

3. METODOLOGI PENELITIAN

3.1 Analisis Kebutuhan

Sering terjadinya pencurian kendaraan bermotor membuat pemilik geram terhadap pencuri. Khususnya sepeda motor yang masih belum dilengkapi sistem pengaman yang memadai. Pemilik pun tidak dapat mengetahui kemana kendaraan dibawa pergi pencuri dan harus melapor ke pihak kepolisian. Sehingga dibutuhkan Sistem Lacak Kendaraan untuk melacak posisi kendaraan yang telah dicuri. Sistem Lacak Kendaraan ini dibuat dalam bentuk prototipe yang terdiri dari alat pelacak, aplikasi *mobile* dan web server. Alat pelacak untuk melacak posisi pada kendaraan bermotor yang terhubung dengan web server dan aplikasi *mobile* menampilkan data dari web server di pemilik *smartphone*.

Pada pembuatan alat pelacak dibutuhkan alat yang dapat menerima data dari satelit dan mengirimkan data posisi terkini berupa koordinat latitude dan longitude serta terhubung dengan jaringan internet. Alat pelacak menerima data posisi terkini menggunakan GPS dan mengirim data melalui protokol HTTP serta GPRS untuk terhubung dengan jaringan internet.

Pada pembuatan aplikasi dibutuhkan data posisi koordinat alat pelacak yang akan ditampilkan pada *smartphone*. Dalam pembangunan aplikasi dibutuhkan API untuk menampilkan data dari server. API yang dibuat yaitu berupa data posisi koordinat latitude dan longitude dari alat pelacak. API dibuat menggunakan PHP Native untuk proses penyimpanan dan pertukaran data antara alat pelacak dan aplikasi *mobile*.

Pada pembangunan Prototipe Sistem Lacak Kendaraan membutuhkan alat pelacak, aplikasi *mobile* dan web server. Spesifikasi komponen alat pelacak, perangkat keras komputer dan perangkat lunak yang digunakan dalam pembuatan aplikasi Android menggunakan *Hybrid Technology*. Perangkat keras komputer yang digunakan adalah :

1. Personal Komputer
2. Intel Core i5-3470S 2.9GHz

3. Memory 8 GB 1600 MHz DDR3
4. Nvidia GeForce GTX 750 Ti 2047MB

Komponen-komponen alat pelacak yang digunakan untuk menerima data posisi dari GPS dan mengirim data ke server adalah :

1. Arduino Uno R3
2. Modul SIM808
3. Kabel *Jumper Female to Male*
4. Kabel USB to Arduino
5. Baterai 9 Volt
6. GPS Antenna
7. GSM/GPRS Antenna
8. *SIM Card*

Perangkat lunak yang digunakan dalam pembangunan alat pelacak dan aplikasi *mobile* dengan teknologi *hybrid* adalah :

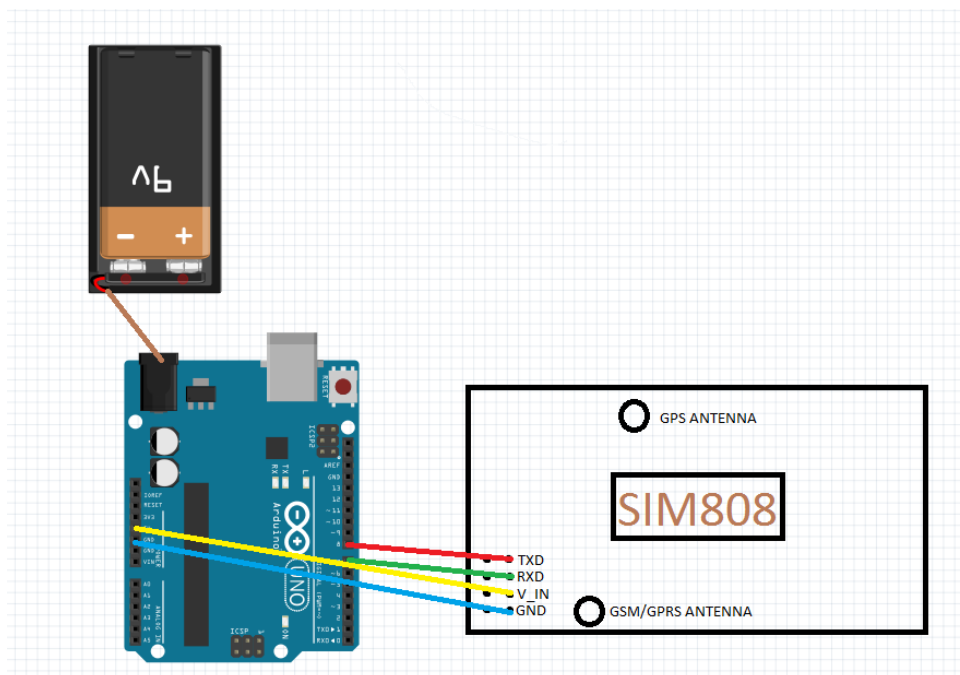
1. Sistem Operasi Windows 10 64-Bit
2. Sublime Text 3
3. Command Prompt
4. Node JS atau *Node Package Manager* (npm)
5. Ionic dan Cordova (menggunakan npm)
6. Bower
7. *Android Software Development Kit* (SDK)
8. *Java Development Kit* (JDK)
9. *Android Debug Bridge* (ADB)
10. *Android Virtual Device* (AVD)
11. *UC Browser Version 9.1.2* (11601.7.7)
12. *Arduino Software* (IDE)
13. *Library TinyGPS++*
14. *Library GSM*

3.2 Perancangan

Prototipe Sistem Lacak Kendaraan dirancang dengan beberapa tahap perancangan, yaitu perancangan rangkaian alat pelacak, perancangan RESTful web service, perancangan *storyboard* tentang alur aplikasi, perancangan struktur navigasi aplikasi, perancangan antarmuka tentang tampilan aplikasi dan perancangan alur program.

3.2.1 Perancangan Rangkaian Alat Pelacak

Perancangan rangkaian alat pelacak adalah perancangan komponen-komponen alat yang dirangkai menjadi satu kesatuan sistem agar dapat menerima data posisi koordinat latitude dan longitude dari alat tersebut dari satelit serta mengirim data ke database server. Komponen-komponen dihubungkan menggunakan kabel *jumper*, seperti berikut:

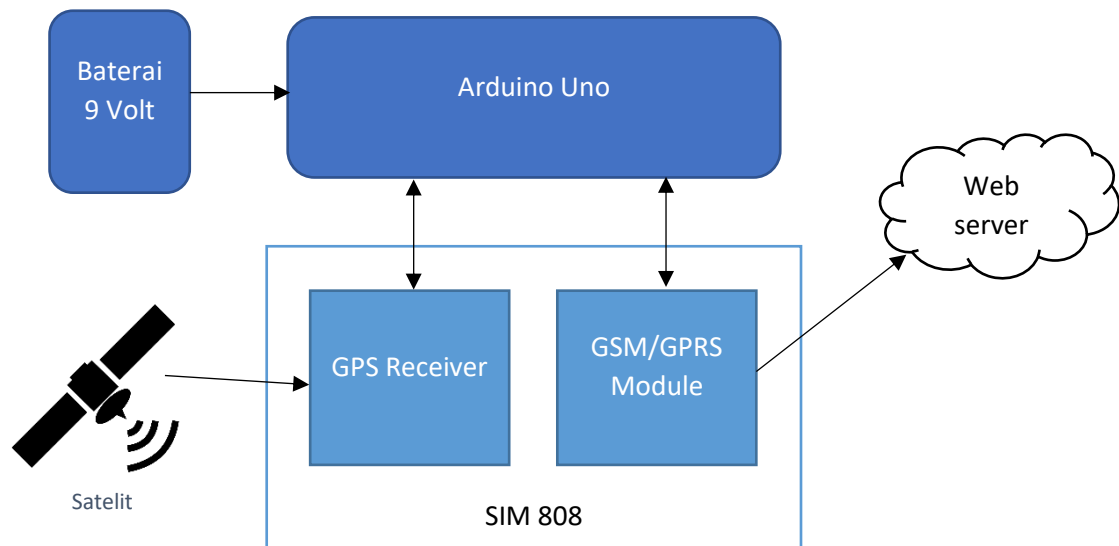


Gambar 3.1 Rangkaian Alat Pelacak

Pada Arduino Uno diberi input daya sebesar 9 volt dari baterai melalui power plug. Lalu pin power 5 volt dari arduino dihubungkan ke pin V_IN SIM808 untuk memberi daya listrik sehingga dapat berfungsi. Pin GND power arduino dihubungkan ke pin GND SIM808. Untuk mengirim dan menerima data, pin 7

arduino sebagai TXD yang dihubungkan ke pin RXD SIM808 dan pin 8 arduino sebagai RXD yang dihubungkan ke pin TXD SIM808. GPS antenna dihubungkan ke soket GPS. GSM/GPRS antena dihubungkan ke soket GSM/GPRS.

Cara kerja dari alat pelacak ini dapat dilihat dari diagram blok berikut:



Gambar 3.2 Diagram Blok Alat Pelacak

Pada Gambar 3.2 dapat dijelaskan bahwa baterai 9 volt memberi daya listrik semua komponen melalui power plug arduino dan diteruskan ke SIM 808. Arduino Uno sebagai mikrokontroler menjalankan program yang telah ditanamkan berupa menyalakan fitur GSM/GPRS dan GPS, menerima data posisi koordinat latitude dan longitude ke satelit, kemudian dikirim kembali menggunakan metode POST dengan protokol HTTP ke database server.

3.2.2 Perancangan RESTful Web Services

Perancangan ini digunakan untuk melakukan pertukaran data antar sistem atau aplikasi. *Web services* dibuat menggunakan PHP native untuk menerima data dari alat pelacak dan terhubung dengan database server serta mengubah data dalam bentuk JSON. Pada *web services* ini, data yang dikirim oleh alat pelacak akan diterima dengan metode GET dari params latitud dan longitude pada URL HTTP serta data disimpan pada database server.

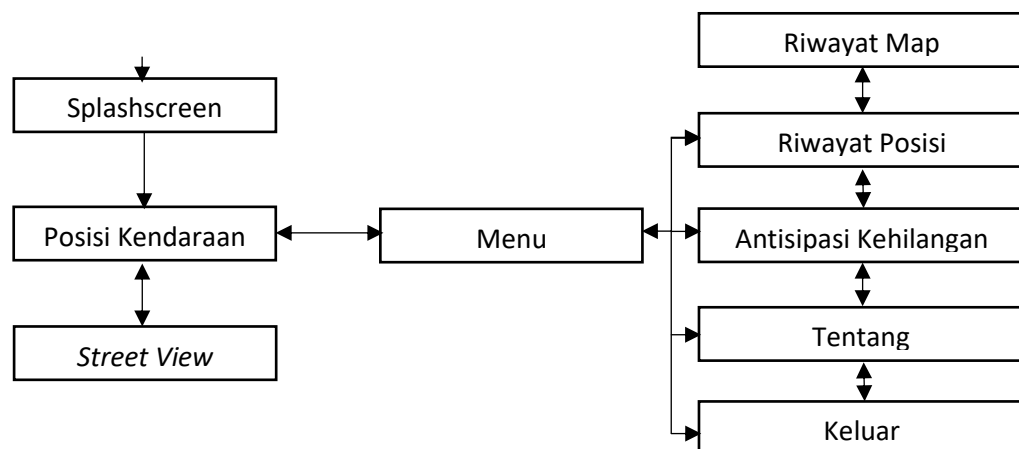
Pada *database server*, struktur tabel *database* terdiri dari 4 *field* data yang berbeda, yaitu *field* id, waktu, latitude dan longitude. *Field* id bertipe integer untuk memberi nomor urutan disetiap data yang masuk dan bersifat *auto_increment* serta *primary key*. *Field* waktu bertipe *datetime* untuk menentukan tanggal dan jam terkini pada setiap data yang masuk serta bersifat *current_time*. *Field* latitude bertipe *varchar* untuk data latitude yang dikirimkan dari alat pelacak. *Field* longitude bertipe *varchar* untuk data longitude yang dikirimkan dari alat pelacak.

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id	int(3)			No	None	AUTO_INCREMENT	Change Drop Browse distinct values Primary
2	waktu	datetime			No	CURRENT_TIMESTAMP		Change Drop Browse distinct values Primary
3	latitude	varchar(10)	latin1_swedish_ci		No	None		Change Drop Browse distinct values Primary
4	longitude	varchar(10)	latin1_swedish_ci		No	None		Change Drop Browse distinct values Primary

Gambar 3.3 Struktur Tabel *Database*

3.2.3 Perancangan Struktur Navigasi Aplikasi

Struktur navigasi merupakan rancangan hubungan atau rantai kerja yang menggambarkan sistem aplikasi secara keseluruhan. Aplikasi ini menggunakan struktur navigasi hirarki atau bercabang untuk menggambarkan sistem aplikasi. Struktur navigasi bercabang yang digunakan pada aplikasi ini karena aplikasi ini memiliki data-data yang beda satu sama lain pada setiap tampilan. Perpindahan tampilan tidak searah karena terdapat tampilan sidemenu yang disediakan oleh kerangka kerja Ionic, struktur navigasi pembangunan aplikasi informasi posisi kendaraan dapat dilihat pada Gambar 3.4.

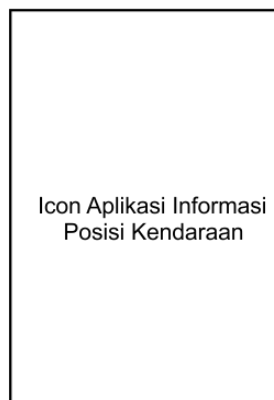


Gambar 3.4 Struktur Navigasi Aplikasi *Mobile*

Struktur navigasi ini dimulai dari *splashscreen*, yang berlanjut secara otomatis ke tampilan posisi kendaraan. Pada tampilan menu akan terdapat 5 kategori yang berupa *ion-item* atau pada struktur navigasi ini dikatakan sebagai menu posisi kendaraan, riwayat posisi, antisipasi kehilangan, tentang dan keluar.

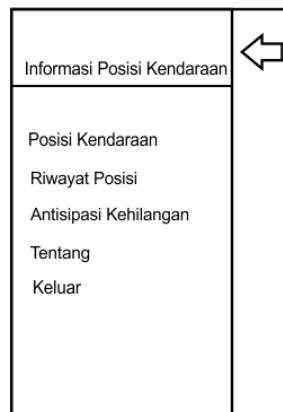
3.2.4 Perancangan Antarmuka Aplikasi

Perancangan antarmuka merupakan salah satu unsur terpenting dalam pembangunan aplikasi. Perancangan antarmuka dapat menentukan keberhasilan dari sebuah produk. Antarmuka yang dibuat harus semenarik mungkin agar pengguna tertarik untuk menggunakan aplikasi informasi posisi kendaraan ini. Sebuah antarmuka dirancang sesuai kebutuhan dari aplikasi. Rancangan ini dibuat dengan menyesuaikan pengguna yang menjadi tujuan akhir dari aplikasi. Rancangan pertama adalah pembuatan *splashscreen*.

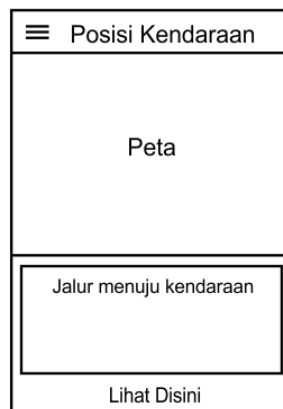


Gambar 3.5 Perancangan Antarmuka *Splashscreen*

Gambar 3.5 merupakan rancangan awal ketika masuk kedalam aplikasi informasi posisi kendaraan. Ketika pengguna membuka aplikasi ini maka akan tampil *splashscreen* seperti diatas. Setelah itu akan ada tampilan selanjutnya dimana terdapat 4 menu yaitu menu posisi kendaraan, riwayat posisi, antisipasi kehilangan, tentang dan satu *button* keluar.

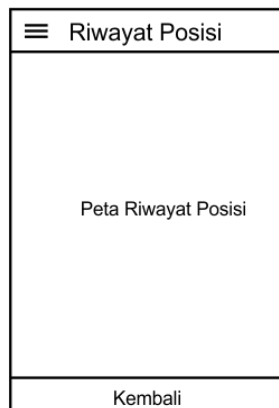


Gambar 3.6 Perancangan Antarmuka *Sidemenu*



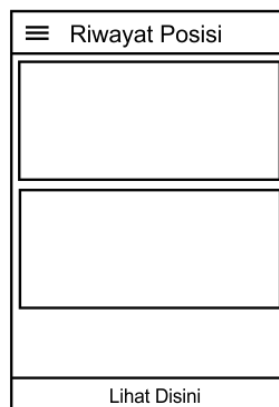
Gambar 3.7 Perancangan Antarmuka Posisi Kendaraan

Gambar 3.6 merupakan rancangan tampilan menu, dimana tampilan ini untuk memilih menu-menu pada aplikasi. Pada tampilan menu posisi kendaraan, tampilan ini tedapat sebuah *card panel* yang didalam nya terdapat peta dari Google Maps, informasi mengenai titik terkini pengguna aplikasi, tanggal dan jam terdeteksi, jalur dari titik pengguna ke titik kendaraan, jarak tempuh dari titik pengguna ke titik kendaraan dan waktu tempuh dari titik pengguna ke titik kendaraan yang terdapat pada Gambar 3.7. Pada menu ini terdapat satu menu tambahan, yaitu tombol “Lihat Disini” untuk pindah ke mode *Street View* dan melihat kondisi lingkungan sekitar pada titik posisi kendaraan seperti pada Gambar 3.8.

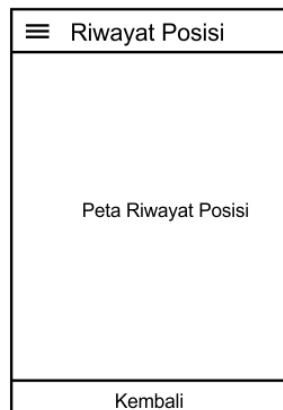


Gambar 3.8 Perancangan Antarmuka *Street View*

Jika pengguna kembali memilih menu pada sidemenu, akan keluar beberapa menu seperti pada Gambar 3.6. Jika pengguna memilih menu riwayat posisi, akan muncul informasi riwayat posisi kendaraan. Informasi tersebut disajikan dalam bentuk *list card* berupa tanggal dan jam terdeteksi serta titik koordinat latitude dan longitude seperti pada Gambar 3.9. Pada menu ini terdapat 1 menu tambahan, yaitu tombol “Lihat Disini” untuk melihat titik-titik koordinat yang pernah dilewati kendaraan dalam bentuk peta dari Google Maps seperti pada Gambar 3.10.

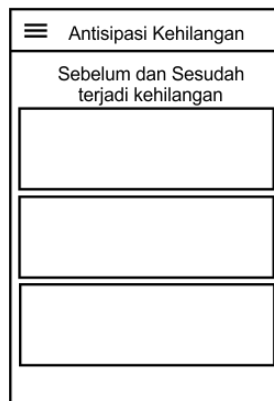


Gambar 3.9 Perancangan Antarmuka Riwayat Posisi



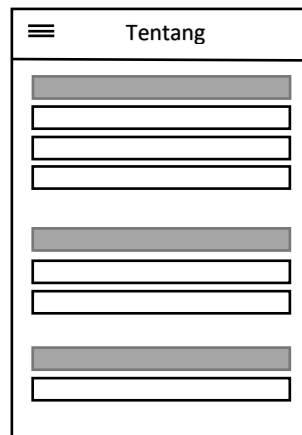
Gambar 3.10 Perancangan Antarmuka Peta Riwayat Posisi

Jika pengguna memilih menu antisipasi kehilangan, pengguna dapat mengetahui apa yang harus dilakukan sebelum dan sesudah terjadinya kehilangan kendaraan. Pada tampilan antisipasi kehilangan terdapat *card panel* yang berisi informasi yang dilengkapi dengan animasi gambar dan deskripsi mengenai apa yang dapat dilihat pada Gambar 3.11.



Gambar 3.11 Perancangan Antarmuka Antisipasi Kehilangan

Pada tampilan yang dirancang pada Gambar 3.12 menjelaskan tentang informasi pembuatan aplikasi ini, aplikasi dibuat menggunakan *Hybrid Technology* yaitu kerangka kerja Ionic dan menggunakan beberapa library yang dibutuhkan dalam aplikasi informasi posisi kendaraan.



Gambar 3.12 Perancangan Antarmuka Tentang

3.2.5 Perancangan Storyboard

Perancangan *storyboard* adalah perancangan tentang jalan cerita atau alur cerita dari aplikasi yang disajikan dalam bentuk gambar. *Storyboard* ini akan menggambarkan tentang alur dari aplikasi informasi posisi kendaraan. User atau pengguna akan disajikan beberapa menu untuk memilih tampilan informasi yang ingin ditampilkan pada *smartphone* pengguna.

Pengguna akan dihadapkan dengan 4 pilihan menu yang diatur oleh tampilan sidemenu dan satu button keluar untuk menutup aplikasi. Menu pada aplikasi ini terdiri dari menu posisi kendaraan, menu riwayat posisi, menu antisipasi kehilangan dan menu tentang.

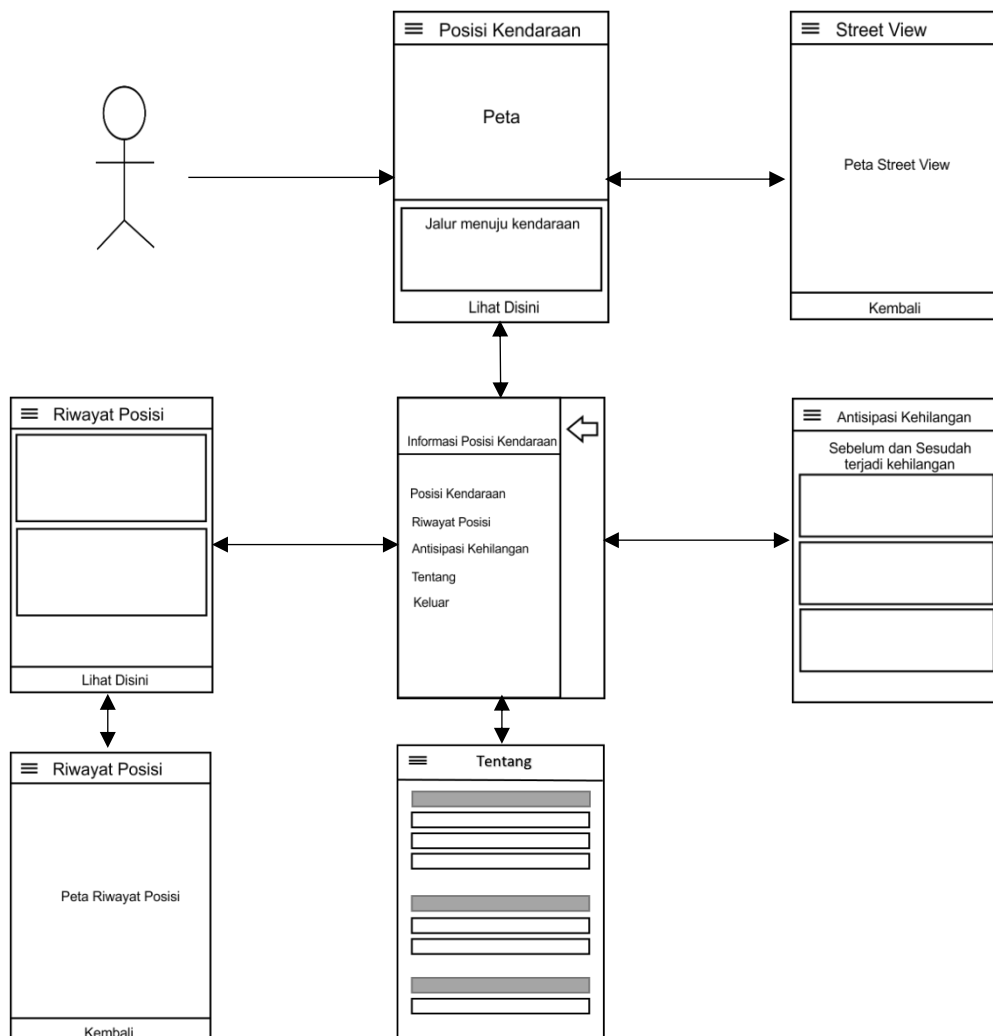
Apabila pengguna memilih menu posisi kendaraan, maka akan muncul posisi terkini *smartphone* dan informasi posisi kendaraan. Informasi tersebut disajikan dalam peta dari Google Maps dan terdapat informasi posisi terkini *smartphone*, posisi kendaraan, tanggal dan jam terdeteksi dan jalur dari posisi terkini pengguna sampai posisi kendaraan. Pada menu ini terdapat 1 menu tambahan, yaitu menu *Street View* untuk melihat kondisi lingkungan sekitar pada titik posisi kendaraan.

Apabila pengguna memilih menu riwayat posisi, maka akan muncul informasi riwayat posisi kendaraan. Informasi tersebut disajikan dalam bentuk *list card* berupa tanggal dan jam terdeteksi serta titik koordinat latitude dan longitude.

Pada menu ini terdapat satu menu tambahan, yaitu titik-titik koordinat yang pernah dilewati kendaraan dalam bentuk peta dari Google Maps.

Apabila pengguna memilih menu antisipasi kehilangan, maka akan muncul tips atau langkah-langkah untuk pengguna cara mengantisipasi agar kendaraan tidak mudah hilang maupun setelah kendaraan hilang. Informasi disajikan dalam bentuk gambar dan dideskripsikan pada setiap informasi antisipasi kehilangan.

Apabila pengguna memilih menu tentang, akan muncul informasi tentang pembuat, *library* dan software yang digunakan untuk membuat aplikasi ini. Menu keluar untuk keluar dari aplikasi.

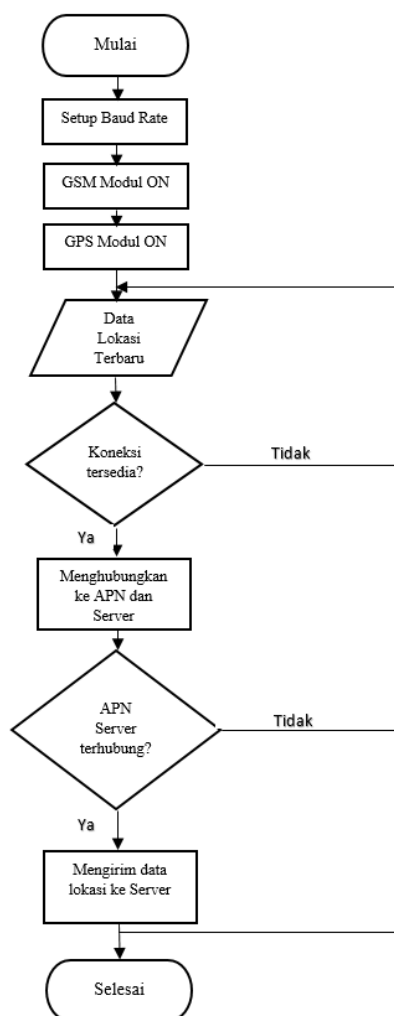


Gambar 3.13 Storyboard Aplikasi Mobile

3.2.6 Perancangan Alur Program

Perancangan sistem lacak kendaraan ini juga menggunakan *Unified Model Language* (UML) untuk memvisualisasikan rancangan model sistem atau alur program. Model sistem ini akan dirancang menggunakan sebuah *flowchart* untuk alur program alat pelacak, dua buah diagram yang terdapat pada UML yaitu Use Case Diagram dan Activity Diagram untuk alur program aplikasi *mobile* dan Use Case Diagram, Activity Diagram dan Class Diagram untuk alur program Prototipe Sistem Lacak Kendaraan secara keseluruhan.

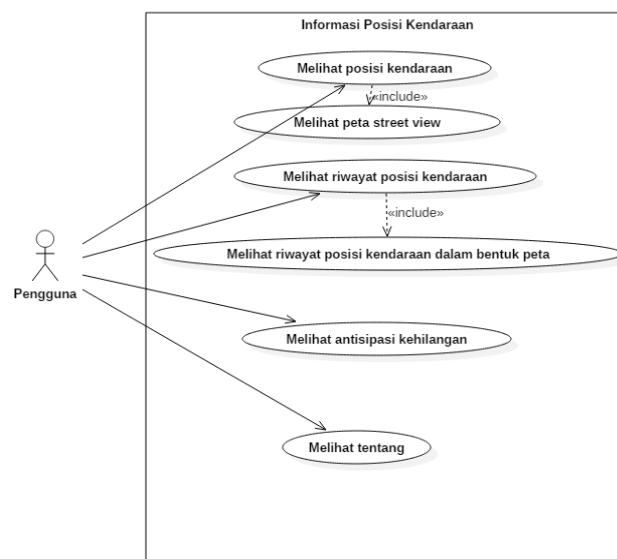
Pada alat pelacak, alur program yang digunakan yaitu *flowchart*. Alur program *flowchart* menggambarkan simbol atau notasi urutan dan hubungan antar proses beserta pernyataan dari suatu sistem.



Gambar 3.14 *Flowchart* Alat Pelacak

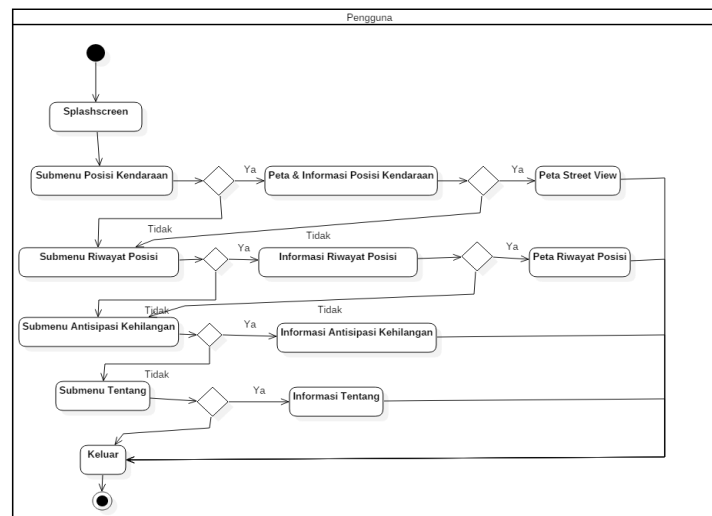
Pada Gambar 3.14 menjelaskan bahwa program akan mengatur *baud rate* arduino uno dan SIM 808 terlebih dahulu. Kemudian menyalakan fitur GSM Modul dan GPS Modul. Setelah menyalakan GPS Modul, program menerima data posisi koordinat alat tersebut ke satelit. GSM Modul memeriksa koneksi jaringan yang ada. Saat koneksi jaringan tersedia, GSM Modul akan menghubungkan dengan APN dan Server untuk mengirim data lokasi yang telah diterima ke database server. Program akan kembali berulang ketika data lokasi telah dikirim ke database server, koneksi tidak tersedia dan tidak terhubung dengan APN dan Server.

Pada Use Case Diagram Aplikasi menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Diagram ini akan mempresentasikan antara *actor* dengan sistem yang ada terdapat pada Gambar 3.15.



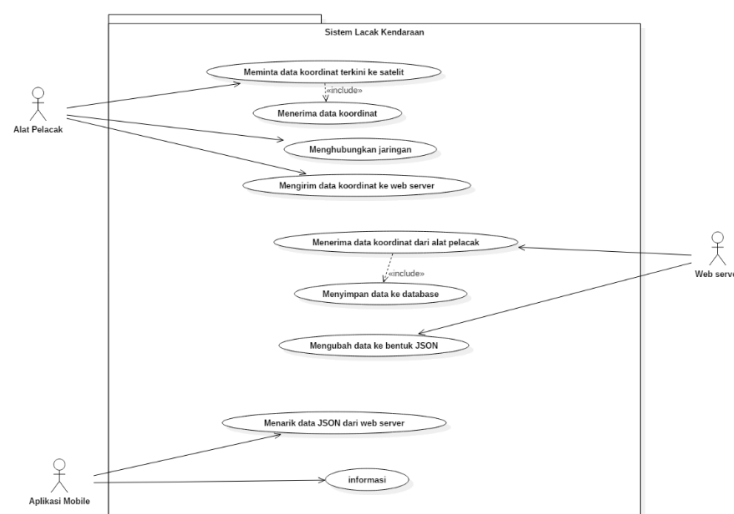
Gambar 3.15 Use Case Diagram Aplikasi *Mobile*

Pada rancangan Use Case Diagram Aplikasi terlihat bahwa pengguna berinteraksi secara langsung dengan aplikasi informasi, dimana pengguna dapat melihat informasi posisi kendaraan, *street view*, riwayat posisi kendaraan, antisipasi kehilangan dan tentang. Selanjutnya alur dari aplikasi akan digambarkan dengan Activity Diagram. Diagram ini akan menggambarkan alur aktifitas di dalam aplikasi, bagaimana aktifitas berawal dan aplikasi berakhir yang dapat dilihat pada Gambar 3.16.



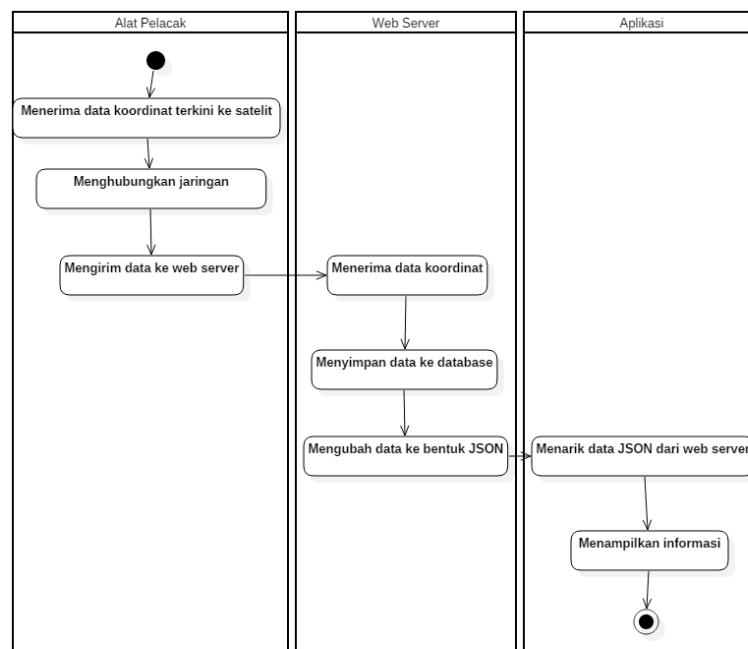
Gambar 3.16 Activity Diagram Aplikasi *Mobile*

Pada Gambar 3.16 menggambarkan alur aktifitas dari aplikasi *mobile* informasi posisi kendaraan. Aplikasi ini dimulai dari sebuah *smartphone* yang di dalamnya terdapat aplikasi *mobile* informasi posisi kendaraan. Pada aplikasi informasi posisi kendaraan, akan terdapat tampilan awal sebelum masuk kedalam menu aplikasi yaitu tampilan *splashscreen*, dimana tersedia 4 menu yang akan menampilkan beberapa tampilan informasi posisi kendaraan. Apabila pengguna memilih salah satu dari menu tersebut, maka menu tersebut akan mengarah kepada informasi yang dimiliki setiap menu tersebut.



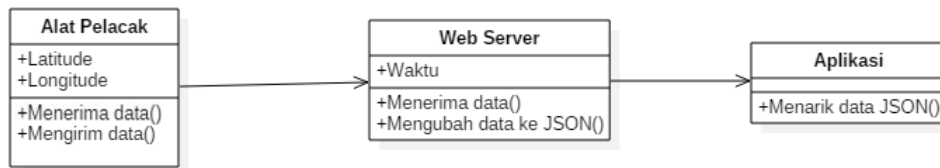
Gambar 3.17 Use Case Diagram Prototype Sistem Lacak Kendaraan

Pada gambar 3.17 menggambarkan kegiatan yang terjadi di sistem lacak kendaraan. Terdapat 3 *actor* pada Use Case Diagram Prototipe Sistem Lacak Kendaraan, yaitu Alat Pelacak, Web Server dan Aplikasi *Mobile*. Tugas tiap *actor* yaitu alat pelacak menerima data posisi dari satelit dan menghubungkan jaringan ke web server. Kemudian alat pelacak mengirim data lokasi ke web server. Web server menyimpan data lokasi yang diterima dari alat pelacak, data koordinat dikonversi ke dalam bentuk JSON untuk digunakan sebagai API. Aplikasi menarik data lokasi JSON dari server dan menampilkan data JSON ke dalam informasi.



Gambar 3.18 Activity Diagram Prototipe Sistem Lacak Kendaraan

Pada gambar 3.18 menggambarkan alur aktivitas dari keseluruhan kerja sistem secara garis besar. Prototipe Sistem Lacak Kendaraan dimulai dari alat pelacak menerima data posisi dari satelit dan menghubungkan jaringan ke web server. Kemudian alat pelacak mengirim data posisi ke web server. Web server menyimpan data posisi yang diterima dari alat pelacak. Data-data dikonversi ke dalam bentuk JSON untuk digunakan sebagai API *web service*. Aplikasi menarik data posisi JSON dari server dan menampilkan data JSON ke dalam informasi.



Gambar 3.19 Class Diagram Prototipe Sistem Lacak Kendaraan

Pada gambar 3.19 menggambarkan tentang hubungan 3 *class* yaitu alat pelacak, web server dan aplikasi *mobile*. Pada *class* alat pelacak terdiri dari 2 *attribut* dan 2 *operations*. 2 *attribut* meliputi latitude dan longitude sebagai data posisi koordinat dari alat pelacak tersebut. 2 *operations* meliputi menerima data posisi koordinat dari satelit dan mengirim data posisi koordinat ke web server.

Pada *class* web server terdiri dari satu *attribut* dan 2 *operations*. *Attribut* pada web server yaitu waktu. Waktu berperan untuk memberi keterangan waktu tanggal dan jam pada data yang telah diterima web server dari alat pelacak. 2 *operations* meliputi menerima data dari alat pelacak dan mengubah data dari *database* ke dalam bentuk JSON.

Pada *class* Aplikasi *mobile* terdiri dari satu *operations* yaitu menarik data JSON dari API web server.

3.3 Pembuatan Prototipe Sistem Lacak Kendaraan

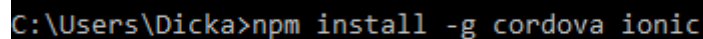
Setelah melakukan perancangan rangkaian alat pelacak, RESTful web service, *storyboard*, struktur navigasi aplikasi, antarmuka dan alur program. Dalam pembuatan Prototipe Sistem Lacak Kendaraan terdiri dari 3 tahap, yaitu tahap pembuatan aplikasi *mobile* dengan Ionic, RESTful *web services* dan alat pelacak. Tahap-tahap yang dilakukan adalah pelaksanaan dari rancangan yang telah dibuat secara matang dan terperinci. Seluruh perancangan akan diimplementasikan ke dalam pembangunan program dan dalam pembangunan program terdapat beberapa proses dan langkah-langkah.

3.3.1 Pembuatan Aplikasi Mobile dengan Ionic

Pembuatan Ionic dalam aplikasi informasi posisi kendaraan ini menggunakan tampilan *sidemenu* yang telah disediakan Ionic. Pembuatan halaman-halaman tampilan posisi kendaraan dengan mengikuti judul menu pada *sidemenu project* Ionic. Pada tampilan utama akan menampilkan peta posisi kendaraan paling terbaru dengan memanfaatkan *library* dari Google Maps.

3.3.1.1 Persiapan Project Ionic

Sebelum memulai pembuatan aplikasi atau pengkodean program Ionic, ada beberapa dependency yang harus terpasang di dalam komputer kita, seperti *Node Package Manager* (npm), *Front-End Package Manager* (bower) , Ionic dan Cordova. Pertama yang harus dilakukan adalah menginstal npm terlebih dahulu yang dapat diunduh dari <https://nodejs.org/en/download/>, tahap selanjutnya menginstal Ionic bersamaan dengan Cordova.



```
C:\Users\Dicka>npm install -g cordova ionic
```

Gambar 3.20 Install Ionic dan Cordova

Ionic merupakan sebuah kerangka kerja untuk mempermudah pengembang *website* dalam membuat aplikasi *Cross Platform* tanpa harus menguasai pemrograman *Android Native* dan *Objectiv-C* untuk iOS. Dalam membangun aplikasi menggunakan Ionic, pengembang *website* sudah terbiasa dengan bahasa pemrograman yang digunakan karena Ionic menggunakan HTML, CSS dan JS yang dibungkus oleh Angular.

Ionic memerlukan *command prompt* pada komputer, konfigurasi Ionic seluruhnya dilakukan pada *command prompt* seperti mengambil berkas Ionic pun menggunakan *command prompt*.

Ionic serta menggunakan *plugin cordova-geolocation* agar dapat mendapatkan posisi terkini dari *smartphone* yang digunakan.

```
C:\Users\Dicka\Documents\Ionic\infolokasi>bower install ionic-material
```

Gambar 3.23 Install *Library* ionic-material

```
C:\Users\Dicka\Documents\Ionic\infolokasi>bower install ngmap
```

Gambar 3.24 Install *Library* ngMap

```
C:\Users\Dicka\Documents\Ionic\infolokasi>npm install angularjs-geolocation@0.1.3
```

Gambar 3.25 Install *Library* angularjs-geolocation

```
C:\Users\Dicka\Documents\Ionic\infolokasi>cordova plugin add cordova-plugin-geolocation
```

Gambar 3.26 Install *Plugin* cordova-geolocation

Setelah mengunduh *library* dan *plugin* yang dibutuhkan menggunakan bower, setiap berkas pada folder *library* yang telah diunduh harus melakukan *path* berkas kedalam *index.html* di dalam folder *www/templates/*. Apabila berkas CSS menggunakan *tag link* dan jika berkas JS menggunakan *tag script*.

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="initial-scale=1, maximum-scale=1, user-scalable=no, width=device-width">
  <title>Informasi Posisi Kendaraan</title>
  <link href="https://fonts.googleapis.com/css?family=RobotoDraft:400,500,700,400italic" rel="stylesheet">
  <link href="lib/ionic/css/ionic.css" rel="stylesheet">
  <link href="lib/ionic-material/dist/ionic.material.min.css" rel="stylesheet" />
  <link rel="stylesheet" href="css/animate/animate.min.css" media="screen" title="no title" charset="utf-8">
  <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
  <link href="css/style.css" rel="stylesheet">
  <script src="lib/ionic/js/ionic.bundle.js"></script>
  <script src="lib/ionic-material/dist/ionic.material.min.js"></script>
  <script src="cordova.js"></script>
  <script src="js/app.js"></script>
  <script src="js/controllers/Menu.js"></script>
  <script src="js/controllers/TentangController.js"></script>
  <script src="js/controllers/AntisipasiKehilangan.js"></script>
  <script src="js/services.js"></script>
  <script src="js/controllers/DataController.js"></script>
  <script src="https://maps.googleapis.com/maps/api/js?libraries=places,visualization,drawing,geometry,places"></script>
  <script type="text/javascript" src="lib/ngmap/build/scripts/ng-map.min.js"></script>
  <script src="lib/ngCordova/dist/ng-cordova.js"></script>
  <script src="lib/ngmap/build/scripts/ng-map-min.js"></script>
  <script src="https://rawgit.com/allenhkim/angularjs-google-maps/master/build/scripts/ng-map.js"></script>
  <script src="https://code.angularjs.org/1.3.15/angular.js"></script>
  <script src="https://maps.google.com/maps/api/js"></script>
</head>
<body ng-app="app">
  <ion-nav-view></ion-nav-view>
  <div id="modal" class=""></div>
</body>
</html>
```

Gambar 3.27 *Path Link Library* yang Digunakan

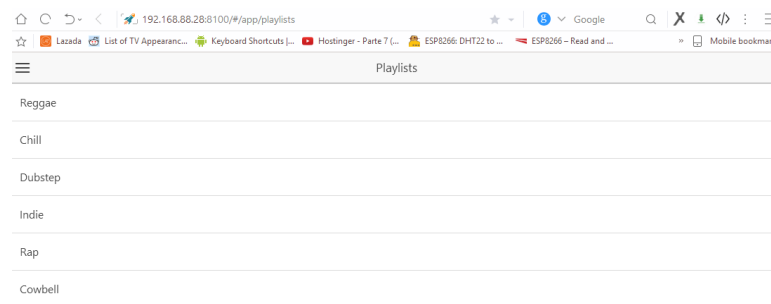
Pada Angular setiap pemberian *library* baru, Angular harus mengetahui tentang *library* tersebut. Angular terdapat *module* untuk mendefinisikan aplikasi

pada *tag* HTML dengan perintah *ng-app*. Setiap pembuatan *controllers* atau *library* baru, nama *controllers* atau *library* tersebut harus dimasukan kedalam *module app* Angular.

```
var app = angular.module('app', ['ionic', 'ionic-material', 'app.services', 'ngMap', 'ngCordova']);
```

Gambar 3.28 *Module* pada Angular

Setelah sudah melakukan *path* berkas pada *index.html* dan memasukan nama *library* ke dalam *module* Angular, terminal, untuk mencoba *project* Ionic menggunakan coba untuk menjalankan Ionic. Konfigurasi Ionic selalu didalam perintah *ionic serve*.



Gambar 3.29 Ionic pada Web Browser

Ionic tidak hanya dijalankan di dalam sistem operasi Andriod, akan tetapi aplikasi Ionic terbentuk oleh HTML, CSS dan JS. Jadi aplikasi Ionic dapat dibuka didalam *web browser*. Tampilan diatas adalah *sidemenu default* dari Ionic dan ada beberapa perintah untuk menjalankan Ionic Server.

Perintah *restart* atau *r* untuk mengulang aplikasi *client* dari folder *root*, *goto* atau *g* untuk pindah ke lain tampilan dengan menggunakan URL, *consolelogs* atau *c* untuk melihat kerja dari Ionic dan melihar *error* dari aplikasi yang dibuat, *quit* atau *q* untuk menutup aplikasi Ionic yang sedang berjalan pada Ionic Server.

Pada pembangunan aplikasi informasi posisi kendaraan, menggunakan API yang sudah dibuat pada *web services* sebelumnya, pada Ionic menggunakan Angular *\$http* untuk menarik data JSON dari API.

Pada setiap *ion-item* memiliki *ui-sref* yang berfungsi sebagai alamat untuk perpindahan alamat. Angular bekerja untuk melakukan *routing* antara menu dan halaman .html yang akan dipanggil dalam *routing*. Sistem kerja Angular dalam *routing* dapat disebut sebagai *HTML templating*. Pada Ionic untuk melakukan *routing* menggunakan *library* angular-ui-route.

Pada *routing* Angular, terdapat *state* untuk *route* ke tujuan halaman yang sudah ditentukan oleh *ui-sref* di dalam *file* .html di *www/templates*. Karena Ionic menggunakan bahasa *website*, pada *route* terdapat *Url Address Bar* yang berguna ketika Ionic Server dijalankan terbuka didalam *browser*. Cara penggunaan *route* Angular Ionic terdapat pada *file menu.html* atribut pada *tag* HTML yaitu *ui-sref*. pada *routing* halaman menggunakan *library* angular-ui-router, *library* tersebut milik Angular dalam melakukan perpindahan halaman yang biasa disebut dengan *HTML templating*. Penulisan kode program *route* pada *file* app.js terdapat pada Gambar 3.33.

```
app.config(function ($stateProvider, $urlRouterProvider) {
  $stateProvider

  .state('app', {
    url: '/app',
    abstract: true,
    templateUrl: 'templates/menu.html',
    controller: 'MenuController'
  })

  .state('app.AntisipasiKehilangan', {
    url: '/AntisipasiKehilangan',
    views: {
      'menuContent': {
        templateUrl: 'templates/AntisipasiKehilangan.html',
        controller: 'AntisipasiController'
      }
    }
  })

  .state('app.DataTerbaru', {
    url: '/DataTerbaru',
    views: {
      'menuContent': {
        templateUrl: 'templates/DataTerbaru.html',
        controller: 'DataTerbaruController'
      }
    }
  })

  .state('app.DataStreet', {
    url: '/DataStreet',
    views: {
      'menuContent': {
        templateUrl: 'templates/DataStreet.html',
        controller: 'DataStreetController'
      }
    }
  })

  .state('app.DataRiwayat', {
    url: '/DataRiwayat',
    views: {
      'menuContent': {
        templateUrl: 'templates/DataRiwayat.html',
        controller: 'DataRiwayatController'
      }
    }
  })
})
```

Gambar 3.32 Penggalan Kode Program *Route* pada *Sidemenu*

Ionic menggunakan Angular untuk memberikan aktifitas-aktifitas pada setiap halaman .html yang biasa disebut sebagai *controller*. Pembahasan *controller*

akan lebih terstruktur apabila dibahas bersamaan dengan berkas .html yang berada pada Gambar 3.27.

3.3.1.3 Pembuatan Halaman Posisi Kendaraan

Pembuatan aplikasi informasi posisi kendaraan memanfaatkan Angular data *Binding* untuk menerapkan JSON dari API kedalam tampilan HTML yang berada pada *www/templates/DataTerbaru.html* dan menggunakan *library* ngMap untuk menampilkan data posisi terbaru kedalam peta dari Google Maps dengan memanfaatkan data latitude dan longitude yang berada di dalam JSON.

Tampilan halaman posisi kendaraan akan dibangun berdasarkan rancangan pada Gambar 3.7 dan akan muncul pertama kali ketika aplikasi dibuka setelah *splashscreen* pada aplikasi. Sebelum membuat tampilan pada berkas *DataTerbaru.html* terlebih dahulu membuat *controllers* untuk berkas html tersebut, terdapat pada Gambar 3.33.

```
angular.module('app')
.controller('DataTerbaruController', function($scope, services, $ionicLoading, $ionicPlatform, $ionicPopup, $cordovaNetwork, $state, $cordovaGeolocation, $interval) {
  $ionicPlatform.ready(function() {
    if ($cordovaNetwork.isOffline()) {
      $ionicLoading.hide();
      $ionicPopup.alert({
        title: 'Peringatan Terputus',
        template: 'Tidak dapat dimuat, Hubungkan dengan Internet'
      })
      .then(function(res) {
        ionic.Platform.exitApp();
      })
    }
  })
  $ionicLoading.show({
    template: '<div class="loader"><svg class="circular"><circle class="path" cx="50" cy="50" r="20" fill="none" stroke-width="2" stroke-miterlimit="10"/></svg></div>'
  });
  $ionicPopup.alert({
    title: '<div class="a"><b>Aplikasi Lacak Kendaraan</b></div> Aplikasi ini untuk melacak kendaraan yang telah dipasang mikrokontroler Arduino dan SIM800. </div>Jangan',
    template: ''
  });
  $scope.dataPosisi = {};
  function getData() {
    var options = {
      enableHighAccuracy: true
    };
    $cordovaGeolocation.getCurrentPosition(function(pos) {
      $scope.latitudeCur = pos.coords.latitude;
      $scope.longitudeCur = pos.coords.longitude;
      console.log('JSON.stringify($scope.latitude)');
      console.log('JSON.stringify($scope.longitude)');
    },
    function(error) {
      alert('Unable to get location: ' + error.message);
    }, options);
    services.getData().success(function(data) {
      $ionicLoading.hide();
      $scope.dataPosisi = data.data.pop();
      $scope.date = new Date($scope.dataPosisi.waktu);
      $scope.date = $filter('date')($scope.date, 'dd MM yyyy HH:mm:ss');
      $scope.$apply();
      console.log(data.data);
    });
  }
  getData();
  $interval(getData(), 1000);
})
})
```

Gambar 3.33 *Controller* Tampilan Posisi Kendaraan

Pada Angular, *Controller* sangat berperan penting dan pada Ionic *Controller* sudah harus ditentukan berkas .html ketika *routing* halaman *Templates*. Data JSON diambil dari nama objek Data, dan menarik data JSON array terakhir.

Data yang digunakan pada *controller* *DataTerbaruController* diambil menggunakan fungsi *get* yaitu *getData*. Setiap detik fungsi *getData* akan melakukan proses pengambilan data array terakhir pada data JSON web server melalui API. Pada berkas yang berada dalam *www/js/services.js* terdapat fungsi Angular *\$http* *get* data JSON dari API yang terdapat pada Gambar 3.30. Setelah mengkonfigurasi *controller* di *DataTerbaruController*, pindah kepada berkas *DataTerbaru.html* untuk membuat *view* pada menu posisi kendaraan terdapat pada Gambar 3.34.

```
<ion-view view-title="Posisi Kendaraan" style="background-color:#E6F9FF;">
<ion-content>
<div>
<ng-map zoom="15" center="[[{dataPosisi.latitude}],[{dataPosisi.longitude}]]">
<custom-marker id="start" position="current-location">
<div>

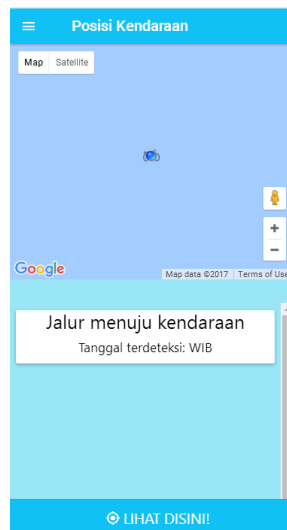
</div>
<marker position="current-location" title="Posisi HP" icon="https://www.robotwoods.com/dev/misc/bluecircle.png"></marker>
</custom-marker>
<custom-marker id="end" position="[[{dataPosisi.latitude}],[{dataPosisi.longitude}]]">
<div>

</div>
<marker position="[[{dataPosisi.latitude}],[{dataPosisi.longitude}]]" title="Posisi HP" icon="img/logken.png"></marker>
</custom-marker>
<directions
suppress-markers="true"
draggable="false"
panel="directions-panel"
travel-mode="DRIVING"
origin="current-location"
destination="[[{dataPosisi.latitude}],[{dataPosisi.longitude}]]">
</directions>
<ion-content style="top:53%;>
<div id="directions-panel" class="list card">
<div align="center">Jalur menuju kendaraan</div>
<div align="center">tanggal terdeteksi: {{date | date:'EEEE', 'dd MMMM yyyy HH:mm:ss'}} WIB</div>
</div>
</ion-content>
</ng-map>
</div>
</ion-content>
<div class="bar bar-footer bar-calm" style="width: 100%; nav-clear menu-close ui-sref="app.DataStreet">
<h1 class="title"><i class="icon ion-android-locate"></i> LIHAT DISINI</h1>
</div>
</ion-view>
```

Gambar 3.34 Kode Program *DataTerbaru.html*

Pembangunan tampilan posisi kendaraan berhubungan dengan *DataTerbaruController* dan *service.js* untuk dapat menggunakan Angular binding, karena Angular data binding dapat diaplikasikan dan dapat dilihat hasilnya apabila kode program sudah terhubung dengan benar, antara *view* dengan *controller* dan antara *controller* dengan *service* API.

Tampilan menu posisi kendaraan menampilkan data JSON *web service* yang telah dibuat sebelumnya dengan menggunakan Google Maps untuk memberikan posisi kendaraan. Setiap terdapat data terbaru pada informasi data posisi yang dikirimkan alat pelacak ke database server, secara otomatis pada tampilan menu posisi kendaraan akan berubah mengikuti data dari database server, seperti pada Gambar 3.35.



Gambar 3.35 Tampilan Posisi Kendaraan

3.3.1.4 Pembuatan Halaman Street View

Pembuatan halaman riwayat posisi memanfaatkan Angular data *Binding* untuk menerapkan JSON dari API kedalam tampilan HTML yang berada pada *www/templates/DataStreet.html* dan menggunakan *library* ngMap untuk menampilkan data terbaru kedalam peta dari *Street View Google* Maps dengan memanfaatkan data latitude dan longitude yang berada di dalam JSON.

Tampilan halaman posisi kendaraan akan dibangun berdasarkan rancangan pada Gambar 3.8 dan akan muncul ketika tombol ‘Lihat Disini’ ditekan pada menu posisi kendaraan, terdapat pada Gambar 3.34. Sebelum membuat tampilan pada berkas *DataStreet.html* terlebih dahulu membuat *controllers* untuk berkas html tersebut.

```
.controller('DataStreetController', function($scope, services, $ionicLoading, $ionicPlatform, $ionicPopup, $cordovaNetwork, $state, $cordovaGeolocation) {
    $ionicPopup.alert({
        title: '<div class="a"><b>Peringatan!</b></div> Posisi ini merupakan posisi terakhir yang terdeteksi. Silahkan cari disekitar atau berkomunikasi',
        template: ''
    });
    $scope.dataPosisi = {};
    function getData() {
        services.getData().success(function(data) {
            $ionicLoading.hide();
            $scope.dataPosisi = data.data.pop();
            $scope.$apply();
            console.log(data.data);
        });
    }
    getData();
    $interval(getData(), 1000);
})
```

Gambar 3.36 Controller Tampilan Street View

Pada Angular, *Controller* sangat berperan penting dan pada Ionic *Controller* sudah harus ditentukan berkas .html ketika *routing* halaman *Templates*. Data JSON diambil dari nama objek Data dan menarik data JSON array terakhir.

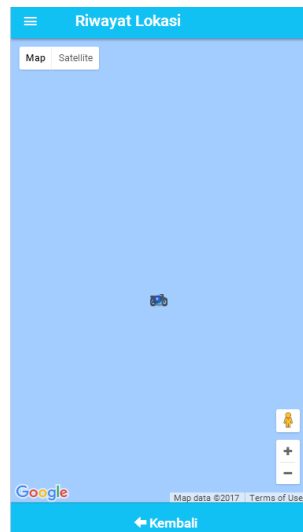
Data yang digunakan pada *controller* *DataStreetController* diambil menggunakan fungsi *get* yaitu *getData*. Setiap detik fungsi *getData* akan melakukan proses pengambilan data array terakhir pada data JSON web server melalui API. Pada berkas yang berada dalam *www/js/services.js* terdapat fungsi Angular *\$http* *get* data JSON dari API yang terdapat pada Gambar 3.30. Setelah mengkonfigurasi *controller* di *DataStreetController*, pindah kepada berkas *DataStreet.html* untuk membuat *view* pada menu posisi kendaraan terdapat pada Gambar 3.37.

```
<ion-view view-title="Kendaraan anda di sekitar sini!" style="background-color:#98E6F6;">
  <ion-content>
    <div style="width:100%; min-height: 100%;" id="street-view">
      <ng-map style="width: 100%; height: 93%;" street-view="StreetViewPanorama(document.querySelector('#street-view'), {position:new
        google.maps.LatLng({{dataPosisi.latitude}}, {{dataPosisi.longitude}})}))">
      </ng-map>
    </div>
    <div class="bar bar-footer bar-calm" nav-clear menu-close ui-sref="app.DataTerbaru">
      <h1 class="title"><i class="icon ion-arrow-left-a" ></i> Kembali</h1>
    </div>
  </ion-content>
</ion-view>
```

Gambar 3.37 Kode Program *DataStreet.html*

Pembangunan tampilan *street view* berhubungan dengan *DataStreetController* dan *service.js* untuk dapat menggunakan Angular binding, karena Angular data binding dapat diaplikasikan dan dapat dilihat hasilnya apabila kode program sudah terhubung dengan benar, antara *view* dengan *controller* dan antara *controller* dengan *service* API.

Tampilan menu *street view* menampilkan data JSON *web service* yang telah dibuat sebelumnya dengan menggunakan Google Maps untuk memberikan posisi kendaraan dengan mode *street view*. Setiap terdapat data terbaru pada informasi data posisi yang dikirimkan alat pelacak ke database server, secara otomatis pada tampilan menu *street view* akan berubah mengikuti data dari database server, seperti pada Gambar 3.38.

Gambar 3.38 Tampilan *Street View*

3.3.1.5 Pembuatan Halaman Riwayat Posisi

Pembuatan halaman riwayat posisi memanfaatkan Angular data *Binding* untuk menerapkan JSON dari API kedalam tampilan HTML yang berada pada *www/templates/DataRiwayat.html* dan menampilkan data-data riwayat posisi yang telah dilewati kendaraan kedalam *panel card*.

Tampilan halaman riwayat posisi akan dibangun berdasarkan rancangan pada Gambar 3.9 dan akan muncul ketika menu riwayat posisi dipilih pada aplikasi. Sebelum membuat tampilan pada berkas *DataRiwayat.html* terlebih dahulu membuat *controllers* untuk berkas html tersebut, terdapat pada Gambar 3.39.

```
.controller('DataRiwayatController', function($scope, services, $ionicLoading, $ionicPlatform, $ionicPopup, $cordovaNetwork, $state, $cordovaGeolocation) {
    var options = {
        enableHighAccuracy: true
    };
    $cordovaGeolocation.getCurrentPosition(function(pos) {
        $scope.latitudeCur = pos.coords.latitude;
        $scope.longitudeCur = pos.coords.longitude;
        console.log(JSON.stringify($scope.latitude));
        console.log(JSON.stringify($scope.longitude));
    },
    function(error) {
        alert('Unable to get location: ' + error.message);
    }, options);
    $scope.dataPosisi = [];

    function getData() {
        services.getData().success(function(data) {
            $ionicLoading.hide();
            $scope.dataPosisi = data.data;
            $scope.data10 = $scope.dataPosisi[$scope.dataPosisi.length - 1];
            $scope.data9 = $scope.dataPosisi[$scope.dataPosisi.length - 2];
            $scope.data8 = $scope.dataPosisi[$scope.dataPosisi.length - 3];
            $scope.data7 = $scope.dataPosisi[$scope.dataPosisi.length - 4];
            $scope.data6 = $scope.dataPosisi[$scope.dataPosisi.length - 5];
            $scope.data5 = $scope.dataPosisi[$scope.dataPosisi.length - 6];
            $scope.data4 = $scope.dataPosisi[$scope.dataPosisi.length - 7];
            $scope.data3 = $scope.dataPosisi[$scope.dataPosisi.length - 8];
            $scope.data2 = $scope.dataPosisi[$scope.dataPosisi.length - 9];
            $scope.data1 = $scope.dataPosisi[$scope.dataPosisi.length - 10];
            $scope.$apply();
            console.log(data.data);
        });
    }
    getData();
    $interval(getData(), 1000);
});
```

Gambar 3.39 *Controller* Tampilan Riwayat Posisi

Pada Angular, *Controller* sangat berperan penting dan pada Ionic *Controller* sudah harus ditentukan berkas .html ketika *routing* halaman *Templates*. Data JSON diambil dari nama objek Data dan menarik data JSON array terakhir.

Data yang digunakan pada *controller* DataRiwayatController diambil menggunakan fungsi *get* yaitu *getData*. Setiap detik fungsi *getData* akan melakukan proses 10 data array terakhir pada data JSON web server melalui API. Pada berkas yang berada dalam *www/js/services.js* terdapat fungsi Angular *\$http* *get* data JSON dari API yang terdapat pada Gambar 3.30. Setelah mengkonfigurasi *controller* di DataRiwayatController, pindah kepada berkas *DataRiwayat.html* untuk membuat *view* pada menu posisi kendaraan terdapat pada Gambar 3.40.

```

<ion-view view-title="Riwayat Lokasi" style="background-color:#000000">
  <ion-content>
    <ion-list>
      <div class="card animated jello">
        <div class="item">
          <div align="center"><i>Riwayat Lokasi</i></div>
        </div>
      </div>
      <div class="card animated jello">
        <div class="item">
          <div class="item">
            Informasi ke {{data0.id}}
          </div>
          <div class="item item-body">
            <div>Tanggal Terdeteksi : {{data0.waktu}} WIB </div>
            <div></div>
            <div>Titik Koordinat
              <div>{{data0.latitude}}</div>
              <div>{{data0.longitude}}</div>
            </div>
          </div>
        </div>
      </div>
      <div class="card animated jello">
        <div class="item">
          <div class="item">
            Informasi ke {{data1.id}}
          </div>
          <div class="item item-body">
            <div>Tanggal Terdeteksi : {{data1.waktu}} WIB </div>
            <div></div>
            <div>Titik Koordinat
              <div>{{data1.latitude}}</div>
              <div>{{data1.longitude}}</div>
            </div>
          </div>
        </div>
      </div>
      <div class="card animated jello">
        <div class="item">
          <div class="item">
            Informasi ke {{data2.id}}
          </div>
          <div class="item item-body">
            <div>Tanggal Terdeteksi : {{data2.waktu}} WIB </div>
            <div></div>
            <div>Titik Koordinat
              <div>{{data2.latitude}}</div>
              <div>{{data2.longitude}}</div>
            </div>
          </div>
        </div>
      </div>
    </ion-list>
  </ion-content>
</ion-view>

```

Gambar 3.40 Penggalan Kode Program *DataRiwayat.html*

Pembangunan tampilan *street view* berhubungan dengan DataRiwayatController dan *service.js* untuk dapat menggunakan Angular binding, karena Angular data binding dapat diaplikasikan dan dapat dilihat hasilnya apabila kode program sudah terhubung dengan benar, antara *view* dengan *controller* dan antara *controller* dengan API *web service*.

Tampilan menu *street view* menampilkan data JSON *web service* yang telah dibuat sebelumnya dengan menampilkan data-data riwayat posisi yang telah dilewati kendaraan kedalam *panel card*. Setiap terdapat data terbaru pada informasi data posisi yang dikirimkan alat pelacak ke database server, secara otomatis pada

tampilan menu riwayat posisi akan berubah mengikuti data dari database server, seperti pada Gambar 3.41.



Gambar 3.41 Tampilan Riwayat Posisi

3.3.1.6 Pembuatan Halaman Antisipasi Kehilangan

Pada tampilan antisipasi kehilangan akan memberikan informasi mengenai hal-hal yang dibutuhkan sebelum terjadi kehilangan kendaraan dan hal-hal apa saja yang harus dilakukan setelah terjadi kehilangan. Tampilan antisipasi kehilangan dibuat sedikit lebih menarik dengan adanya gambar yang menggambarkan setiap nilai-nilai yang terkandung pada teks yang dijelaskan pada informasi antisipasi kehilangan.

Tampilan ini tidak memiliki aktifitas yang harus dilakukan oleh *controller* pada Angular, akan tetapi harus mencantumkan nama *controller* pada *route* dan harus menuliskan *.controller* dengan nama *controller* yang harus sama pada *route* dan memberikan fungsi *scope*, karena *scope* adalah *super variable* untuk menjalankan *binding* atau *viewcontroller*. AntisipasiController dapat dilihat pada Gambar 3.42.

```
app.controller('AntisipasiController', function($scope, $stateParams) {
});
```

Gambar 3.42 *Controller* Tampilan Antisipasi Kehilangan

Tampilan antisipasi kehilangan hanya menampilkan teks dan gambar agar lebih menarik sebagai informasi untuk pengguna. Pada berkas *AntisipasiKehilangan.html* kode program lebih panjang karena seluruh informasi teks atau gambar ditaruh pada *tag* HTML pada berkas tersebut. Pada berkas gambar untuk tampilan antisipasi kehilangan terdapat pada *path* *img/antisipasi/sebelum/* dan *img/antisipasi/sesudah/*.

Pada *view* untuk antisipasi kehilangan berbeda dengan tampilan sebelum nya, karna tidak sama sekali memanfaatkan fungsi-fungsi pada Angular. Seluruh nya menggunakan *tag* HTML yang sudah disediakan Ionic dan *tag* HTML dasar dalam membangun *website*, dapat dilihat pada Gambar 3.43.

```
<ion-view view-title="Antisipasi Kehilangan" style="background-color:#E6F9FF;">
  <ion-content>
    <div class="card animated rubberBand">
      <div class="item item-text-wrap">
        <h3 class="has-subheader" style="color: #000; ">SEBELUM dan SESUDAH terjadi Kehilangan</h3>
      </div>
    </div>

    <div class="card">
      <div class="item item-divider">
        Sebelum Terjadi Kehilangan
      </div>
      <div class="list-card">
        <ion-list>
          <ion-item>
            <div class="item">
              <h2>A. Pasang Alat Pelacak ini</h2>
            </div>
            <div class="item item-body">
              
              <p>
                Taruh alat pelacak ini di kendaraan anda pada bagian tempat yang aman dah memungkinkan. Jangan lupa untuk menyalakan terlebih dahulu dengan menekan tombol dan colokan dengan baterai 9 volt. Jika baterai habis, segera ganti dengan baterai baru.
              </p>
            </div>
          </ion-item>
        </ion-list>
      </div>
    </div>
  </ion-content>
</ion-view>
```

Gambar 3.43 Penggalan Kode Program *AntisipasiKehilangan.html*

Pada penggalan kode program tampilan antisipasi kehilangan menggunakan nama *class* yang sudah disediakan oleh *documentation* Ionic di *website* resmi Ionic. Membuat header menggunakan *tag* HTML dasar seperti *h1* sampai *h6* tergantung tingkat besar *value font*, untuk membuat paragraf menggunakan *tag p*, untuk menampilkan gambar menggunakan *tag img* dan harus menentukan letak berkas gambar tersebut pada atribut *src* dan tampilan Antisipasi Kehilangan dapat dilihat pada Gambar 3.44.



Gambar 3.44 Tampilan Antisipasi Kehilangan

3.3.1.7 Pembuatan Halaman Tentang

Tampilan tentang berisi informasi mengenai *library* yang digunakan dalam pembangunan aplikasi. Pada pembuatan tampilan tentang, tidak perlu mengisikan kode pada *controller* karena tidak ada aktifitas apapun didalam tampilan tentang. Penulisan kode program menggunakan *class card item-icon-left* agar dapat menginformasikan *library* beserta *icon* dari *library* tersebut.

Pembuatan tampilan halaman tentang mengikuti rancangan yang sudah dibuat pada Gambar 3.12, tampilan dibuat dari bahasa HTML dan CS yang tidak berbeda dengan tampilan lainnya dan terdapat animasi yang menggunakan *library animate.css*. Pada TentangController tidak berisi kode apapun dapat dilihat pada Gambar 3.45.

```
app.controller('TentangController', function($scope, $stateParams) {
});
```

Gambar 3.45 Controller Halaman Tentang

Pada kode program controller halaman tentang, terdapat pemberian *variable app* yang mendefinisikan *module* pada *app.js*. Selanjutnya mengenai kode pada *view* tentang menggunakan *class im-wrapper* yang berfungsi memberikan padding

pada setiap sisi agar konten didalamnya menjadi lebih mendalam. Penggalan kode program dapat dilihat pada Gambar 3.46.

```
<ion-view view-title="Tentang" style="background-color:#98E6F6;">
  <ion-content class="ion-wrapper">
    <div class="card stable-bg animated wobble">
      <div class="item">
        <h2>Dibuat Oleh</h2>
      </div>
    </div>
    <div class="card animated bounceIn">
      <a class="item item-icon-left">
        <i class="icon ion-ionic"></i>
        Dicka Ariptian R
      </a>
    </div>
    <div class="a"></div>
    <div class="card stable-bg animated wobble">
      <div class="item">
        <h2>Terbuat Oleh</h2>
      </div>
    </div>
    <div class="card animated bounceIn">
      <a class="item item-icon-left">
        <i class="icon ion-ionic"></i>
        Ionic 1.3
      </a>
    </div>
    <div class="card animated bounceIn">
      <a class="item item-icon-left">
        <i class="icon ion-social-html5"></i>
        HTML 5
      </a>
    </div>
  </div>
</ion-view>
```

Gambar 3.46 Penggalan Kode Program *Tentang.html*

Pada penggalan kode program tampilan tentang, terdapat 10 *card stable-bg*. Fungsi *card* sudah terdapat pada *class* Ionic dan mendapati *design* lebih menarik karena *library* ionic-material. Terdapat pula didalam *class* tersebut *animated wobble*, itu merupakan nama *class* pada *library animate.css* yang memberikan animasi pada *class card* tersebut.

Halaman ini merupakan tampilan terakhir pada aplikasi informasi posisi kendaraan. Pembangunan yang terakhir pada aplikasi informasi posisi kendaraan adalah membuat aksi menutup aplikasi menggunakan menu kembali dan tombol kembali pada *smartphone* pengguna. Pembahasan tersebut terdapat pada subab selanjutnya.

3.3.1.8 Pembuatan Tombol Keluar

Pada *smartphone* pengguna terdapat tombol kembali yang berfungsi untuk kembali pada halaman sebelumnya dan menutup aplikasi. Pada pembangunan aplikasi lokasi kendaraan ini Angular berperan penting untuk aktifitas ini, terdapat suatu metode utilitas yang dapat digunakan untuk mengambil informasi perangkat dengan perintah *\$ionicPlatform*.

Penulisan kode program untuk fungsi menutup aplikasi harus diketikan pada *app.run function* pada *app.js*, hal tersebut pendefinisian *state* fungsi yang dijalankan berdasarkan *module* Angular di Ionic. Penulisan kode program dapat dilihat pada Gambar 3.47.

```
$ionicPlatform.registerBackButtonAction(function(event) {
  if (true) { |
    $ionicPopup.confirm({
      title: 'Keluar',
      template: 'Anda yakin ingin menutup aplikasi ini ?'
    }).then(function(res) {
      if (res) {
        ionic.Platform.exitApp();
      }
    })
  }
}, 100);
```

Gambar 3.47 Kode Program *BackButtonAction*

Pada penulisan kode program di Gambar 3.33 menggunakan metode *registerBackButtonAction* yang berfungsi agar aplikasi Ionic dapat mengetahui aksi tombol kembali pada *smartphone* pengguna. Metode tersebut memiliki percabangan yang berfungsi apabila pengguna menekan tombol kembali maka aplikasi akan mengeluarkan *Popup* konfirmasi untuk menutup aplikasi.

Pada tampilan *sidemenu* terdapat menu keluar, berfungsi sama seperti *BackButtonAction* pada *smartphone* untuk menutup aplikasi. Menu keluar terdapat pada *sidemenu* atau *menu.html* yang memiliki *controller* dengan nama *MenuController*. Terdapat *div id* pada *menu.html*, lalu Angular mengambil *id* dan menunggu kejadian klik pada menu keluar. Apabila terjadi aktifitas klik pada menu keluar, aplikasi akan mengeluarkan *Popup* konfirmasi untuk menutup aplikasi. Kode program dapat dilihat pada Gambar 3.48.

```

app.controller('MenuController', function ($scope, $ionicPopup) {
  var fab = document.getElementById('fab');
  fab.addEventListener('click', function () {
    $scope.showConfirm = function() {
      var confirmPopup = $ionicPopup.confirm({
        title: 'Keluar',
        template: 'Anda yakin ingin menutup aplikasi ini ?'
      });
      confirmPopup.then(function(res) {
        if(res) {
          // console.log('You are sure');
          ionic.Platform.exitApp();
        } else {
          console.log('You are not sure');
        }
      });
    };
  });
});

```

Gambar 3.48 *Controller Tampilan Menu*

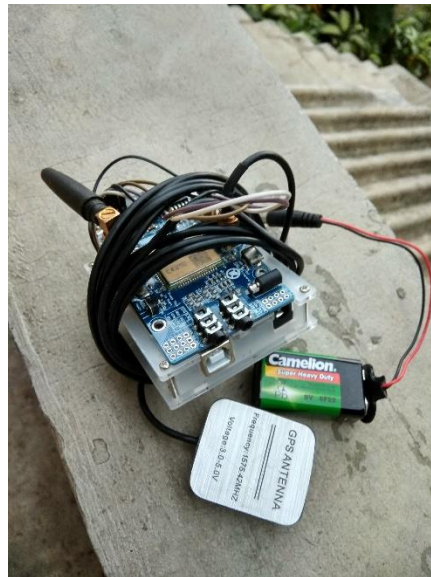
Pada Gambar 3.31 mengambil elemen *id* pada HTML dengan nama *fab*, lalu pada id *fab* ditambahkan pendeteksi klik dan didalam nya terdapat fungsi *showConfirm*. Fungsi tersebut menggunakan fungsi *ionicPopup* untuk menampilkan *alert* pilihan. Terdapat kondisi apabila pengguna memilih OK pada *alert* maka aplikasi akan tertutup, lalu apabila pengguna memilih *Cancel* maka aplikasi tidak tertutup.

3.3.2 Pembuatan Alat Pelacak

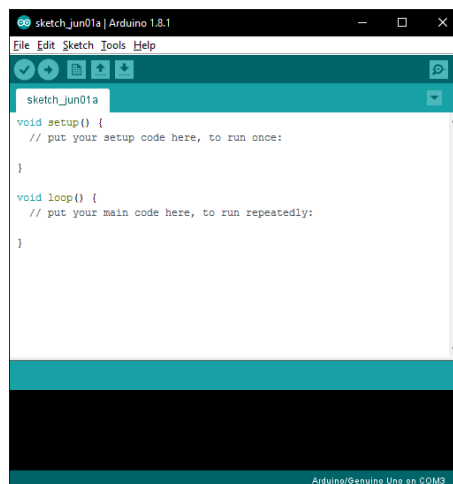
Pembuatan alat pelacak menggunakan mikrokontroller arduino uno dan SIM 808 untuk menerima data posisi dari satelit serta mengirim data tersebut ke *database server*. Pada pembuatan ini program ditanamkan pada arduino untuk mengatur komponen secara keseluruhan. Program menggunakan *library* dari TinyGPS dan TinyGSM untuk memudahkan kode proses pada arduino.

3.3.2.1 Persiapan Alat Pelacak

Pada pembuatan alat pelacak, komponen-komponen dirangkai sesuai dengan gambar 3.1. Setelah rangkaian dibuat, tanamkan program pada arduino uno dari PC (*Personal Computer*) menggunakan USB TL. Untuk menanamkan program, yang harus dilakukan adalah menginstal Arduino IDE terlebih dahulu yang dapat diunduh dari <https://www.arduino.cc/en/Main/Software>.



Gambar 3.49 Alat Pelacak



Gambar 3.50 Arduino IDE

3.3.2.2 Kode Program Alat Pelacak

Setelah menginstall Arduino IDE, unduh *library* TinyGPS dan TinyGSM di <http://arduiniiana.org/libraries/> untuk digunakan dalam program. Pada program yang akan digunakan, deklarasikan *library* dan variabel pada baris pertama atau blok awal kode program. Sehingga dapat menggunakan kode yang dapat memudahkan pada proses Arduino IDE.

```

#include <TinyGsmClient.h>
#include <TinyGPS++.h>
#include <SoftwareSerial.h>
SoftwareSerial SerialAT(7,8); // RX, TX
TinyGPSPlus gps;
TinyGsm modem(SerialAT);
TinyGsmClient client(modem);
const char apn[] = "internet";
const char user[] = "";
const char pass[] = "";
const char server[] = "dickaariptian.xyz";
const char resource[] = "/api/addDb.php?";
int port = 80;
double longg, latt, spd;

```

Gambar 3.51 Penggalan Kode Blok Awal

Pada Gambar 3.51 menjelaskan bahwa ada 3 *library* yang digunakan, yaitu TinyGsmClient, TinyGPS++ dan SoftwareSerial. *Library* TinyGsmClient untuk memudahkan menulis kode program pada bagian proses GSM Modul. *Library* TinyGPS++ untuk memudahkan menulis kode program pada bagian GPS Modul. *Library SoftwareSerial* digunakan untuk jalur perpindahan data dari komponen SIM 808 dan arduino uno melalui pin 7 dan 8.

Pada blok awal variabel-variabel dideklarasikan sesuai dengan kebutuhan. SoftwareSerial dideklarasikan dengan nama variabel SerialAT dengan pin 7 dan 8. TinyGPSPlus dideklarasikan dengan variabel gps. TinyGsm dideklarasikan dengan nama variabel modem dan termasuk fungsi dari variabel SerialAT. TinyGsmClient dideklarasikan dengan nama variabel client dan termasuk fungsi dari variabel modem. Pada variabel apn, server dan resource dideklarasikan dengan tipe *const char*. Variabel apn diisi dengan ‘internet’ untuk pengaturan APN pada jaringan GSM. Variabel server diisi dengan ‘tesarduino.esy.es’ untuk pengaturan server yang dituju. Variabel resource diisi dengan ‘/addDb.php?’ untuk pengaturan berkas pada server yang dituju. Variabel port bertipe integer untuk pengaturan port pada server. Variabel longg dan latt bertipe double untuk diisi dengan data lokasi longitude dan latitude.

Pada kode program arduino umumnya terdiri dari 2 intruksi, yaitu instruksi *void setup()* dan *void loop()*. Instruksi *void setup()* digunakan untuk menginisialisasi variabel-variabel yang akan digunakan dan hanya dijalankan satu kali saat Arduino mulai menyala. Sedangkan instruksi *void loop()* digunakan untuk menjalankan suatu siklus program, yang akan dilakukan terus-menerus hingga Arduino mati/reset.

```
void setup() {
  Serial.begin(9600);
  delay(1000);
  SerialAT.begin(9600);
  delay(3000);
  modem.init();
  delay(30000);
  SerialAT.println("AT+CGNSPWR=1");
  SerialAT.println("AT+CGNSTST=1");
  delay(120000);
}
```

Gambar 3.52 Penggalan Kode *void setup()*

Pada gambar 3.52 menjelaskan tentang intruksi *void setup()* atau pengaturan awal saat alat pelacak dihidupkan dan diberi daya listrik oleh baterai. Setelah dihidupkan, pengaturan awal yang dilakukan yaitu dimulai pada *Serial* dengan *baud rate* 9600 bps untuk kecepatan aliran data. Kemudian ada jeda 1000 *miliseconds* atau 1 detik dan dilanjutkan dengan variabel *SerialAT* dengan *baud rate* 9600 bps. Setelah pengaturan pada *SerialAT*, jeda 3000 *miliseconds* atau 3 detik.

Kemudian dilanjutkan untuk inisialisasi variabel modem serta ada jeda setelahnya 30 detik. Setelah variable modem telah melakukan inisialisasi, *SerialAT* memberikan perintah AT+CGNSPWR untuk menyalakan GPS Modul dan perintah AT+CGNSTST menerima data posisi terkini dari satelit serta diberi jeda 120 detik. Setelah 120 detik, alat pelacak menjalankan blok program *void loop()* .

```

void loop() {
    longg = gps.location.lng();
    latt = gps.location.lat();
    SerialAT.println("AT+CGNSTST=0");
    if (!modem.waitForNetwork()) {
        Serial.println(" fail");
        delay(10000);
        return;
    }
    if (!modem.gprsConnect(apn, user, pass)) {
        Serial.println(" fail");
        delay(10000);
        return;
    } ....

    client.print(String("POST ") + resource + "latitude=" + String(latt,10) + "&longitude=" +
String(longg,10) + " HTTP/1.0\r\n");

    client.print(String("Host: ") + server + "\r\n");
    SerialAT.println("AT+CGNSTST=1"); ...
}

```

Gambar 3.53 Penggalan Kode *void loop()*

Pada gambar 3.53 menjelaskan tentang intruksi *void loop()* untuk menjalankan siklus program secara berulang terus menerus. Tahapan yang dilakukan pertama kali adalah memberikan nilai pada variabel *longg* dan *latt* dengan fungsi dari *library* GPS untuk mendapatkan data lokasi longitude dan latitude. Setelah mendapatkan nilai latitude dan longitude, modul GPS dimatikan agar modul GSM/GPRS dapat bekerja. Selanjutnya memeriksa jaringan yang tersedia pada *sim card*. Setelah jaringan tersedia, akan dihubungkan dengan variabel *apn*, *user* dan *pass* yang telah dideklarasikan sebelumnya.

Tahapan setelah jaringan tersedia dan terhubung adalah mengirimkan data lokasi longitude dan latitude melalui protokol HTTP dengan metode POST. Pada pengiriman ini data yang diterima dari satelit diubah ke dalam bentuk *string* sehingga dapat dilakukan menggunakan protokol HTTP. Kemudian modul GPS dinyalakan kembali untuk mendapatkan nilai latitude dan longitude terbaru.

3.3.3 Pembuatan RESTful Web Services

Pembuatan RESTful *Web Services* untuk penyimpanan data dan melakukan pertukaran data antar aplikasi atau sistem. Pada pembuatan ini menggunakan PHP native untuk mengolah seluruh proses yang dikerjakan dan MySQL untuk *database server* atau penyimpanan data. Penyedia layanan hosting yang digunakan yaitu IdWebHost. Penyedia ini menyediakan domain, *bandwith* dan *space* memori yang cukup sesuai dengan kebutuhan. Domain yang digunakan adalah <http://www.dickaariptian.xyz>

Pada gambar 3.3 telah dibuat struktur *table* database yang akan digunakan. Untuk melakukan proses penyimpanan data dan mengolah data dibutuhkan kode program. Kode program berhubungan dengan *host*, *username*, *password* dan *database* yang digunakan pada penyedia layanan IdWebHost.

```
<?php
$servername = "localhost";
$username = "dickaari_user";
$password = "*****";
$my_db = "dickaari_db";

// Create connection
$dbh = mysqli_connect($servername, $username, $password,$my_db);
$selected = mysqli_select_db($dbh, "dickaari_db");

// Check connection
if (!$dbh) {
    die("Connection failed: " . mysqli_connect_error());
}

?>
```

Gambar 3.54 Kode Program *dbcon.php*

Pada gambar 3.54 merupakan kode program PHP untuk menghubungkan *database* sesuai dengan identitas di server. Kode program ini termasuk ke dalam 1 berkas dan dapat digunakan sebagai *include* pada berkas lainnya.

```
<?php
include("dbcon.php");

$lat = $_GET['latitude'];
$lng = $_GET['longitude'];

$SQL = "INSERT INTO GMAP (latitude, longitude) VALUES ('$lat','$lng')";
mysqli_query($dbh, $SQL);

?>
```

Gambar 3.55 Kode Program *addDb.php*

Pada gambar 3.55 merupakan kode program PHP untuk menyimpan data ke *database server*. Data yang diterima dari alat pelacak melalui protokol HTTP dengan metode GET dari params latitude dan longitude pada URL HTTP serta data disimpan pada tabel GMAP di *database server*.

```
<?php
include("dbcon.php");

$SQL = "SELECT * FROM GMAP";

$query = mysqli_query($dbh, $SQL);
while($dt=mysqli_fetch_array($query)){
    $data[] = array(
        "id"=>$dt["id"],
        "waktu"=>$dt["waktu"],
        "latitude"=>$dt["latitude"],
        "longitude"=>$dt["longitude"]
    );
}

$json = array(
    'result' => 'success',
    'data' => $data
);

echo json_encode($json);
?>
```

Gambar 3.56 Kode Program *getmap.php*

Pada gambar 3.56 merupakan kode program PHP untuk mengubah data pada tabel GMAP ke dalam bentuk array dan JSON. Data-data ini yang akan ditarik oleh aplikasi untuk ditampilkan menjadi informasi pada *smartphone* pengguna.

```
{"result":"success","data":null}
```

Gambar 3.57 Data JSON pada *Web Services*

Pada gambar 3.57 merupakan umpan balik dari web server ketika meminta data JSON yang tersimpan pada *database server*. Data-data ini merupakan API *Web Services* yang akan ditampilkan pada aplikasi informasi posisi kendaraan.