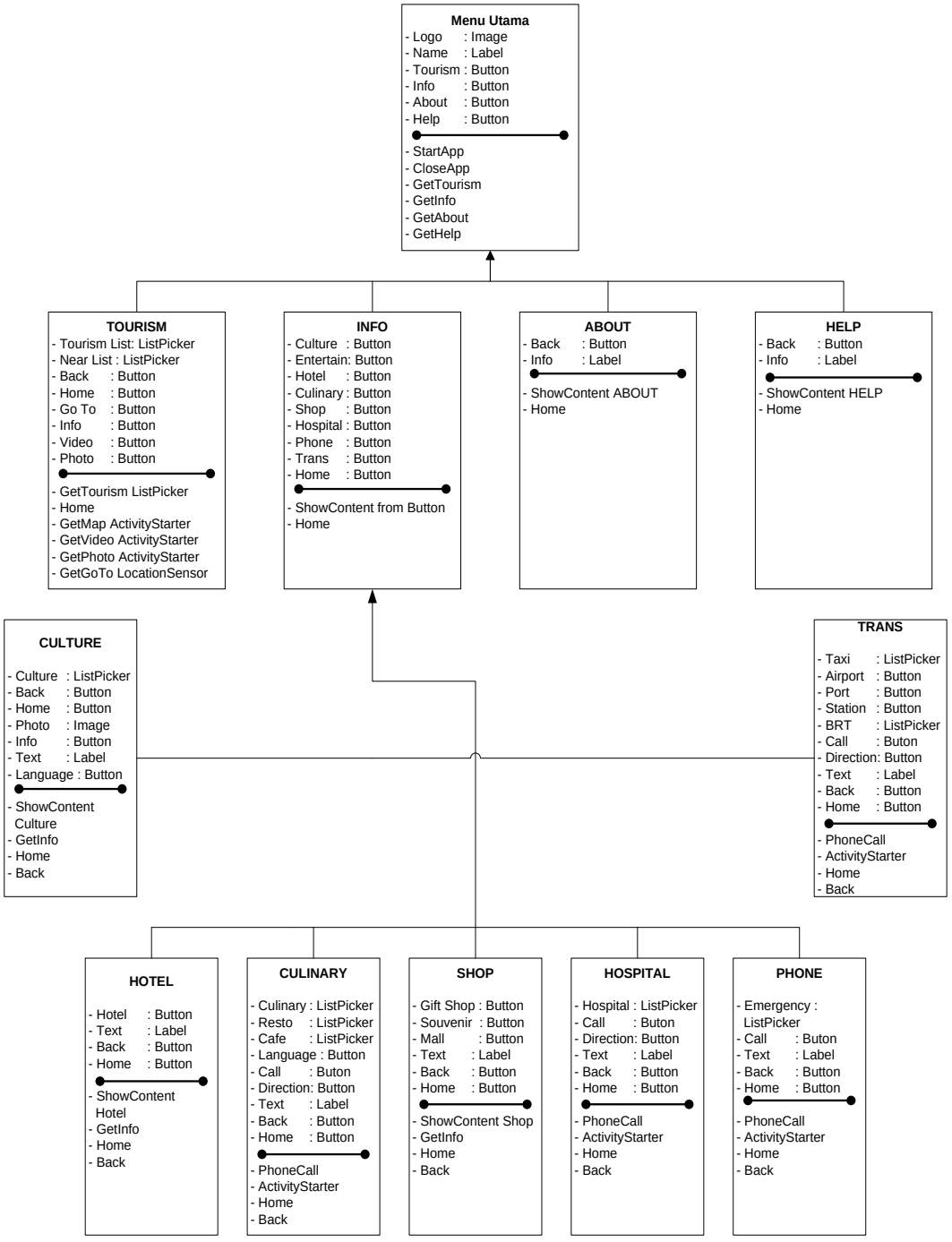
Gambar 3.4 Diagram *Use Case* Aplikasi

Use case diagram pada GeoTourism memaparkan tentang penggunaan aplikasi secara optional oleh *user*. *User* adalah pengguna aplikasi yang dapat

dengan bebas menggunakan aplikasi GeoTourism. Ada beberapa aktifitas yang dapat dilakukan *user* di aplikasi ini yaitu :

a. Pada halaman awal aplikasi GeoTourism, sistem akan me-*load* data-data yang dibutuhkan untuk menjalankan aplikasi seperti data gambar dan fungsi-fungsi algoritma dan lain lain. Di halaman awal aplikasi terdiri dari *button Tourism, Info, About*, dan *Help.*
b. Jika *user* memilih menu *Tourism List* maka akan muncul *ListPicker* pilihan objek wisata, kemudian didalamnya terdapat beberapa *button*, antara lain *Go To, Info, Video, Photo*, dimana *case* ini merupakan bagian dari konten *Tourism List.*
c. Jika *user* memilih *button Info* maka akan muncul case yang berbeda dari Tourism dengan beberapa pilihan informasi antara lain *button Culture, Hotel, Culinary, Shop, Hospital, Phone (Emergency),* dan *Trans (Transportation).* Pada *button-button* pilihan diatas masih memiliki berbagai macam *sub-button.*
d. Jika *user* memilih *button Help* maka akan muncul keterangan dari setiap pilihan *button* baik dari *button Tourism* maupun *button-button* pilihan pada *case Info*.
e. Jika *user* memilih *button About* maka akan muncul riwayat atau keterangan dari aplikasi GeoTourism.
f. Jika *user* memilih *button Quit* maka sistem akan menutup aplikasi.

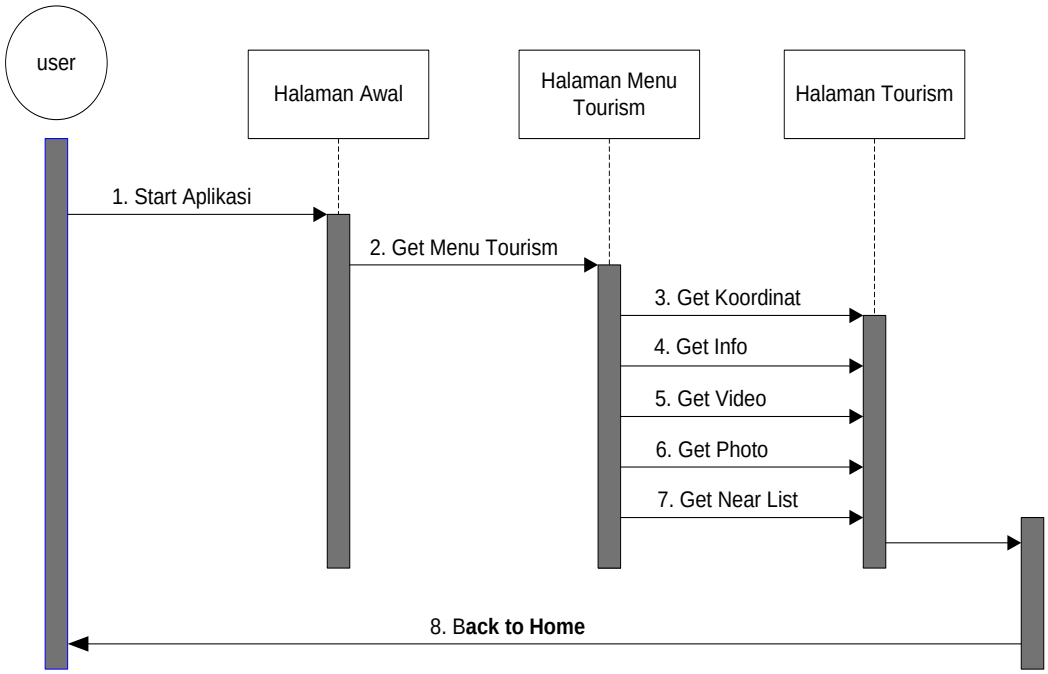
Class diagram merupakan diagram yang digunakan untuk menampilkan beberapa kelas serta paket-paket yang ada dalam sistem atau perangkat aplikasi yang akan dibuat. *Class* diagram memberikan gambaran statis tentang sistem atau komponen perangkat lunak yang kompleks. Aplikasi ini mempunyai *class* diagram dari sisi *user*. Berikut ini adalah gambar rancangan objek berupa *class* diagram tersebut.



Gambar 3.5 Diagram *Class* Aplikasi

Sequence diagram menjelaskan secara detail urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case*. Interaksi yang terjadi antar *class*, operasi apa saja yang akan terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi.

Sequence diagram pada aplikasi GeoTourism, dimulai ketika menjalankan aplikasi kemudian masuk kedalam halaman *Home* dengan berbagai macam pilihan *button*. Untuk melihat spesifikasi dari sebuah pilihan *button* yang ada maka *user* harus memilih sebuah *button* kemudian akan ditampilkan *sub*-menu dari masing-masing *button* tersebut. Dari *sub*-menu tersebut akan ditampilkan informasi yang sesuai dengan pilihan *user*. Selanjutnya adalah melakukan beberapa rangkaian *sequence* diagram dari *button Tourism.* Berikut adalah salah satu sequence diagram pada sistem *button Tourism.*



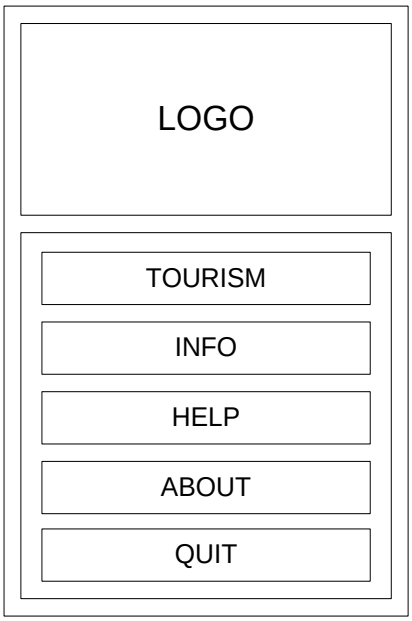
Gambar 3.6 Diagram *sequence* dari *button Tourism*

III.4 Rancangan *User Interface*

Rancangan *user interface* atau tampilan aplikasi yang akan dibuat harus dapat memberikan gambaran dan penjelasan dari setiap gambar, teks dan navigasi dari aplikasi. Rancangan tampilan ini menggambarkan keterkaitan setiap halaman dan juga menjelaskan arah komunikasi pada aplikasi. Rancangan tampilan ini bertujuan agar aplikasi yang dihasilkan akan terlihat lebih menarik, mudah dipahami dan dioperasikan.

Rancangan *user interface* pada aplikasi GeoTourism ini memiliki tampilan yang bervariasi, dikarenakan kebutuhan dari setiap jenis data yang ada pada masing–masing *sub-button*. Berikut adalah beberapa rancangan tampilan dari aplikasi GeoTourism yang lebih difokuskan pada tampilan yang berkaitan dengan objek wisata atau pada *button Tourism.*

Pertama, pada rancangan halaman awal atau *Home* aplikasi GeoTourism ini terdapat beberapa gambaran mengenai letak logo aplikasi dan beberapa *button* penghubung ke halaman-halaman berikutnya seperti *button Tourism, Info, Help, About,* dan *Quit* (*button* untuk keluar dari aplikasi).



Gambar 3.7 Rancangan tampilan *screen Home* aplikasi

Setelah *user* dihadapkan pada tampilan *screen Home* aplikasi GeoTourism, *user* dapat memilih *button Tourism*. Pada halaman ini terdapat dua pilihan *button*, yaitu *button Tourism List* yang akan menampilkan daftar objek wisata yang tersebar di Kota Semarang dan *button Near List* yang akan menampilkan daftar objek wisata di Kota Semarang berdasarkan jarak terdekat dari posisi *user*. Kedua *button* ini menggunakan fungsi *ListPicker* dan memiliki *database* yang sama dan saling berhubungan satu sama lain dengan cara kerja

yang berbeda. Berikut adalah rancangan tampilan yang akan muncul ketika *user* memilih *button Tourism* pada *screen Home* aplikasi GeoTourism.

Gambar 3.8 Rancangan tampilan *screen Tourism* aplikasi

Pada halaman ini *user* dapat memilih salah satu diantara dua buah *button*, yaitu *button Tourism List* dan *Near List*. Fungsi dari kedua *button* ini memiliki perbedaan prosedur. Ketika *user* memilih *button Tourism List*, maka *user* akan dihadapkan pada beberapa pilihan daftar objek wisata unggulan yang ada di Kota Semarang dan *user* dapat memilih objek wisata tujuan. Setelah *user* memilih objek wisata tujuan, pada layar aplikasi GeoTourism akan menampilkan Informasi mengenai objek wisata yang telah dipilih. Pada halaman ini akan menampilkan foto objek wisata dan juga beberapa *buttton* mengenai informasi dari objek wisata. Diantaranya terdapat beberapa *button* yang berisi informasi sebagai berikut :

- *Button Go To*

  Berfungsi menampilkan posisi *user* berdiri berdasarkan koordinat GPS dari perangkat *smartphone* Android. Kemudian juga menampilkan *button Direction* yang berfungsi menghubungkan posisi *user* ke aplikasi *Google Maps* sebagai panduan petunjuk arah ke lokasi objek wisata tujuan.

- *Button Info*

  Berfungsi menampilkan informasi terkait objek wisata tujuan, antara lain alamat, nomor telepon, dan juga review objek wisata yang dapat ditampilkan dalam dua bahasa, yaitu Bahasa Indonesia dan Bahasa Inggris.
- *Button Video*

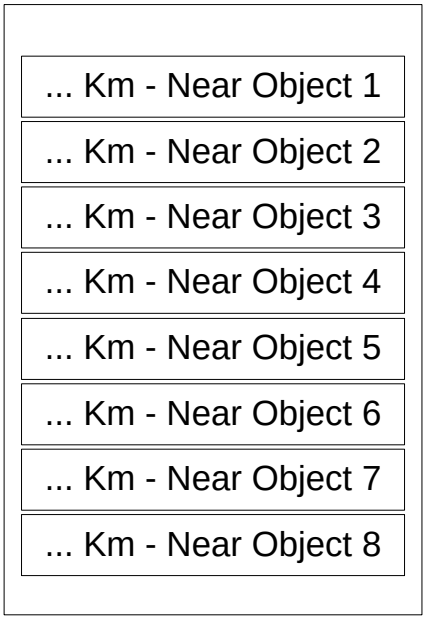
  Berfungsi menampilkan video *preview* dari objek wisata tujuan. Dimana nantinya video tersebut akan langsung terhubung dan ditampilkan melalui aplikasi video *Youtube.*
- *Button Photo*

  Berfungsi menampilkan beberapa foto dari setiap objek wisata tujuan yang akan langsung ditampilkan pada halaman aplikasi GeoTourism.

Gambar 3.9 Rancangan tampilan *screen Tourism List*

Sedangkan pada *button Near List* akan menampilkan daftar objek wisata terdekat dalam radius 5 kilometer dari posisi *user*. Daftar objek wisata terdekat ini didapatkan dari perhitungan beberapa formula dan algoritma dari sistem aplikasi. Selanjutnya setelah memilih salah satu objek wisata dari daftar objek wisata terdekat, maka akan terhubung pada database aplikasi dan ditampilkan dalam

bentuk tampilan yang sama seperti pada Gambar 3.8. Berikut ini adalah rancangan tampilan *list* objek wisata terdekat pada *Near List*.



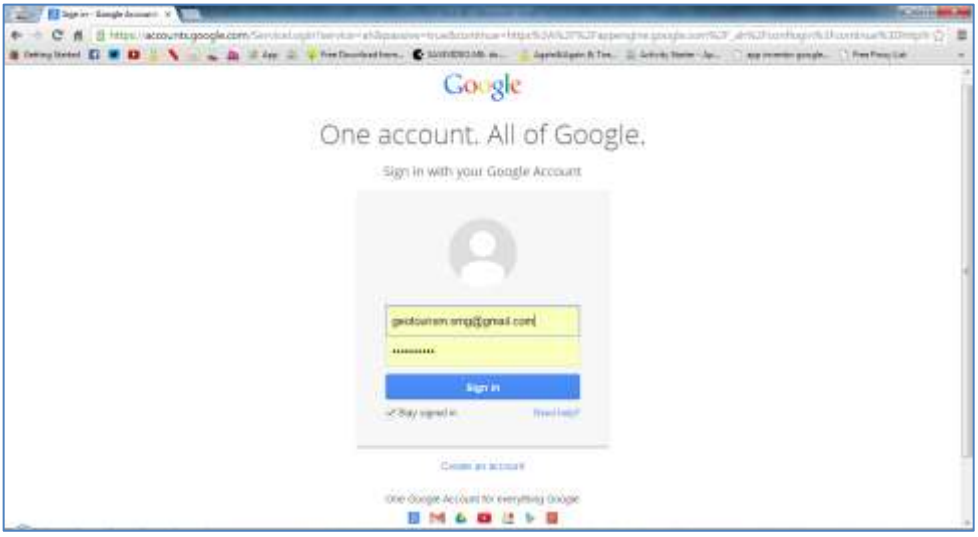
Gambar 3.10 Rancangan tampilan daftar *Near List*

III.5 Pengkodean Aplikasi

Dalam membangun sebuah aplikasi tentunya dibutuhkan pengkodean sistem. Pengkodean pada pembuatan aplikasi dengan *App Inventor* menggunakan koding-koding yang berbentuk *visual block* yang saling berhubungan agar nantinya aplikasi bisa berjalan atau berfungsi sesuai dengan rancangan. *Block Editor App Inventor* menyediakan berbagai macam fungsi yang dapat digunakan sesuai dengan kebutuhan *developer*, misalnya untuk menyimpan berbagai macam data aplikasi, *developer* dapat menggunakan fungsi *variable* dengan struktur *list* yang bisa digunakan untuk menyimpan data-data, seperti data informasi, data gambar, data koordinat, dan data lainnya. Berikut ini adalah langkah pembuatan aplikasi GeoTourism yang difokuskan pada pengkodean sistem *button Tourism* dan beberapa sistem *block activity* pendukung lainnya.

III.5.1 Pembuatan *Project*

Langkah pertama dalam pembuatan aplikasi ini adalah dengan membuka *http://beta.appinventor.mit.edu* melalui *browser*, yang nantinya akan menampilkan halaman *sign in* akun *google mail (gmail)* terlebih dahulu.



Gambar 3.11 Tampilan halaman *sign in* akun *Gmail*

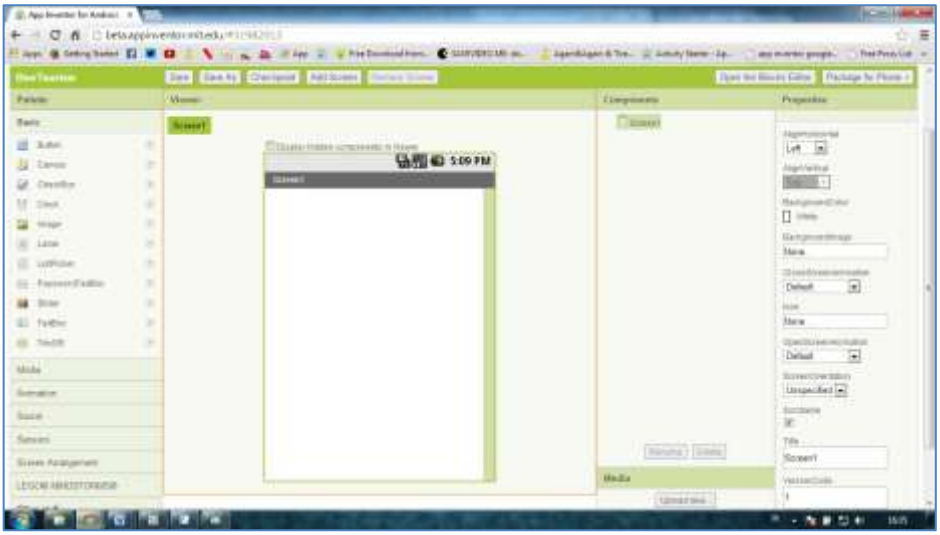
Masukkan alamat email akun *gmail* pada *textbox* yang disediakan, selanjutnya akan ditampilkan halaman konfirmasi dari *App Inventor* dan kemudian *browser* akan menampilkan halaman pembuatan *project App Inventor*.

Gambar 3.12 Halaman *project App Inventor*

Pada halaman *project App Inventor*, untuk memulai membuat aplikasi pilih *New*, kemudian masukkan nama *project* aplikasi yang ingin dibuat dan klik *OK*.

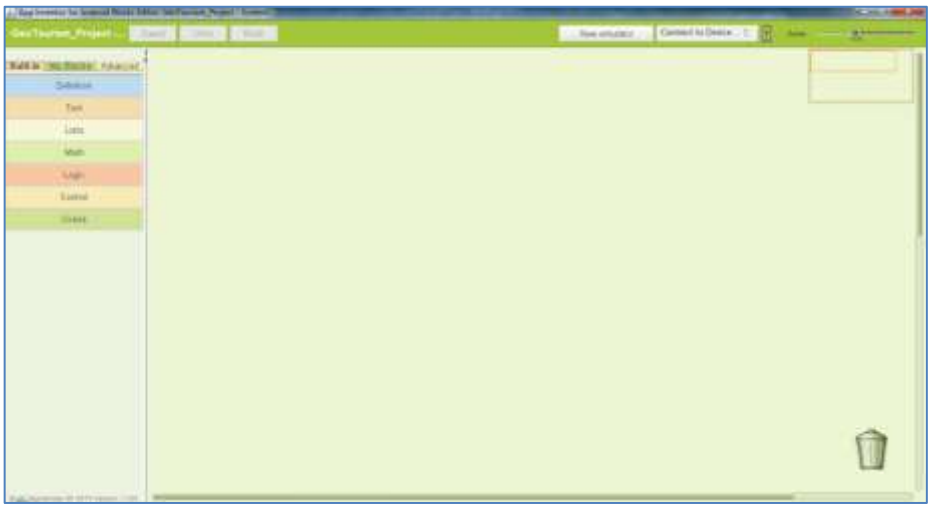
Gambar 3.13 Membuat *project* baru di *App Inventor*

Setelah membuat *project* baru, maka akan ditampilkan halaman *design* untuk *project* dari *App Inventor*. Pada halaman ini dapat dibuat rancangan *user interface* dengan melakukan *drag* a*nd drop* komponen dan menyesuaikan terhadap keperluan aplikasi.



Gambar 3.14 Halaman *design App Inventor*

Untuk memulai pengkodean pada *App Inventor*, klik *Open The Block Editor*. Kemudian pada halaman *block editor* buat rancangan sistem aplikasi dengan menggunakan bahasa pemrograman *visual block*.



Gambar 3.15 Halaman *Block Editor App Inventor*

III.5.2 Pengkodean

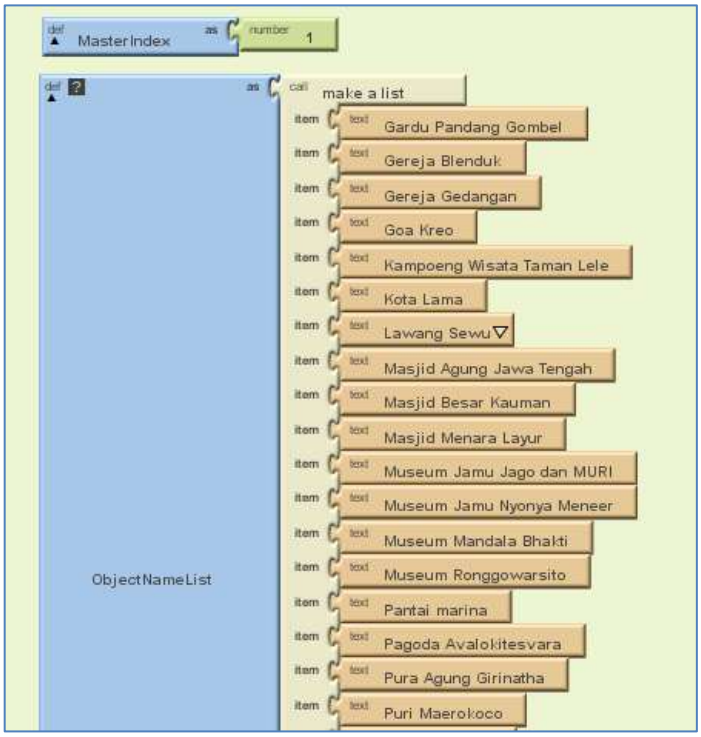
Bahasa pemrograman yang digunakan pada *App Inventor* adalah bahasa pemrograman *visual block*. Yaitu bahasa pemrograman yang dilakukan dengan cara pemasangan antar beberapa *block* untuk memproses suatu perintah dari sebuah aplikasi. *App Inventor* juga dapat melakukan penyimpanan data-data dari aplikasi pada sistemnya, sehingga *user* pun bisa langsung mendapatkan informasi yang diinginkan dari aplikasi tanpa memerlukan koneksi internet.

Ada beberapa macam pengkodean pada sistem aplikasi GeoTourism, seperti algoritma dalam pembacaan posisi, penghitungan jarak, pengambilan informasi, penyusunan daftar objek terdekat, dan beberapa macam pengkodean lainnya.

a. Pengkodean sistem *database button Tourism*

Pada *App Inventor*, *database* dapat dibuat dan disimpan didalam sistem aplikasi. Untuk penyimpanan data aplikasi dilakukan dengan cara membuat daftar dari data dalam bentuk variabel-variabel yang saling berhubungan melalui fungsi *index list*. *Index list* berfungsi sebagai penghubung dari satu *list database* ke *list database* lainnya, yaitu dengan cara memproses data yang

sesuai dengan urutan data yang ada pada *list database* induk. Pembuatan data pada aplikasi ini merupakan salah satu cara penyimpanan *database* pada aplikasi.

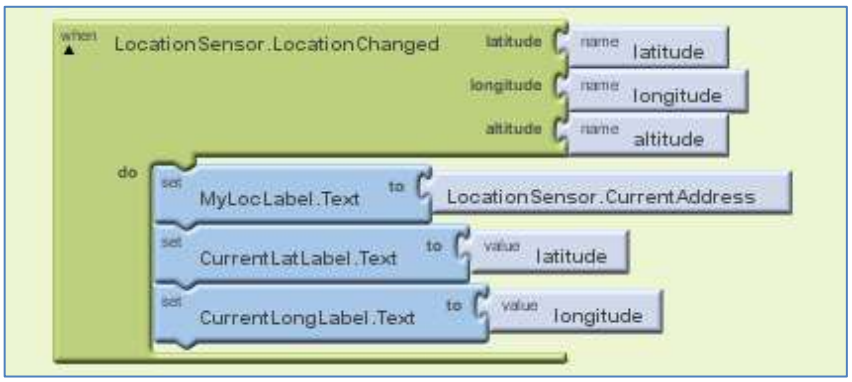


Gambar 3.16 Penyimpanan *database* pada aplikasi

Seperti terlihat pada gambar diatas, struktur penyimpanan data daftar objek wisata dibuat didalam suatu variabel yang bernama *ObjectNameList* kemudian blok variabel tersebut digabungkan dengan block fungsi *make a list* yang berfungsi menyimpan data nama objek wisata dalam bentuk *list*, dan data objek wisata yang dimasukan ke dalam *list* blok *ObjectNameList* ini nantinya akan ditampilkan dalam bentuk daftar pada komponen *ListPicker*. Data informasi setiap *item* akan diambil berdasarkan *index list* yang berhubungan dengan data aplikasi, contohnya ketika *user* memilih objek wisata Lawang Sewu maka sistem aplikasi akan merespon dengan menampilkan informasi-informasi mengenai objek wisata tersebut.

b. Pengkodean Sensor Lokasi

Pada aplikasi GeoTourism juga menggunakan sistem pengkodean untuk membaca posisi koordinat *user* berdasarkan data GPS yang diperoleh langsung dari *perangkat* GPS *smartphone* Android. Pembacaan posisi koordinat ini bertujuan untuk sistem navigasi pada aplikasi, dimana *user* akan diarahkan dari posisi *user* berdiri ke posisi koordinat dari setiap objek wisata yang sudah tersimpan didalam *database* aplikasi. Sistem navigasi pada aplikasi GeoTourism akan langsung terintegrasi ke aplikasi *Google Maps* pada perangkat *smartphone* Android. Dan pada aplikasi *Google Maps* nantinya *user* akan mendapatkan pengarahan langsung ke lokasi objek wisata tujuan dengan beberapa pilihan jalur yang direkomendasikan oleh *Google Maps*. Pada pembuatan algoritma pembacaan lokasi di *App Inventor* dapat menggunakan *palette sensor* yang ada pada halaman *designer App Inventor*, yaitu dengan men-*drag* komponen *LocationSensor* ke dalam *Screen Project*.

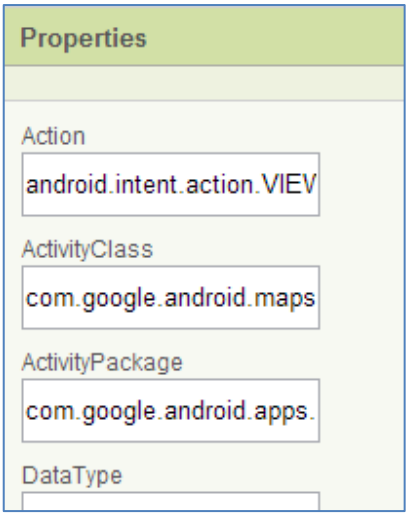


Gambar 3.17 *Block* pembacaan lokasi menggunakan *LocationSensor*

c. Pengaturan *Properties Maps ActivityStarter*

Untuk menghubungkan aplikasi GeoTourism ke Aplikasi *Google Maps*, dilakukan dengan cara menggunakan komponen *ActivityStarter* yang terdapat pada *palette other stuff* dari halaman *designer* dan men-*drag* ke *screen project*. Setelah itu melakukan pengaturan *properties ActivityStarter* pada halaman *designer* dengan mengisi kolom-kolom berikut :

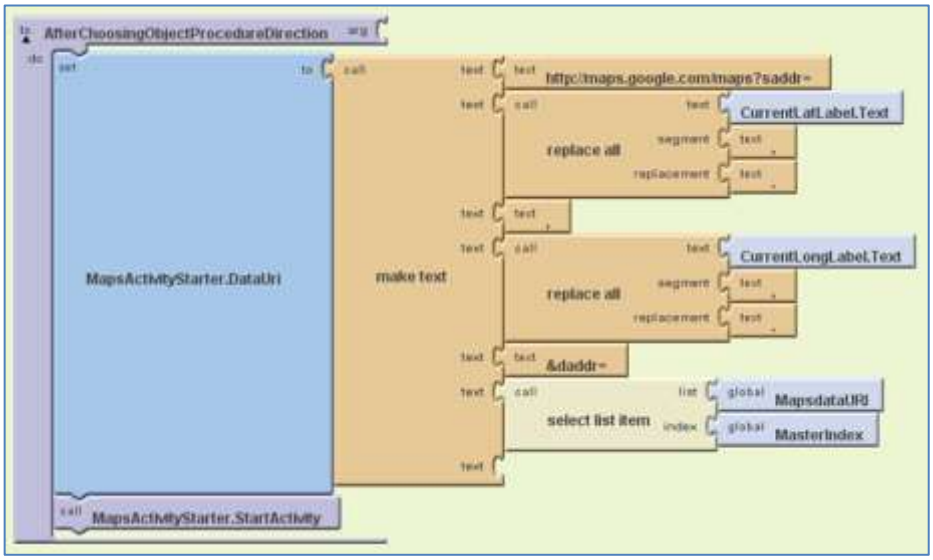
- *Action* : `android.intent.action.VIEW`
- *ActivityClass* : `com.google.android.maps.MapsActivity`
- *ActivityPackage* : `com.google.android.apps.maps`



Gambar 3.18 Pengaturan *Properties Maps ActivityStarter*

d. Pengkodean Integrasi *button Direction* dengan *Google Maps*

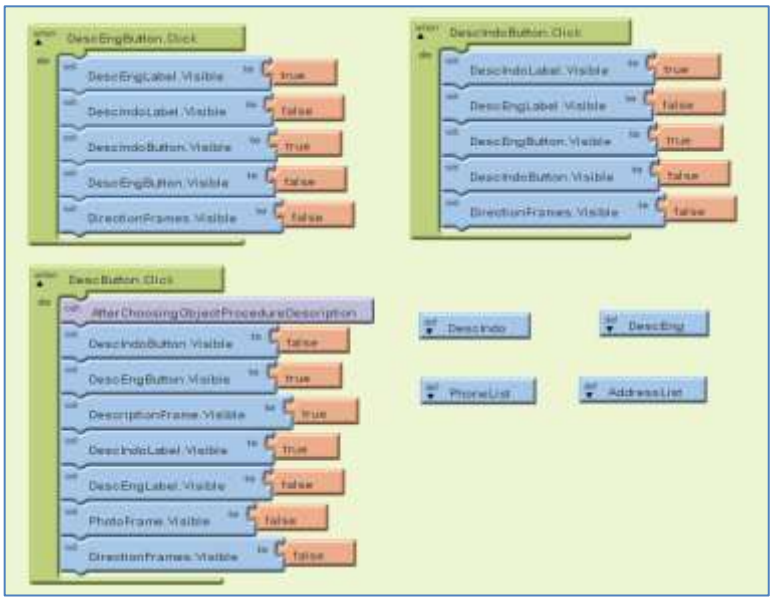
Pengkodean untuk menghubungkan aplikasi GeoTourism ke aplikasi *Google Maps* pada *block editor* dilakukan dengan cara menghubungkan variabel daftar nama objek wisata pada *ObjectNameList* ke variabel daftar koordinat lokasi objek wisata (dalam aplikasi ini diberi label *MapsDataURI*) berdasarkan *index list* melalui sistem block *button direction*. Pengkodean ini dimaksudkan sebagai navigasi ke objek wisata yang dipilih oleh *user*. Sehingga diharapkan dapat memberikan gambaran lokasi dari objek wisata.



Gambar 3.19 *Block* prosedur *Direction*

e. Pengkodean Pengambilan Informasi Teks

Selanjutnya adalah membuat sistem *block* untuk memperoleh informasi deskritif berupa teks dari sistem *database* aplikasi mengenai informasi pendukung dari setiap objek wisata, yang dalam hal ini dilakukan pada *button Info*. Sistem pengkodean masih dilakukan dengan cara menghubungkan variabel daftar objek wisata ke variabel daftar informasi masing-masing objek wisata berdasarkan *index list*.

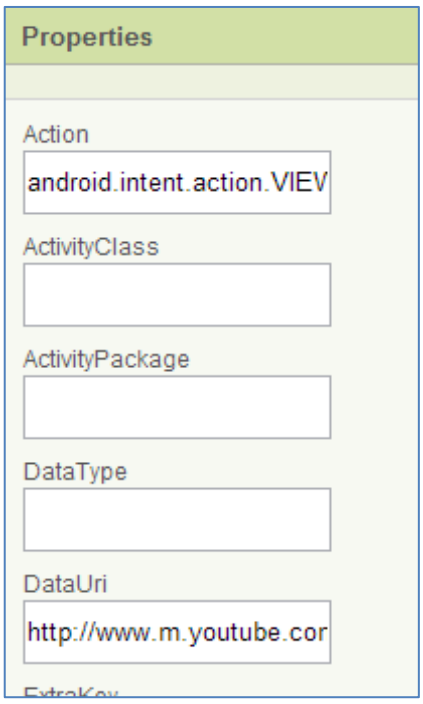


Gambar 3.20 *Block* dan *list database* pada *button Info*

f. Pengkodean *Video ActivityStarter*

Pada aplikasi GeoTourism juga memungkinkan *user* untuk melihat video *preview* dari masing-masing objek wisata. Untuk membuat sistem *block* pada video *ActivityStarter* dari *button Video*, yaitu dengan cara menghubungkan sistem aplikasi GeoTourism dengan aplikasi video *Youtube*. Pertama *upload* video dari masing-masing objek wisata ke *www.youtube.com* melalui *browser* untuk mendapatkan *link* video dari masing-masing video objek wisata. Selanjutnya untuk menghubungkan kedua aplikasi tersebut dilakukan dengan cara men-*drag* komponen *ActivityStarter* melalui *palette other stuff* pada halaman design *App Inventor* ke dalam *screen designer project* aplikasi. Lalu lakukan pengaturan pada *properties ActivityStarter* untuk menghubungkan aplikasi GeoTourism ke aplikasi video *Youtube* dengan mengisi kolom-kolom berikut :

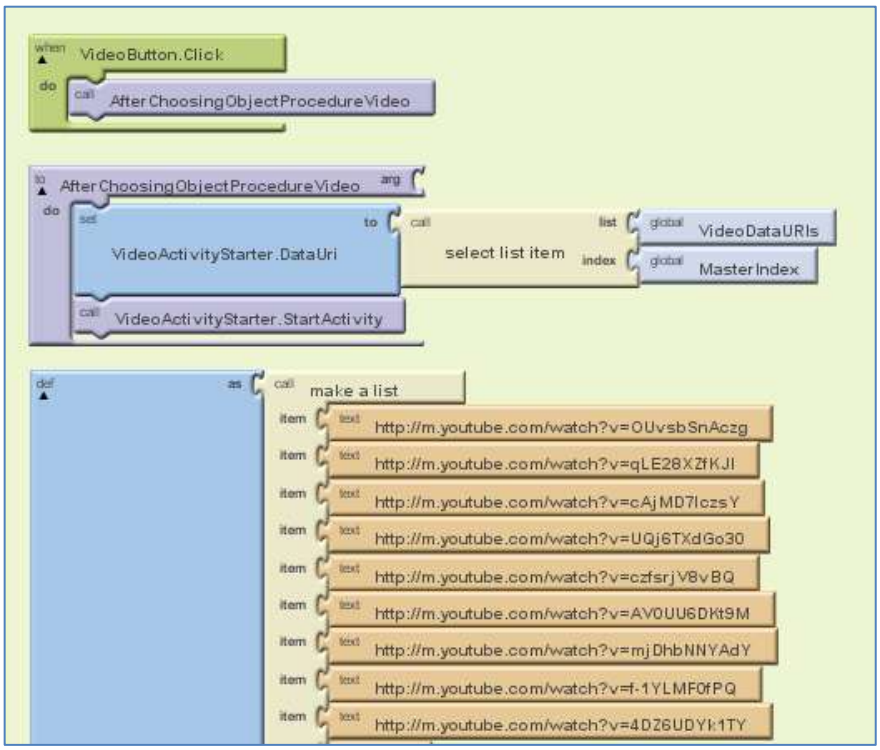
- *Action* : `android.intent.action.VIEW`
- *DataUri* : `http://www.m.youtube.com/watch?v=#######`



Gambar 3.21 Pengaturan *Properties Video ActivityStarter*

g. Pengkodean integrasi video ke *Youtube*

Pada sistem *block* untuk menghubungkan aplikasi GeoTourism dengan aplikasi video *Youtube*, lakukan pengkodean pada *block editor* dengan membuat sistem *block ActivityStarter* yang akan terintegrasi ke aplikasi Youtube. Setelah itu membuat variabel *database* video *URL*, yaitu sebuah variabel daftar *link* video dari masing-masing objek wisata. Dalam pengaplikasiannya, sistem *block* video nantinya akan menghubungkan sistem *ActivityStarter* video ke daftar *link* video, kemudian akan langsung memproses aplikasi GeoTourism untuk membuka aplikasi *Youtube* sesuai dengan video dari objek wisata yang dipilih oleh *user*.

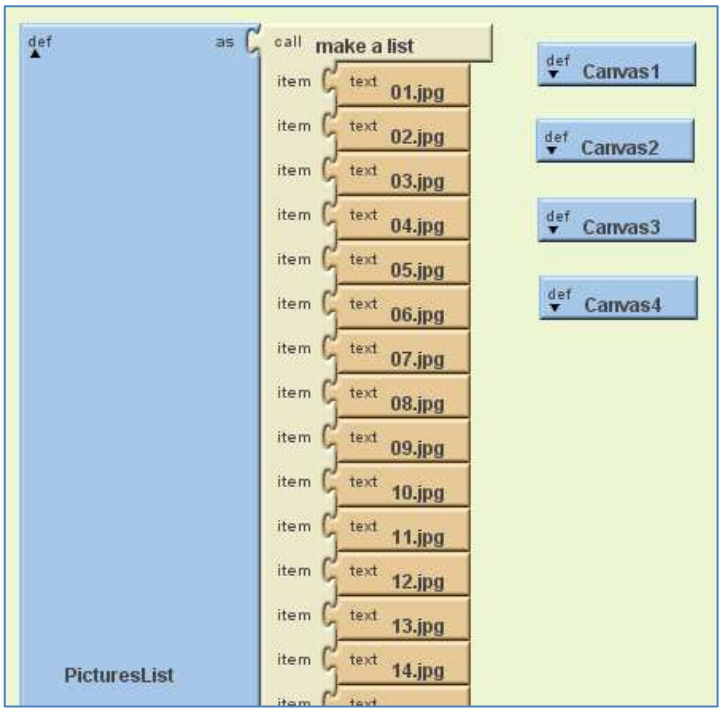


Gambar 3.22 *Block* prosedur dan *database link* video

h. Pengkodean *Database* Komponen *Image*

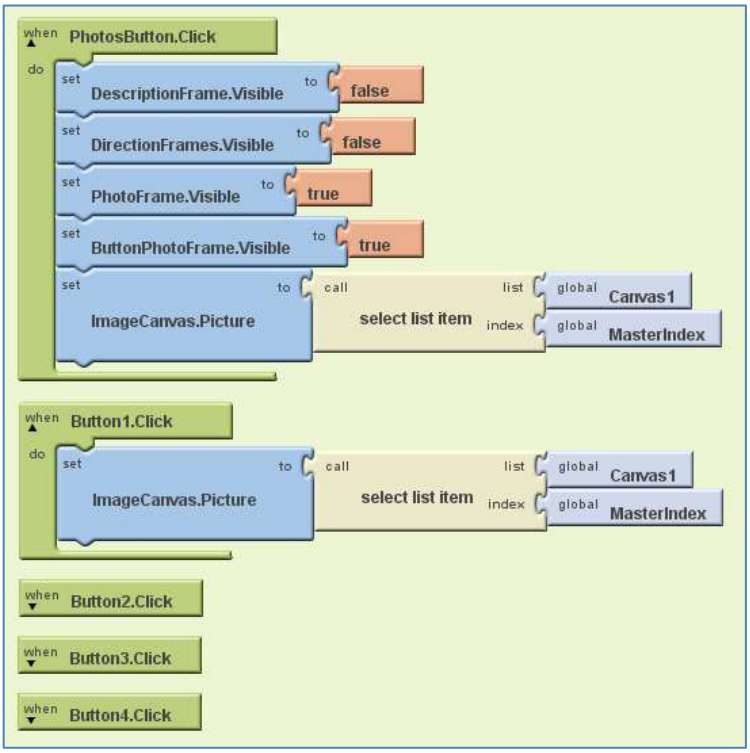
Untuk menampilkan foto pada aplikasi GeoTourism, pertama *upload* foto dari masing-masing objek wisata ke *Google DropBox Picasa* di *www.picasaweb.google.com* melalui *browser,* lalu simpan *link* dari setiap foto objek wisata. Selanjutnya pada *App Inventor*, *drag* komponen *image* dari

palette basic pada halaman *designer App Inventor* kedalam *screen project* aplikasi GeoTourism. Pada halaman *block editor*, buat *list database* baru menggunakan *block* variabel yang berisi *list* alamat *link* atau *URL* dari foto-foto yang telah di *upload. Database link* foto tersebut akan diproses berdasarkan *index list* masing-masing objek wisata. *Database* foto dari *button Photo* aplikasi GeoTourism berjumlah empat *list database*. Sedangkan Foto *Header* dari masing-masing objek hanya berjumlah satu *list database*.



Gambar 3.23 *Database link* foto

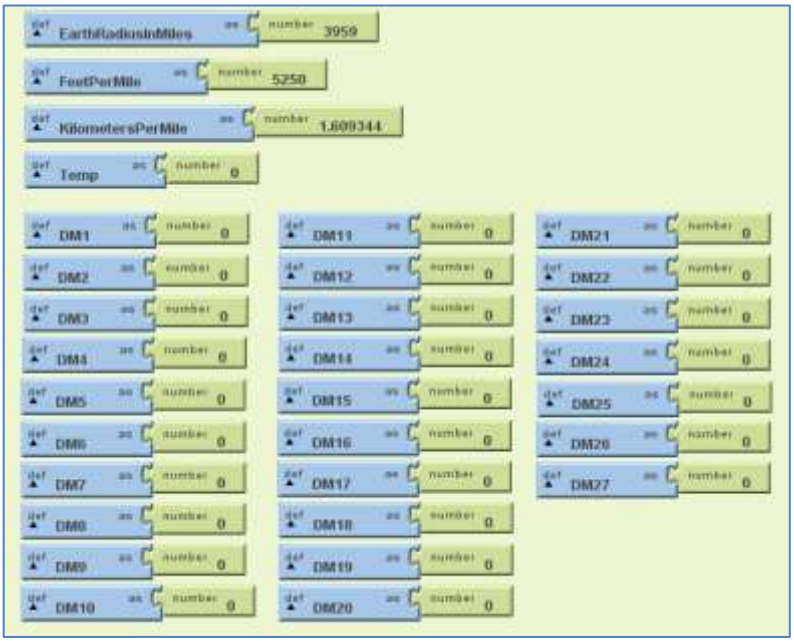
Setelah membuat *database list link* foto dari masing-masing objek wisata, langkah berikutnya yaitu membuat sistem *block* untuk menghubungkan aplikasi ke *database link-link* foto yang telah dibuat, sehingga nantinya akan diproses dan dapat langsung ditampilkan pada konten dari *button Photo.*



Gambar 3.24 *Block Image Activity*

i. Pengkodean *Index* pada *button Near List*

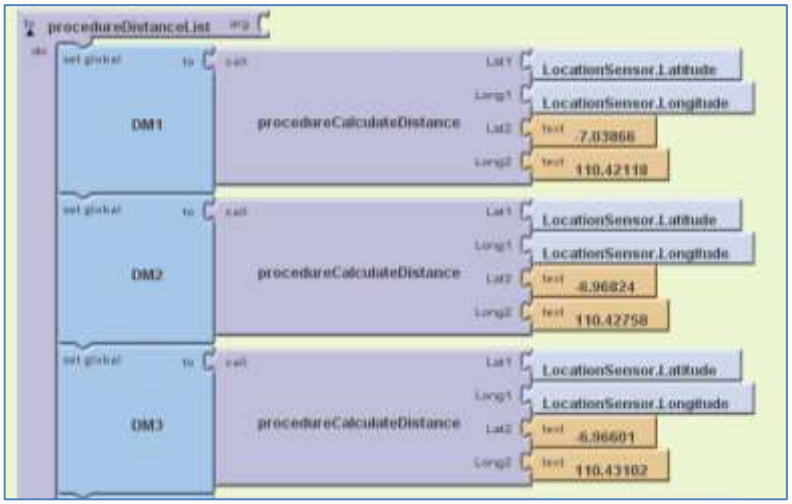
Selain pembuatan sistem pengkodean sebelumnya yang berkaitan dengan *button Tourism List*, selanjutnya adalah pembuatan sistem pengkodean pada *button Near List*. Pada sistem *block Near List* terdapat beberapa proses yang dilakukan aplikasi. Pertama aplikasi melakukan pembacaan koordinat lokasi *user* melalui sistem GPS *smartphone* Android seperti pada Gambar 3.16. Selanjutnya melakukan pengkodean *index* data yang akan digunakan untuk perhitungan jarak dari posisi *user* ke setiap koordinat lokasi yang terdapat pada *database*, serta memasukkan nilai dari setiap ketentuan jarak.



Gambar 3.25 *Block distance value* dan *index* data *Near List*

j. Pengkodean *block ProcedureDistanceList*

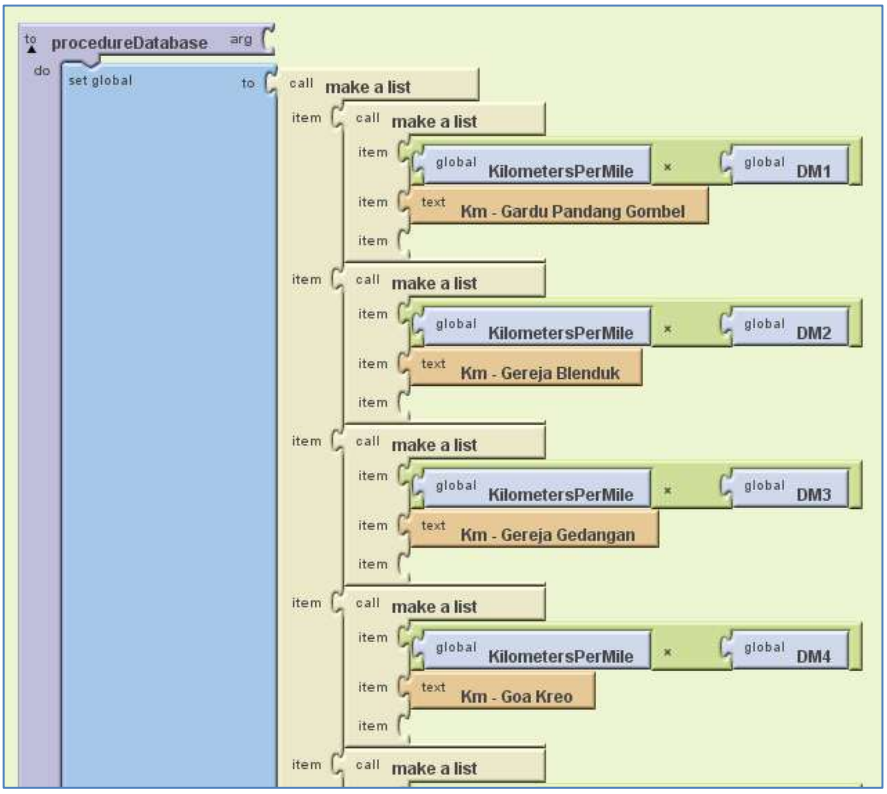
ProcedureDistanceList adalah pengkodean yang berisi *database* koordinat dari setiap objek wisata dan dipasangkan dengan *block LocationSensor* pembacaan posisi *user*. Pengkodean ini bertujuan untuk mengumpulkan perhitungan jarak dalam satu variabel *procedure* yang kemudian akan dihitung dengan rumus penghitungan jarak *haversine formula.*



Gambar 3.26 *Block ProcedureDistanceList*

k. Pengkodean *Database* pada *Button Near List*

Pengkodean ini bertujuan untuk menyusun hasil jarak dari setiap objek wisata yang diperoleh dari *procedureDistanceList* dan mengubahnya kedalam satuan kilometer. *List* jarak ini akan digunakan sebagai *database* daftar objek wisata terdekat pada *button Near List.*

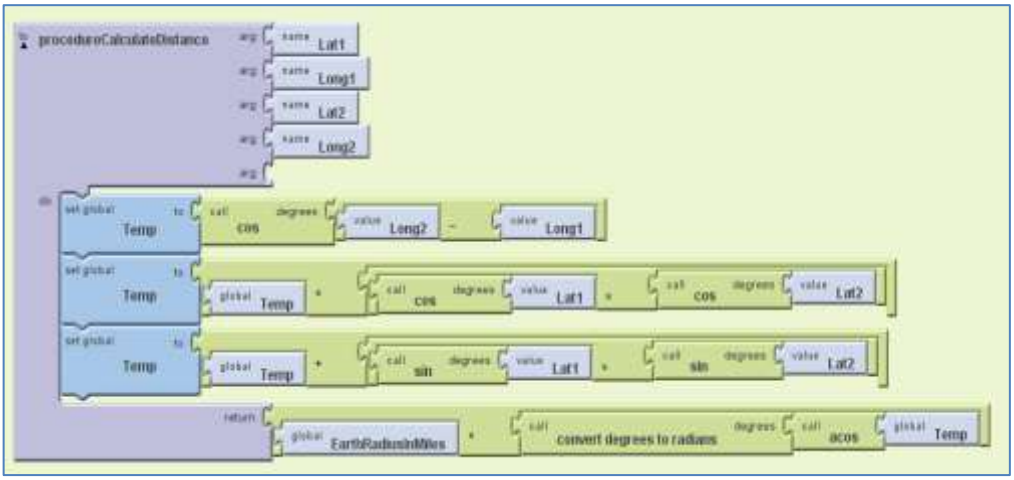


Gambar 3.27 *Block ProcedureDatabase Near List*

l. Pengkodean *Haversine Formula*

Penghitungan jarak pada aplikasi GeoTourism menggunakan *haversine formula* yang merupakan sebuah persamaan yang digunakan dalam navigasi, dimana formula ini memberikan jarak di antara dua titik pada lingkaran bola dari setiap garis bujur *(longitude)* dan garis lintang *(latitude)*. Sistem *block haversine formula* ini nantinya akan menghitung jarak dari koordinat posisi *user* ke semua koordinat lokasi objek wisata secara satu persatu dan menyimpannya secara temporal. Komponen *Lat1* didapatkan dari garis lintang posisi *user*, sedangkan komponen *Long1* didapatkan dari garis bujur

posisi *user*. Kedua komponen tersebut didapatkan melalui proses pembacaan posisi yang dilakukan *LocationSensor*. Kemudian komponen *Lat2* dan *Long2* didapatkan dari *database* koordinat objek wisata.

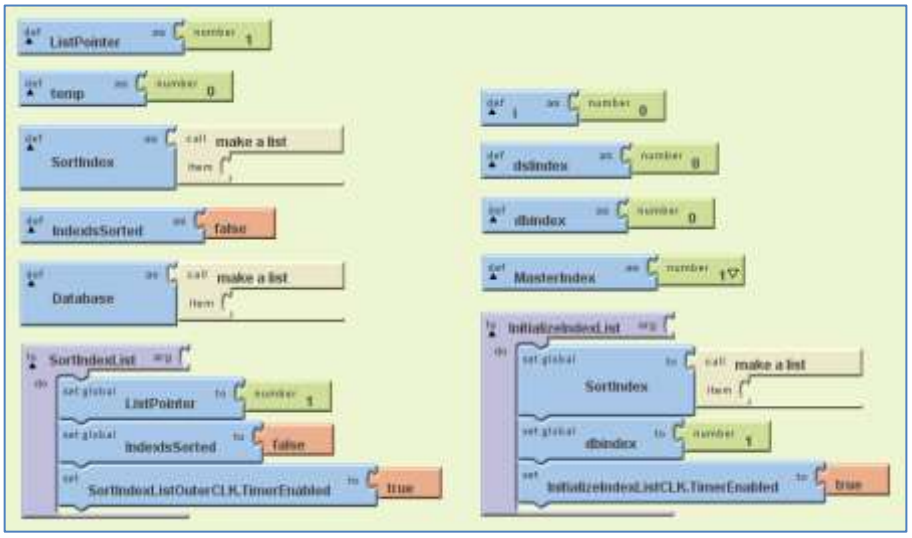


Gambar 3.28 *Block Haversine Formula*

m. Pengkodean *Index Database Bubble Sort Algorithm*

Setelah penghitungan jarak dari posisi *user* ke semua objek wisata didapatkan, sistem *block* akan melakukan langkah selanjutnya yaitu menyortir semua hasil jarak tersebut untuk menyusunnya kedalam daftar jarak terdekat hingga ke jarak terjauh dari posisi user berdiri. Penyusunan daftar *Near List* ini menggunakan *bubble sort algorithm*.

Langkah pertama sebelum membuat algoritma *bubble sort* adalah membuat *block Index* dari algoritma *bubble sort* terlebih dahulu. Komponen *index* dari algoritma *bubble sort* terdiri dari variabel *SortIndex* yang dibuat dalam bentuk *list*. *List database SortIndex* akan didapatkan dari hasil perhitungan jarak menggunakan *haversine formula* yang telah didapatkkan dari proses *ProcedureDatabase Near List.*

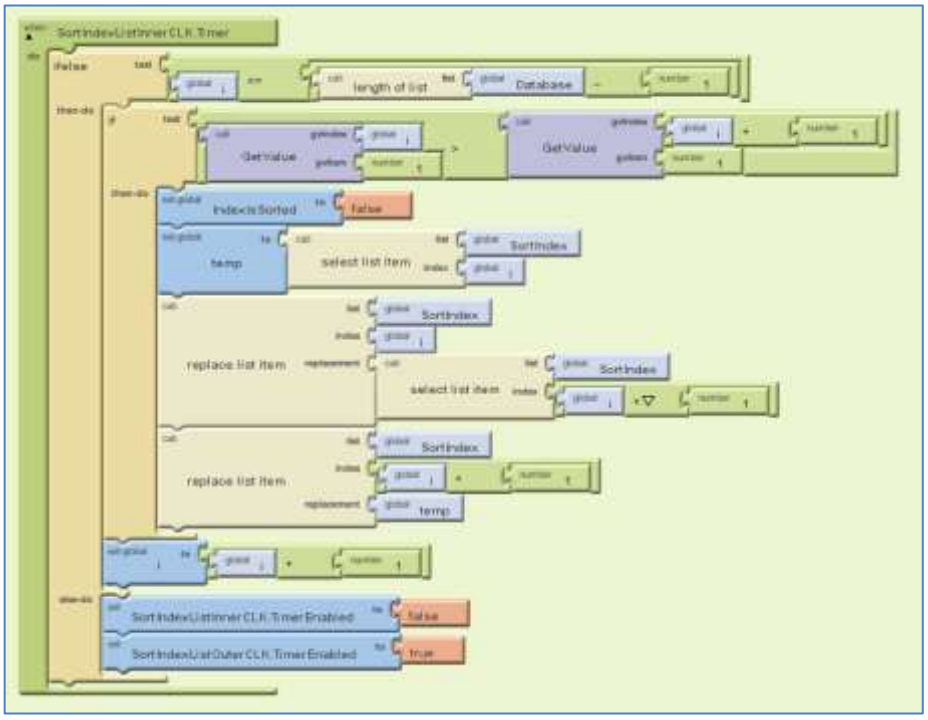


Gambar 3.29 *Block Index* Algoritma *Bubble Sort*

n. Pengkodean Algoritma *Bubble Sort*

Algoritma *bubble sort*, adalah sebuah algoritma penyusunan sederhana yang bekerja dengan cara melakukan pembandingan berulang-ulang terhadap daftar yang akan di sortir (disusun), yaitu membandingkan setiap pasangan *item* yang berdekatan dan menukarnya jika berada di urutan yang tidak tepat, sehingga nantinya dapat menunjukkan bahwa daftar tersebut telah diurutkan.

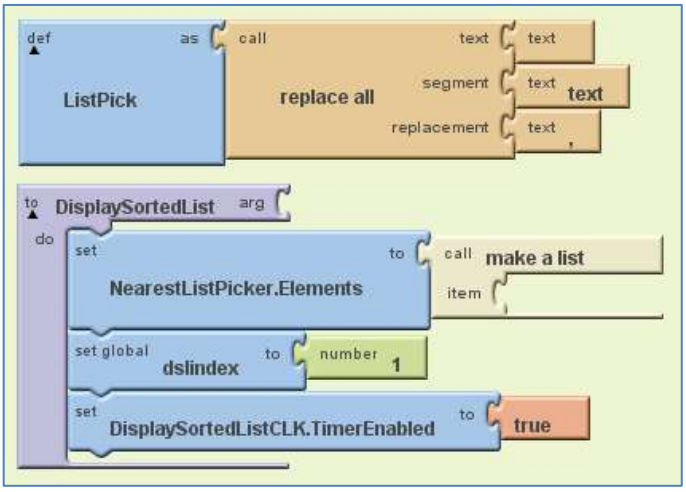
Algoritma *bubble sort* akan membandingkan hasil jarak yang telah didapatkan melalui penghitungan dengan *haversine formula*, kemudian menyusun hasil jarak yang telah dibandingkan tersebut kedalam sebuah *list* dari nilai jarak terkecil ke nilai jarak terbesar.



Gambar 3.30 *Block* Algoritma *Bubble Sort*

o. Pengkodean *List View NearestListPicker*

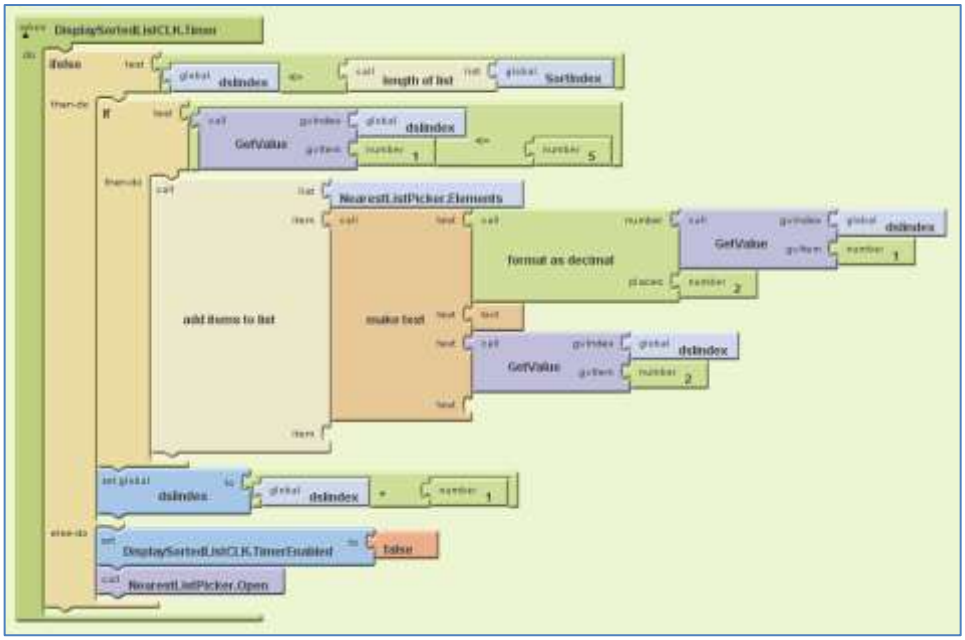
Pengkodean ini bertujuan untuk menghubungkan hasil sortir yang telah didapatkan melalui algoritma *bubble sort* ke dalam elemen komponen *NearestListPicker*, sehingga nantinya akan ditampilkan dalam bentuk *list* objek wisata beserta jaraknya ada aplikasi GeoTourism.



Gambar 3.31 *Block List View*

p. Pengkodean *Sorted List* dengan *Radius*

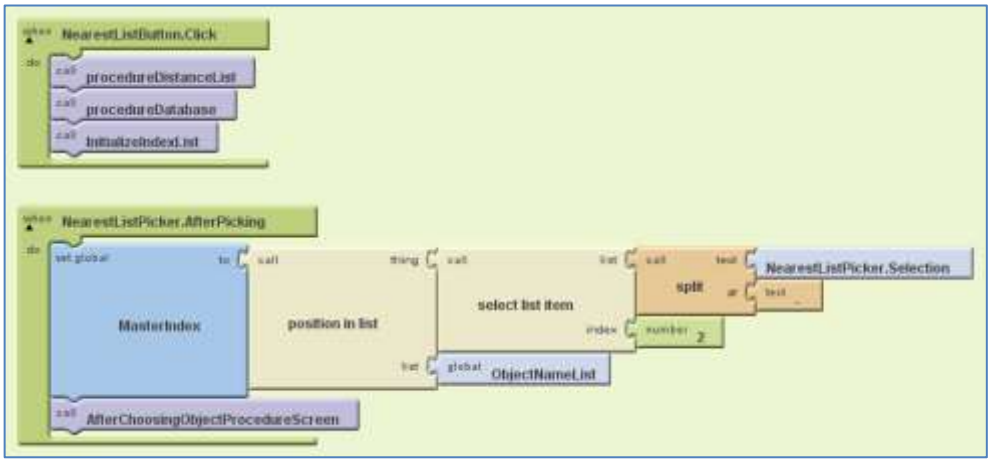
Data jarak yang telah disusun berdasarkan nilai jarak terdekat ke terjauh tersebut akan ditampilkan dalam bentuk *list* melalui komponen ListPicker. Dari semua data jarak tersebut tidak akan ditampilkan seluruhnya, melainkan hanya data jarak yang terdapat dalam radius 5 kilometer dari posisi *user*. Sehingga data jarak yang memiliki nilai lebih dari 5 kilometer tidak akan ditampilkan pada daftar *Near List* aplikasi GeoTourism.



Gambar 3.32 *Block Radius Sorted List*

q. Pengkodean *Display List* pada *Near List ListPicker*

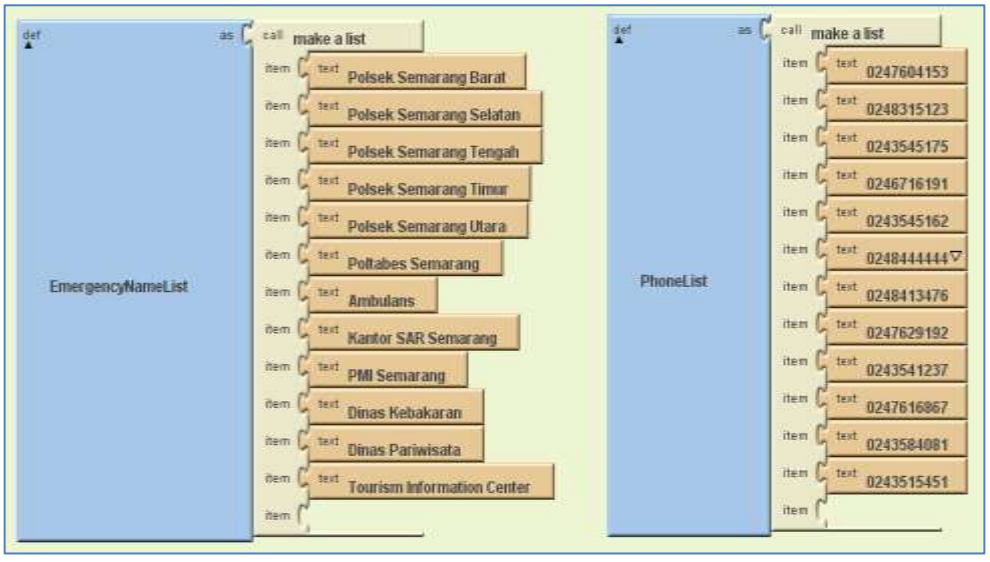
Langkah terakhir dari proses *button Near List* yaitu dengan membuat *block* penghubung antara hasil daftar objek wisata terdekat pada *Near List* ke *database* informasi objek wisata. Proses pelaksanaan *block* ini yaitu ketika *user* memilih salah satu objek wisata terdekat dari *button Near List*, maka aplikasi akan menampilkan informasi objek wisata yang sama persis ditampilkan ketika *user* memilih objek wisata melalui *button Tourism List*.



Gambar 3.33 *Block Near List AfterPicking*

r. Pengkodean *Database Block PhoneCall*

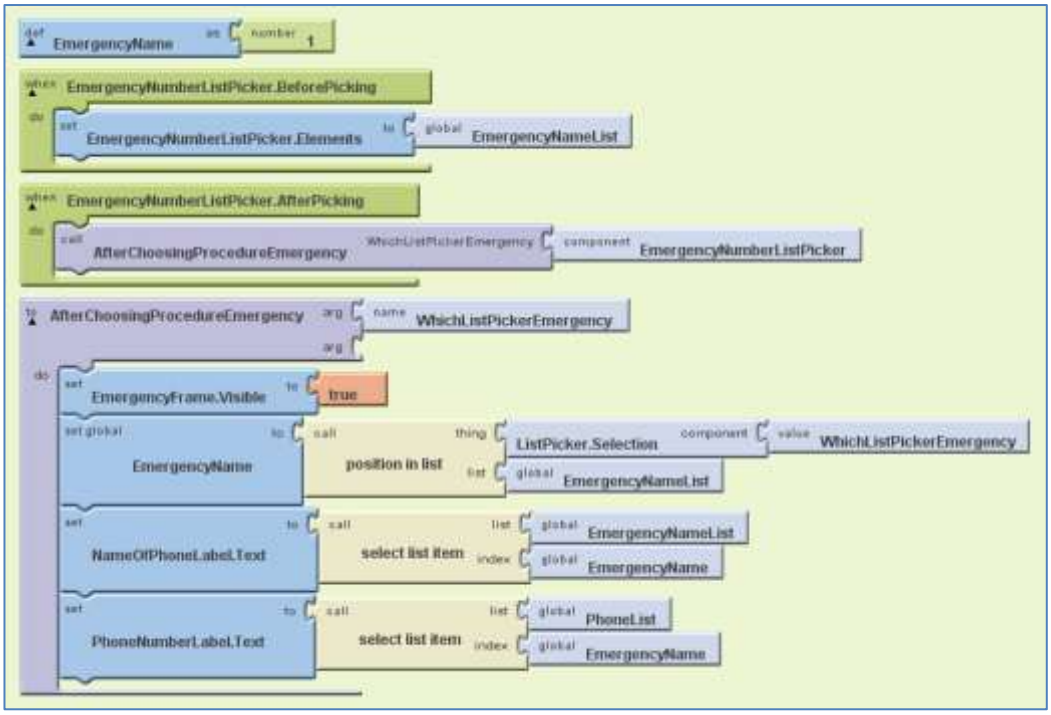
Pada aplikasi GeoTourism juga terdapat fasilitas panggilan telepon langsung melalui aplikasi. Salah satu penerapan fitur ini terdapat pada *button Phone* yang berisi daftar nomor telepon penting ataupun darurat di Kota Semarang. Langkah awal dari pengkodean untuk fitur ini adalah dengan membuat *database* dari nama dan nomor telepon penting melalui variabel *list* yang dihubungkan dengan komponen *index.*



Gambar 3.34 *Block database* pada *button Emergency*

s. Pengkodean *Procedure PhoneCall*

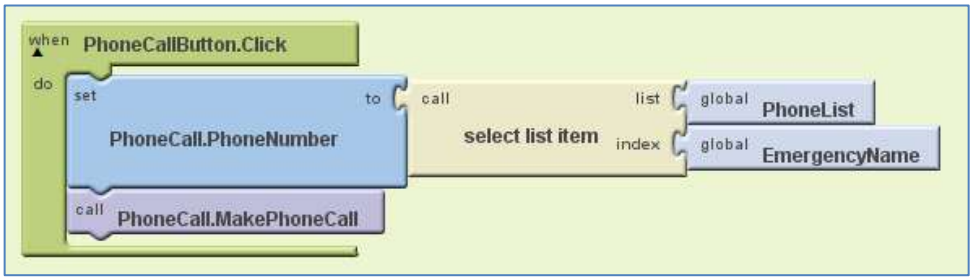
Block ini dibuat untuk menghubungkan antara *database* daftar nama pada *EmergencyNameList* dengan *database* yang berisi nomor telepon pada *PhoneList*. Prosedur *block* ini akan memanggil informasi mengenai nomor telepon yang telah dipilih melalui *EmergencyNumberListPicker button Phone,* kemudian menampilkan nama dan nomor telepon dalam bentuk teks, serta *button Call* pada *screen* aplikasi GeoTourism.



Gambar 3.35 *Block procedure PhoneCall*

t. Pengkodean *PhoneCall Activity*

Block ini berfungsi untuk menghubungkan nomor telepon yang dipilih ke fasilitas panggilan telepon pada *smartphone* Android. Dimana nantinya *button Call* pada aplikasi GeoTourism akan melakukan perintah *calling* atau panggilan terhadap nomor telepon yang telah dipilih oleh *user*. Perintah *calling* sendiri diproses dengan komponen *PhoneCall.MakePhoneCall* setelah *user* mengklik *button Call.*



Gambar 3.36 *Block Phone Call activity*

III.6 Uji Coba dengan *Emulator*

Setelah pengkodean sistem aplikasi selesai dilakukan. Langkah berikutnya adalah menguji *project* yang telah dibuat menggunakan *emulator*. Hal ini dilakukan untuk mengetahui apakah *project* yang dibuat dapat menjalankan fungsi-fungsinya sesuai dengan rancangan aplikasi. Cara menjalankan *emulator* yaitu pada halaman block editor klik *New Emulator* → *OK*, kemudian setelah *emulator* terbuka klik *Connect to Device* → *emulator-5556*, maka *project* akan ditampilkan pada *emulator*.

Gambar 3.37 Tampilan *Emulator*

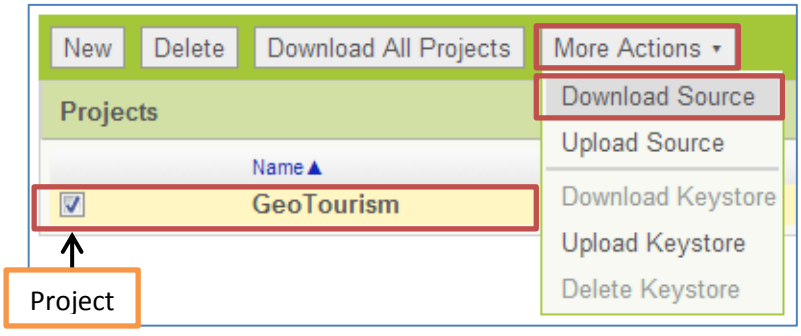
III.7 Implementasi

Pada bagian ini penulis melakukan tiga tahap implementasi yaitu *download project* dan aplikasi, instalasi aplikasi ke *handset* Android, dan *testing blackbox.*

III.7.1 Download *Project* dan Aplikasi

Pada *App Inventor*, *project* yang telah dibuat dapat langsung di *download*. *File project* dapat di *download* dalam dua jenis, yaitu *file source project* dan *file* aplikasi **.Apk*. File *source project* adalah *file *.zip* yang berisi sistem *design* dan *block* dari aplikasi yang telah dibuat. Sedangkan *file* aplikasi **.Apk* merupakan *file* dari hasil pembuatan *project* yang berupa aplikasi. *File* aplikasi **.Apk* dapat di instalasi ke perangkat *smartphone* Android.

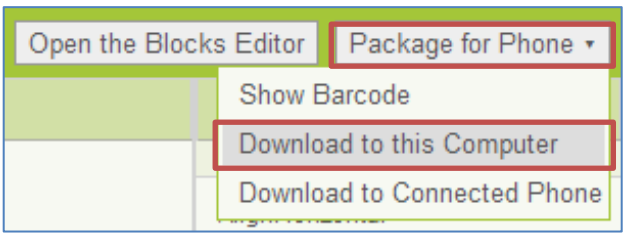
Untuk mendownload *file source project* pada *App Inventor* dapat dilakukan dengan cara membuka halaman *My Project*, kemudian centang *file project* yang akan di *download*. Selanjutnya klik *More Actions* dan pilih *Download Source*. Hasil dari *download file source project* ini berupa *file* berekstensi **.zip*. Pada *App Inventor* juga memungkinkan untuk melakukan *upload project* dengan cara memilih *Upload Source* pada *tab More Actions* di halaman *My project*. *File* yang akan di *upload* harus berekstensi **.zip*.



Gambar 3.38 Tampilan *download file source project* aplikasi

Sedangkan untuk men-*download file* aplikasi pada *App Inventor* yaitu dengan cara membuka halaman *Design*, kemudian klik tab *Package for Phone* dan pilih *Download to this Computer*. *File* hasil download akan berupa *file* aplikasi

berekstensi *.*Apk*. *File* inilah yang nantinya dapat di instalasi pada perangkat *smartphone* Android.



Gambar 3.39 Tampilan *download file* aplikasi

III.7.2 Intalasi Aplikasi ke *Handset* Android

Pada tahap ini penulis melakukan instalasi aplikasi ke perangkat *smartphone* Android. Aplikasi bisa didapatkan setelah melakukan proses *download* pada halaman *design App Inventor*. *File* aplikasi yang telah di *download* berformat *.*Apk.*

III.7.3 Testing *Blackbox*

Tahap terakhir dari fase implementasi adalah melakukan *testing* aplikasi. Hal ini dilakukan untuk mengetahui apakah fungsi yang telah dirancang mulai dari *design* aplikasi dan *activity* dari setiap sistem *block* dapat beroperasi sesuai rancangan. Disini penulis akan melakukan pengujian aplikasi berdasarkan rancangan *activity* yang telah dibuat.