TUGAS AKHIR- TE 141599

INTELLIGENT MARITIME TRANSPORTATION SYSTEM: VISUALISASI DATA KAPAL BERBASIS AIS MENGGUNAKAN PETA DARING

Shidqon Famulaqih
NRP 2211100019

Dosen Pembimbing
Dr. I Ketut Eddy Purnama, ST., MT.
Christyowidiasmoro, ST., MT.

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh November
Surabaya 2016

TUGAS AKHIR- TE 141599

INTELLIGENT MARITIME TRANSPORTATION SYSTEM: AIS BASED VESSEL DATA VISUALIZATION USING ONLINE MAP

Shidqon Famulaqih
NRP 2211100019

Advisor
Dr. I Ketut Eddy Purnama, ST., MT.
Christyowidiasmoro, ST., MT.

Department of Electrical Engineering
Faculty of Industrial Technology
Institut Teknologi Sepuluh November
Surabaya 2016

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "***Intelligent Maritime Transportation System:* Visualisasi Data Kapal Berbasis AIS Menggunakan Peta Daring**" adalah benar-benar hasil karya intelektual mandiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan karya pihak lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun dirujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai peraturan yang berlaku.

Surabaya, 25 Januari 2016

Shidqon Famulaqih
NRP. 2211100019

Halaman ini sengaja dikosongkan

INTELLIGENT MARITIME TRANSPORTATION SYSTEM: VISUALISASI DATA KAPAL BERBASIS AIS MENGGUNAKAN PETA DARING

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Komputer dan Telematika
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui:

Dosen Pembimbing I	Dosen Pembimbing II
Dr. I Ketut Eddy Purnama, ST., MT.	Christyowidiasmoro,ST., MT.
NIP. 196907301995121001	NIP. 198301272009121004

SURABAYA
JANUARI, 2016

Halaman ini sengaja dikosongkan

ABSTRAK

Nama Mahasiswa	:	Shidqon Famulaqih
Judul Tugas Akhir	:	Intelligent Maritime Transportation System: Visualisasi Data Kapal Berbasis AIS Menggunakan Peta Daring
Pembimbing	:	1. Dr. I Ketut Eddy Purnama, S.T., M.T. 2. Christyowidiasmoro, S.T., M.T.

Penelitian *Intelligent Maritime Transportation System* (IMTS) dilakukan untuk mengatasi masalah kemaritiman di Indonesia. Diantaranya adalah pencurian ikan oleh kapal yang tidak terdeteksi dan tidak adanya informasi publik terkait kapal yang sedang beroperasi di Indonesia. Salah satu fungsi IMTS adalah sebagai *information provider* yang menyediakan segala informasi yang dibutuhkan oleh pelaku kemaritiman seperti pihak pelabuhan, perusahaan perkapalan, perusahaan ekspedisi, dan *Search and Rescue* (SAR). Informasi yang diberikan berupa informasi kapal dari AIS yang didapatkan melalui sistem penerima data AIS dan hubunganya dengan pelabuhan yang ada. Karena itu diperlukan modul visualisasi untuk IMTS agar bisa menerima dan menyimpan data dari sistem penerima data AIS dan menampilkannya agar bisa diakses oleh semua pelaku kemaritiman. Tugas akhir ini dilakukan untuk membuat aplikasi *database* berbasis *website* untuk melakukan visualisasi data dari IMTS menggunakan peta daring.

Kata kunci: Visualisasi Data, AIS (*Automatic Identification System*), GIS (*Geographical Information System*), Google Maps API

Halaman ini sengaja dikosongkan

ABSTRACT

Name	:	Shidqon Famulaqih
Title	:	Intelligent Maritime Transportation System: AIS Based Vessel Data Visualization Using Online Map
Advisor	:	1. Dr. I Ketut Eddy Purnama, S.T., M.T. 2. Christyowidiasmoro, S.T., M.T.

Research on Intelligent Maritime Transportation System (IMTS) aims to solve marritime issues in Indonesia. Some of those issues are illegal fishing by vessels that are not detected and the absence of public information related to the ship currently operating in Indonesia. One of the functions of the IMTS is as an information provider that provides all the information required by maritime stakeholder such as the ports, shipping companies, and Search and Rescue (SAR). The information provided is AIS vessel information obtained through the AIS data receiving system and its relations with the existing port. Therefore it is necessary to create visualization module for IMTS in order to receive and store data from the receiving system and display AIS data to be accessed by all maritime stakeholder. The final task is done to create a web-based database applications to visualize data from IMTS using an online map.

Keywords: Data visualization, AIS (Automatic Identification System), GIS (Geographical Information System), Google Maps API

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan Syukur kehadirat Allah SWT atas segala limpahan berkah, rahmat serta hidayah-Nya, penulis dapat menyelesaikan penelitian ini dengan judul : ***Intelligent Maritime Transportation System:*** **Visualisasi Data Kapal Berbasis AIS Menggunakan Peta Daring**.

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Jurusan Teknik Elektro ITS, Bidang Studi Teknik Komputer dan Telematika, serta digunakan sebagai persyaratan menyelesaikan pendidikan S-1. Penelitian ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu, danBapak yang telah memberikan dorongan spiritual dan material serta seluruh kerabat dan kolega penulis yang banyak membantu proses dalam menyelesaikan buku penelitian ini.
2. Bapak Dr. Ardyono Priyadi, ST., M.Eng. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember
3. Secara khusus penulis mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Dr.I Ketut Eddy Purnama, ST., MT. dan Bapak Christyowidiasmoro, ST., MT. atas bimbingan selama mengerjakan penelitian.
4. Bapak-ibu dosen pengajar Bidang Studi Teknik Komputer dan Telematika, atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
5. Seluruh teman-teman *B201-crew* Laboratorium Bidang Studi Teknik Komputer dan Telematika.

Kesempurnaan hanya milik Allah SWT, untuk itu penulis memohon segenap kritik dan saran yang membangun serta menghatur maaf atas segala kekurangan yang ada dalam penulisan buku ini. Semoga penelitian ini dapat memberikan manfaat bagi kita semua.Amin.

Surabaya, Desember 2015

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

Halaman ini sengaja dikosongkan

DAFTAR KODE

BAB I
PENDAHULUAN

Penelitian ini dilatar belakangi oleh berbagai kondisi yang menjadi acuan. Selain itu juga terdapat beberapa permasalahan yang akan dijawab sebagai luaran dari penelitian ini.

1.1 Latar Belakang

Penelitian *Intelligent Maritime Transportation System* (IMTS) dilakukan untuk mengatasi masalah kemaritiman di Indonesia. Diantaranya adalah pencurian ikan oleh kapal yang tidak terdeteksi dan tidak adanya informasi publik terkait kapal yang sedang beroperasi di Indonesia [1]. IMTS memiliki empat fungsi utama yaitu, *monitoring, controlling, enforcement* dan *information provider*. *Monitoring* adalah pemantauan gerak kapal, baik yang memiliki ijin maupun yang tidak memiliki ijin. *Controlling* adalah untuk mengatur jalur yang dapat dan yang tidak dapat dilalui kapal. *Enforcement* untuk penegakan hukum terkait kapal yang melakukan pelanggaran. Sedangkan *information provider* yaitu sebagai penyedia segala informasi yang dibutuhkan oleh pelaku kemaritiman seperti pihak pelabuhan, perusahaan perkapalan, perusahaan ekspedisi, dan *Search and Rescue* (SAR).

IMTS menggunakan sistem penerima data AIS yang disebar diseluruh kota di area pesisir di Indonesia untuk mendapatkan data kapal. Sebagai *information provider* untuk menampilkan informasi dari data tersebut, diperlukan modul yang dapat menerima dan menyimpan data dari semua sistem penerima dan menampilkannya agar bisa diakses oleh seluruh pelaku kemaritiman. Aplikasi yang umum untuk modul seperti ini adalah aplikasi *database* berbasis *website*.

Data IMTS terdiri dari data kapal dari AIS dan data tentang pelabuhan. Data tersebut ada banyak dan kompleks membuat informasi yang diinginkan sulit dimengerti jika ditampilkan apa adanya. Teknik penampilan data yang tidak sesuai dengan jenisnya juga mengakibatkan tingkat pemahaman pengguna kurang maksimal. Oleh karena itu bagaimana data ditampilkan sesuai jenisnya sangat mempengaruhi pengguna menyerap informasi dari data tersebut. Data kapal dari AIS berisi informasi umum kapal, rute, lokasi, dan navigasi kapal. Sedangkan data pelabuhan yang ditampilkan berisi nama, negara, kode, dan lokasi

pelabuhan [7]. Data-data ini berkaitan dengan lokasi bisa dikategorikan sebagai data spasial. Data spasial biasanya disimpan sebagai koordinat dan topologi, dan merupakan data yang dapat dipetakan. Data spasial sering diakses, dimanipulasi atau dianalisis menggunakan *Geographic Information Systems* (GIS). Salah satu contoh aplikasi dari GIS adalah peta daring Google Maps. Dengan Google Maps API data spasial bisa divisualisasi dengan jelas [2]. Data lokasi bisa ditandai dengan sebuah penanda (*marker*). Rute perjalanan bisa digambarkan sebagai garis vektor berdasarkan perpindahan posisi *marker*.

1.2 Permasalahan

Permasalahan pada tugas akhir ini adalah:

1. Sebagai *information provider* IMTS membutuhkan modul visualisasi yang dapat menerima dan menyimpan data dari semua sistem penerima data AIS dan menampilkannya agar dapat diakses semua pelaku kemaritiman atau pengguna lain yang membutuhkan.
2. Data AIS yang diterima sistem penerima data AIS ada banyak dan komplek. Tidak semua orang bisa mengerti jika data tersebut ditampilkan apa adanya. Untuk itu diperlukan visualisasi dalam bentuk peta daring agar informasi dari data tersebut mudah dimengerti semua orang.

1.3 Tujuan Penelitian

Tujuan dari pembuatan tugas akhir ini adalah membuat modul visualisasi dalam bentuk aplikasi *database* berbasis *website* sebagai bagian dari penelitian IMTS yang dapat menerima dan menyimpan data dari semua sistem penerima untuk menampilkan informasi dari data tersebut dalam bentuk peta daring.

1.4 Batasan Masalah

Batasan masalah pada tugas akhir ini adalah :

1. Data yang divisualisasikan adalah data kapal dari AIS dan data pelabuhan yang sudah dalam bentuk *database*.
2. Data kapal yang divualisasikan hanya sebatas area jangkauan antena sistem penerima data AIS.

3. Visualisasi data dalam bentuk peta dari Google Maps.

1.5 Sistematika Penulisan

Laporan penelitian Tugas akhir ini tersusun dalam sistematika dan terstruktur sehingga lebih mudah dipahami dan dipelajari oleh pembaca maupun seseorang yang hendak melanjutkan penelitian ini. Alur sistematika penulisan laporan penelitian ini yaitu :

1. BAB I Pendahuluan
   Bab ini berisi uraian tentang latar belakang permasalahan, penegasan dan alasan pemilihan judul, tujuan penelitian, metodologi penelitian dan sistematika laporan.

2. BAB II Dasar Teori
   Pada bab ini berisi tentang uraian secara sistematis teori-teori yang berhubungan dengan permasalahan yang dibahas pada penelitian ini. Teori-teori ini digunakan sebagai dasar dalam penelitian, yaitu informasi terkait Speech recognition, Bahasa Isyarat, dan teori-teori penunjang lainya.

3. BAB III Perancangan Sistem dan Implementasi
   Bab ini berisi tentang penjelasan-penjelasan terkait sistem yang akan dibuat. Guna mendukung itu digunakanlah blok diagram agar sistem yang akan dibuat dapat terlihat dan mudah dibaca untuk diimplentasikan pada pembuatan perangkat lunak.

4. BAB IV Pengujian dan Analisis
   Bab ini menjelaskan tentang pengujian yang dilakukan terhadap sistem dalam penelitian ini dan menganalisa sistem. Spesifikasi perangkat keras dan perangkat lunak yang digunakan juga disebutkan dalam bab ini. Sehingga ketika akan dikambangkan lebih jauh, spesifikasi perlengkapannya bisa dipenuhi dengan mudah tanpa harus melakukan ujicoba perangkat lunak maupun perangkat keras lagi.

5. BAB V Penutup
   Bab ini merupakan penutup yang berisi kesimpulan yang diambil dari penelitian dan pengujian yang telah dilakukan. Saran dan kritik yang

membangun untuk pengembangkan lebih lanjut juga dituliskan pada bab ini.

1.6 Relevansi

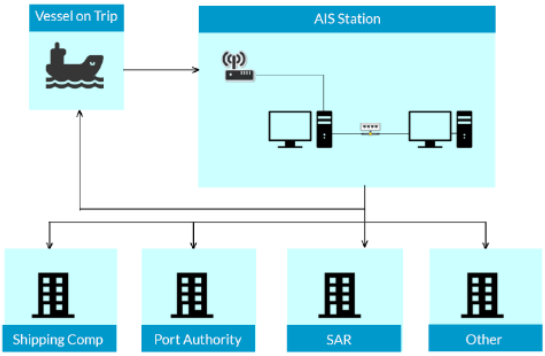
Penelitian mengenai visualisasi data menggunakan peta daring sudah pernah dilakukan sebelumnya [2]. Data yang divisualisasikan dengan dengan peta selalu berkaitan dengan lokasi, data ini bisa dikategorikan sebagai data spasial. Dari penelitian ini dihasilkan sebuah visualisasi data IMTS dalam bentuk peta daring sebagai *web service* untuk mempermudah pengguna menangkap informasi dari data tersebut.

BAB II
DASAR TEORI

Demi mendukung penelitian ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan refrensi. Dengan demikian penelitian ini menjadi lebih terarah.

2.1 *Intelligent Maritime Transportation System* (IMTS)

IMTS dibuat untuk mengatasi masalah kemaritiman di Indonesia. Diantaranya adalah pencurian ikan oleh kapal yang tidak terdeteksi dan tidak adanya informasi publik terkait kapal yang sedang beroperasi di Indonesia. IMTS memiliki empat fungsi utama yaitu, *monitoring, controlling, enforcement* dan *information provider* [1]. *Monitoring* adalah pemantauan gerak kapal, baik yang memiliki ijin maupun yang tidak memiliki ijin. *Controlling* adalah untuk mengatur jalur yang dapat dan yang tidak dapat dilalui kapal. *Enforcement* untuk penegakan hukum terkait kapal yang melakukan pelanggaran. Sedangkan *information provider* yaitu sebagai penyedia segala informasi yang dibutuhkan oleh pelaku kemaritiman seperti pihak pelabuhan, perusahaan perkapalan, perusahaan ekspedisi, dan *Search and Rescue* (SAR) yang diilustrasikan oleh gambar 2.1. Data untuk informasi tersebut diterima menggunakan sistem penerima data AIS yang dipasang di area pesisir seperti ditunjukkan pada gambar 2.2.

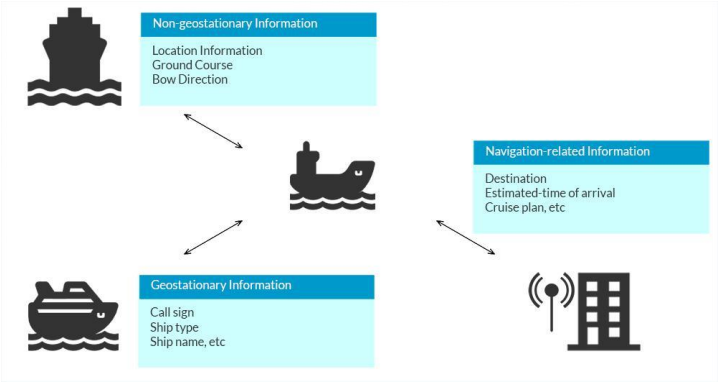


Gambar 2.1 IMTS sebagai *information provider*

Gambar 2.2 Sistem penerima data AIS di area pesisir

2.2 *Automatic Identification System* (AIS)

AIS adalah sebuah sistem pelacakan otomatis yang digunakan pada kapal untuk mengidentifikasi dan menemukan kapal melalui pertukaran data dengan kapal lain di dekatnya, BTS AIS, dan satelit. Ilustrasi pertukaran data ditunjukan oleh gambar 2.3. Informasi yang didapat dari AIS bisa dikelompokkan menjadi tiga yaitu geostasioner, non-geostasioner, dan navigasi. Informasi geostationer secara umum adalah spesifikasi kapal seperti nama kapal, jenis kapal, dan dimensi kapal. Informasi non-geostasioner antara lain lokasi, arah kapal. Informasi navigasi antara lain tujuan, perkiraan waktu kedatangan.



Gambar 2.3 Pertukaran data AIS

AIS pada kapal mengirimkan data berikut ini 2 sampai 10 detik tergantung kecepatan kapal ketika sedang bergerak, dan setiap 3 menit ketika kapal sedang dalam kondisi menurunkan jangkar [6].

1. Maritime Mobile Service Identity (MSSI) adalah sembilan digit angka unik untuk identifikasi kapal.
2. Status navigasi seperti *"at anchor", "underway using engine(s)",* dan *"not under command".*
3. *Rate of turn*, dari 0 sampai 720 derajat per menit.
4. *Speed over ground*, 0 – 102 knots (189 km/jam) dengan ketelitian 0.1 knot (0.19 km/jam).
5. Posisi berupa latitude dan longtitude.
6. *Course over ground* relatif terhadap arah utara.
7. *True Heading*, 0 to 359 derajat.
8. *True bearing* terhadap posisinya. 0 to 359 derajat.
9. UTC dalam detik – Waktu UTC dalam detik ketika data ini dikirim.

Lalu data berikut dikirim setiap 6 menit:

1. IMO yaitu tujuh digit angka yang tidak berubah setelah transfer pendaftaran ke negara lain.
2. *Radio call sign* adalah kode internasional mencapai tujuh karakter yang diberikan kepada kapal tersebut oleh negara tempat kapal terdaftar.
3. Nama yang terdiri dari dua puluh karakter.
4. Jenis kapal atau muatannya.
5. Dimensi kapal.
6. Lokasi dari antena sistem penentuan posisi sperti GPS pada kapal
7. Jenis sistem penentuan posisi yang digunakan seperti GPS, DGPS atau LORAN-C.
8. *Draught* adalah bagian dari kapal yang berada di bawah permukaan air dengan nilai 0.1 to 25.5 meter
9. Tujuan dengan maksimal dua puluh karakter.
10. ETA (*estimated time of arrival*) ke tujuan dalam format bulan/tanggal jam:menit berdasarkan waktu UTC.

2.3 Visualisasi Data Spasial

Teknik penampilan data mempengaruhi tingkat pemahaman pengguna dalam memahami informasi dari data tersebut [3]. Data spasial secara sederhana dapat di artikan sebagai data yang memiliki referensi

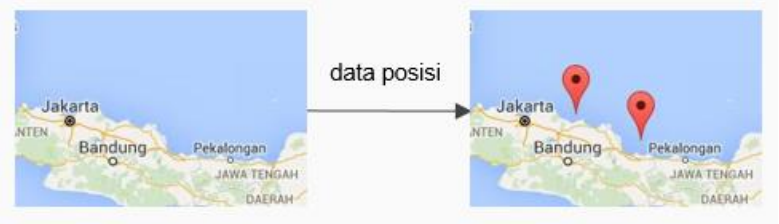
keruangan (geografi), biasanya divisualisasikan dalam tiga notasi. Notasi titik digunakan untuk pencitraan objek atau benda tunggal, tanpa panjang dan tanpa luasan serta ditampilkan dengan koordinat tunggal, contohnya adalah lokasi kapal. Notasi garis merupakan pencitraan terhadap beberapa notasi titik yang terhubung menjadi garis, mempunyai panjang namun tanpa luasan, mempunyai koordinat awal dan akhir, dapat mewakili suatu koordinat diskrit contohnya rute kapal. Notasi polygon merupakan gabungan dari beberapa notasi garis yang membentuk kurva tertutup, mempunyai panjang dan luasan dengan koordinat awal berhimpit dengan koordinat akhir contohnya adalah cuaca di suatu area.

2.4 *Geographic Information Systems* (GIS)

GIS adalah adalah sistem informasi khusus yang mengelola data yang memiliki informasi spasial (bereferensi keruangan). Atau dalam arti yang lebih sempit, adalah sistem komputer yang memiliki kemampuan untuk membangun, menyimpan, mengelola dan menampilkan informasi berefrensi geografis, misalnya data yang diidentifikasi menurut lokasinya, dalam sebuah *database* [2]. Data GIS biasanya terdiri dari data lokasi yang bersifat temporal dan informasi terkait lokasi tersebut [4]. Teknologi GIS dapat digunakan untuk investigasi ilmiah, pengelolaan sumber daya, perencanaan pembangunan, kartografi dan perencanaan rute.

2.5 Google Map API

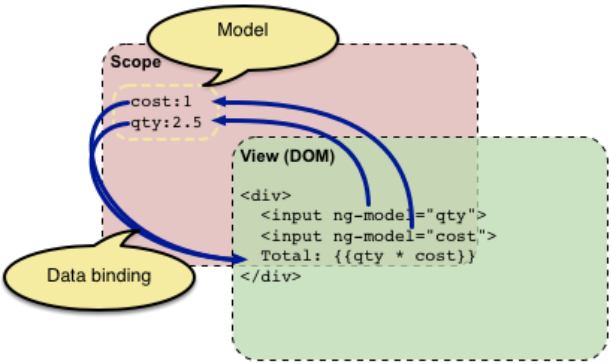
Salah satu contoh aplikasi GIS adalah Google Maps. Google Maps memiliki API yang memungkinkan pengembang menggunakan fitur-fitur dari Google Maps untuk menampilkan data terkait posisi pada Google Maps sebagai *marker* seperti ditunjukkan pada gambar 2.4. Google Maps API berbasis *Asynchronous Javascript* dan XML (AJAX), mendukung interaksi klien / server yang dapat mempertahankan hubungan secara kontinyu antara klien dan server [5]. Dengan ini pengguna dapat mengunduh informasi tambahan secara langsung dari peta secara efisien tanpa perlu memuat ulang data secara keseluruhan [2]. API ini terdiri dari struktur data, kelas objek atau fungsi yang dapat digunakan oleh pengembang menggunakan *Javascript*, PHP atau bahasa pemrograman lainnya.



Gambar 2.4 Menampilkan data posisi pada Google Maps

2.6 Angular JS

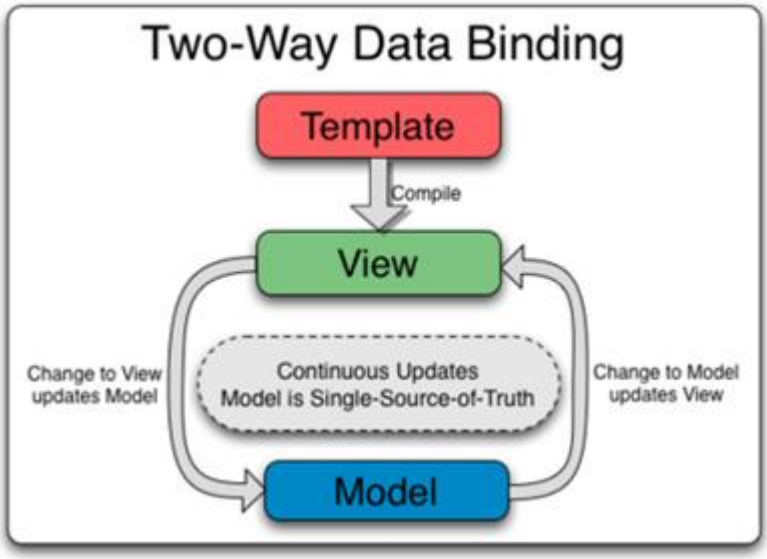
Angular JS atau biasa disebut Angular adalah *framework* terstruktur untuk website yang dinamis. Angular memungkinkan pengembang menambah sintak di HTML yang statis untuk mengekspresikan komponen aplikasi secara jelas dan ringkas. Angular juga mengusung konsep data *binding* dan *dependency injection* untuk mempermudah pengguna dalam hal mengurangi jumlah kode yang harus ditulis. Semua proses dari program Angular dilakukan di *browser*, hal ini membuat angular ramah dengan segala jenis server.



Gambar 2.5 Konsep data *binding*

Cara kerja Angular adalah dengan pertama kali membaca halaman HTML, yang mana telah ditambahkan *custom tag attribute.* Kemudian Angular akan menafsirkan *tag* tersebut sebagai arahan untuk mengatur masukan atau keluaran untuk model yang diwakili oleh variabel Javascript standar seperti ditunjukkan pada gambar 2.4. Nilai dari variabel

bisa diatur manual dalam kode, diambil dari sumber statis ataupun sumber dinamis dalam bentuk JSON. Selain itu, Angular konsep data *binding* yang digunakan angular adalah data *binding 2* arah yang memungkinkan untuk menampilkan perubahan data secara *realtime* yang diilustrasikan oleh gambar 2.5.



Gambar 2.6 Data *binding* 2 arah

2.6.1 NgMap.Js

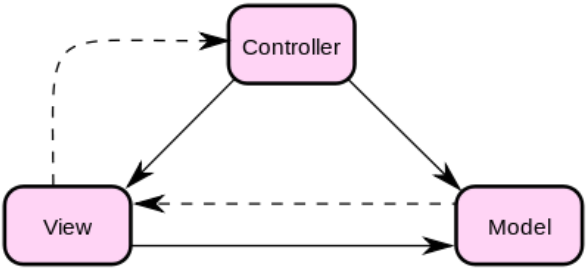
NgMap.js adalah salah satu *framework* untuk Google Map berbasis Angular JS. NgMap menyediakan fungsi dari Google Map API dalam bentuk *service, directive* dan *proveider* Angular JS sehingga implementasi Google Map API ke aplikasi yang menggunakan *framework* Angular menjadi lebih mudah. NgMap mengekspose semua Google Maps v3 API kepada pengguna tanpa melakukan *hiding, wrapping* dan sebagainya. Dengan begitu, pengembang tidak perlu mempelajari modul ini melainkan hanya perlu tahu Google Maps v3 API.

2.6.2 Angular UI Grid

UI Grid adalah modul Angular JS untuk mengolah data dalam bentuk tabel. UI Grid menggunakan implementasi native dari Angular JS tanpa jQuery. UI Grid dapat bekerja dengan baik untuk jumlah data yang besar. UI grid mengusung konsep *plugin architecture* yang memungkinkan pengembang bisa hanya menggunakan fitur yang diperlukan. Fitur umum dari UI Grid antara lain *sorting, filtering* dan *user interaction.*

2.7 Pemprograman MVC

Model-View-Controller atau MVC adalah sebuah metode untuk membuat sebuah aplikasi dengan memisahkan data (*Model*) dari tampilan (*View*) dan cara bagaimana memprosesnya (*Controller*) seperti yang diilustrasikan oleh gambar 2.4. Dalam implementasinya kebanyakan *framework* dalam aplikasi website adalah berbasis arsitektur MVC. MVC memisahkan pengembangan aplikasi berdasarkan komponen utama yang membangun sebuah aplikasi seperti manipulasi data, antarmuka pengguna, dan bagian yang menjadi kontrol dalam sebuah aplikasi web.



Gambar 2.7 Struktur MVC

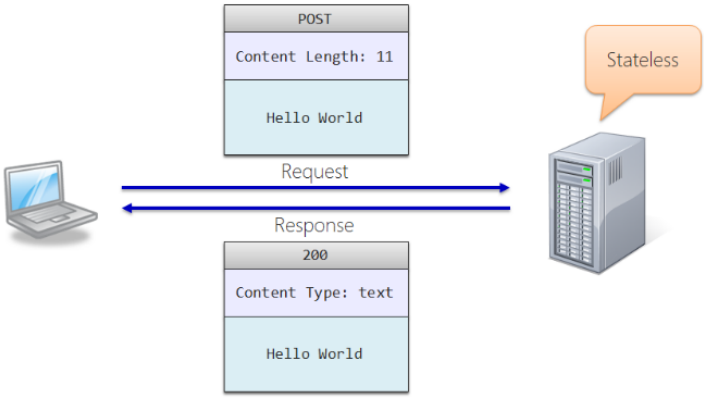
2.8 Code Igniter

CodeIgniter (CI) adalah *framework* untuk membangun *website* berbasis PHP. Tujuannya adalah untuk mempercepat pengerjaan proyek dengan disediakannya *libraries* atau fungsi yang umum digunakan sehingga tak perlu menulis dari awal. CI memiliki struktur MVC (*Model View Controller*), dengan mengikuti struktur ini kode yang dihasilkan menjadi lebih rapi.

2.9 REST API

API adalah singkatan dari *Application Program Interface*. API memungkinkan komunikasi dan pertukaran data antara dua sistem perangkat lunak terpisah. Sebuah sistem perangkat lunak yang menerapkan API berisi fungsi yang dapat dieksekusi oleh sistem perangkat lunak lain.

Pada REST API, REST adalah singkatan dari *Representational State Transfer*. REST API adalah metodologi yang dirancang agar program dapat berkomunikasi satu sama lain dengan cara yang sesederhana mungkin melalui HTTP protokol. Dengan kata lain REST API adalah mekanisme *request/response* sederhana. Ilustrasi sederhana untuk mekanisme *request/response* bisa dilihat dari gambar 2.1.



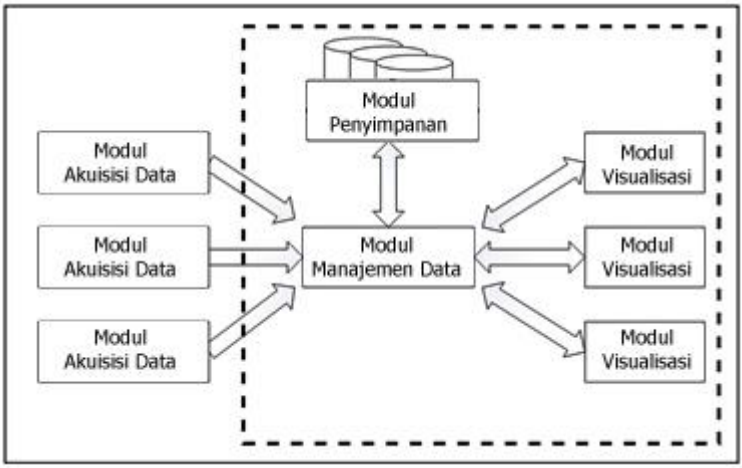
Gambar 2.8 Ilustrasi *request / response* REST API

Lebih jelasnya karakteristik dari REST API adalah sebagai berikut:

1. Menggunakan HTTP *methods* secara eksplisit. Standar HTTP *methods* antara lain PUT, GET, POST and DELETE.
2. *Stateless*, atau klien mengikut-sertakan semua kondisi informasi saat melakukan *request* ke server dan sebaliknya.
3. Mengirim data dalam format XML atau *Javascript Object Notation* (JSON).

BAB III
DESAIN DAN IMPLEMENTASI SISTEM

Desain IMTS memiliki 4 komponen utama, yaitu modul untuk akuisisi data, modul untuk menyimpan data, modul untuk mengatur data, dan modul untuk visualisasi data. Tugas akhir ini berfokus pada pembuatan sistem perangkat lunak dari IMTS untuk menampilkan data IMTS dari modul akuisisi data yang terdiri dari modul penyimpanan data, modul pengaturan data dan modul visualisasi data serta API untuk menjebatani antara modul pengaturan data dengan modul akuisisi data yang ditunjukkan oleh gambar 3.1.



Gambar 3.1 Modul yang dikerjakan

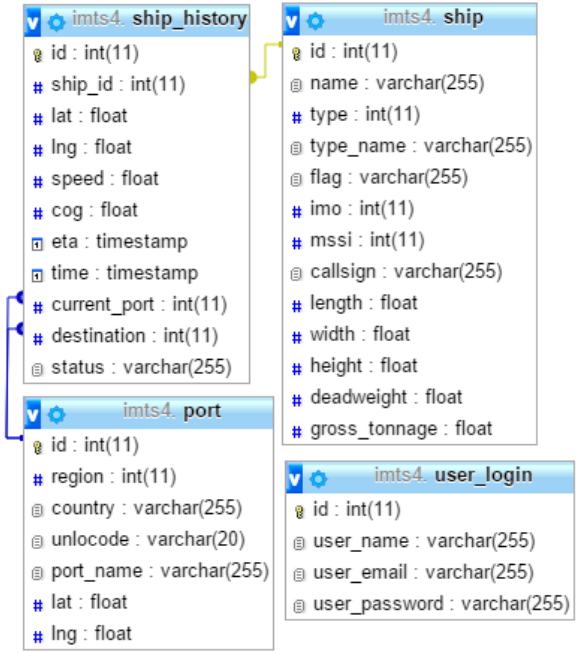
3.1 Modul Penyimpanan Data

Data IMTS adalah data dari AIS yang terpasang di kapal dan data pelabuhan. Contoh data AIS yang ditangkap sistem penerima dalam bentuk CSV ditunjukkan pada kode 3.1 yang berisi data MSSI, nama, *callsign*, dimensi, *course on ground*, latitude dan longtitude, tujuan, estimasi kedatangan dan tipe kapal. Mengacu dari data tersebut di desain

database IMTS dalam bentuk relasi diagram yang ditunjukkan gambar 3.3. *Database* dibagi menjadi empat tabel utama yaitu tabel ship untuk data kapal, tabel port untuk data pelabuhan, tabel ship_history untuk data riwayat pergerakan kapal dan tabel user_login untuk data akun pengguna. Pada tabel ship_history kolom ship_id memiliki relasi dengan kolom id pada tabel ship, sedangkan kolom current_port dan destination memiliki relasi dengan kolom id pada tabel port. *Database* yang digunakan untuk desain ini adalah MySQL *database*.

```
1,311376000,[A],13,1,,,  8.76 nm,317.3¢X, 0.10
Kn,201.9¢X,N/A,  8.73 nm,397.8 min,
7¢X10'37"S,112¢X41'47"E,N/A,0m,0m,,0/0
00:00,(0)Undefined ship type!        ,(0)N/A;
Harmless                ,
...,
```

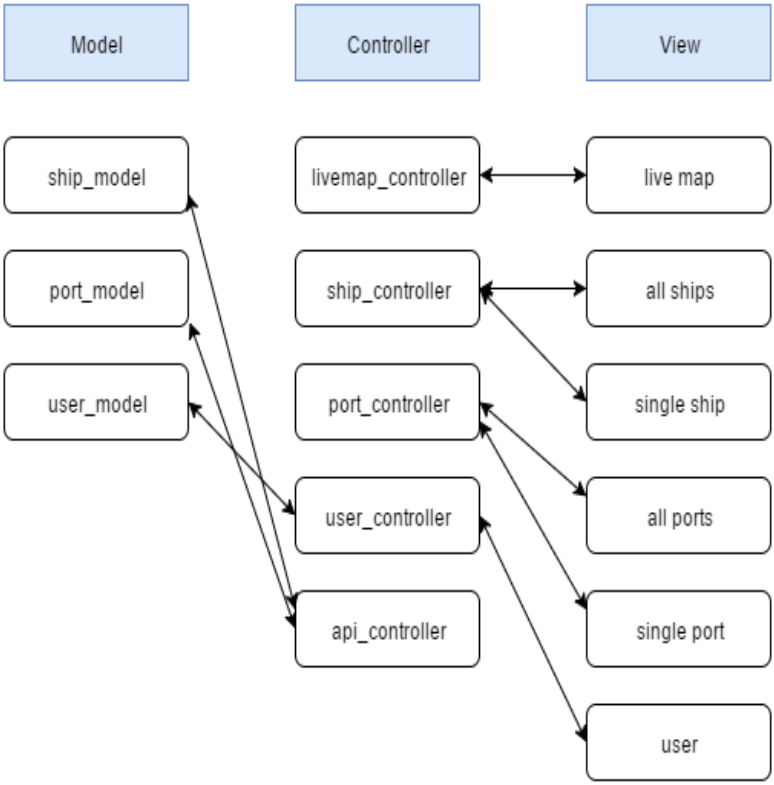
Kode 3.1 Data AIS dalam format CSV



Gambar 3.2 Desain *database*

3.2 Modul Manajemen Data dan Visualisasi Data

Modul manajemen data dibuat dengan struktur *Model-View-Controller* (MVC) yang ditunjukkan oleh gambar 3.3. *Model* dibagi menjadi tiga, yaitu model untuk kapal, pelabuhan dan pengguna. *Controller* dibagi menjadi empat, yaitu *controller* untuk kapal, pelabuhan, pengguna, dan API. Sedangkan *view* dibagi menjadi tujuh, yaitu *view* untuk halaman *live map, all ships, all ports, single ship, single port* dan *user*.



Gambar 3.3 Struktur Modul Manajemen Data

3.2.1 Desain Model

Ship_model bertanggung jawab melakukan *query* untuk data kapal. *Port_model* bertanggung jawab melakukan *query* untuk data pelabuhan. Sedangkan *user_model* bertanggung jawab melakukan *query* data akun pengguna. Setiap *model* berisi *query* untuk membaca, menambah, menghapus dan mengganti data terkait yang ada di database.

Pada *ship_model query* untuk membaca data di bagi menjadi tiga yaitu data untuk semua kapal, data untuk kapal yang tampil di layar peta (*viewport*), dan data untuk 1 kapal berdasarkan ID kapal. Berikut ini adalah *query* yang dijalankan oleh *ship_model*:

1. *Query* membaca data semua kapal ditunjukkan oleh kode 3.2 memberikan informasi hasil berupa spesifikasi kapal, data pergerakan terbaru kapal, port dimana kapal sedang berada, port terakhir kapal diketahui, port tujuan kapal serta status kapal untuk setiap kapal yang tersimpan di *database.*
2. *Query* membaca data kapal berdasarkan ID ditunjukkan oleh kode 3.3, memberikan informasi hasil berupa spesifikasi kapal, riwayat pergerakan kapal, port dimana kapal sedang berada, port terakhir kapal diketahui, port tujuan kapal serta status kapal dari kapal dengan ID terkait.
3. *Query* membaca data kapal berdasarkan *viewport* ditunjukkan oleh kode 3.4, memberikan informasi hasil berupa ID dari setiap kapal yang tampil di kapal. Data ID ini bisa digunakan untuk melakukan *query* data kapal sesuai ID kapal yang didapat.
4. *Query* untuk menambah, menghapus dan mengubah data ditunjukkan oleh kode 3.5. *Query* menambah dan mengubah data, format data dan data apa saja yang ditambah atau diubah diproses di *api_controller* sebelum dilewatkan ke *ship_model.*

```
public function get_all() {
  $query = $this->db->query("
    ...
  ");

  return $query->result()

}
```

Kode 3.2 *Query* data untuk semua kapal

```
public function get_ship($id) {
  $query = $this->db->query("...");
  return $query->result();
}
```

Kode 3.3 *Query* data kapal berdasarkan ID

```
public function
get_ship_vp($lat1, $lat2, $lng1, $lng2){
  $query = $this->db->query("...");
  return $query->result();
}}
```

Kode 3.4 *Query* data ID untuk kapal yang tampil di *viewport*

```
public function post_ship($data) {
  $this->db->insert('ship',$data);
}
public function delete_ship($id) {
  $this->db->where('id_tmp',$id);
  $this->db->delete('ship');
}
public function update_ship($id, $data) {
  $this->db->where('id_tmp',$id);
  $this->db->update('ship',$data);
}
```

Kode 3.5 *Query* untuk menambah, menghapus dan mengubah data kapal

Pada port_model *query* untuk membaca data di bagi menjadi tiga yaitu data untuk semua pelabuhan, data untuk pelabuhan yang tampil di layar peta (*viewport*), data untuk 1 pelabuhan berdasarkan ID pelabuhan. Berikut ini adalah *query* yang dijalankan pada port_model:

1. *Query* membaca data untuk semua port ditunjukkan oleh kode 3.6 memberikan hasil informasi berupa data umum dari pelabuhan, jumlah kapal yang berada di pelabuhan, jumlah keberangkatan kapal, jumlah kedatangan kapal dan jumlah kapal yang sedang menuju pelabuhan untuk setiap pelabuhan yang tersimpan di *database.*
2. *Query* membaca data pelabuhan yang tampil di *viewport* ditunjukkan kode 3.7 memberikan informasi hasil berupa data umum dari pelabuhan, jumlah kapal yang berada di pelabuhan dan jumlah kapal yang sedang menuju pelabuhan untuk pelabuhan yang tampil di *viewport*.

3. *Query* membaca data pelabuhan berdasarkan ID pelabuhan ditunjukkan oleh kode 3.8 memberikan informasi hasil berupa data umum dari port, jumlah kapal yang berada di port, jumlah keberangkatan kapal, jumlah kedatangan kapal dan jumlah kapal yang sedang menuju port beserta daftar nama kapal terkait untuk pelabuhan dengan ID terkait.
4. *Query* untuk menambah, menghapus dan mengubah data ditunjukkan oleh kode 3.9. Untuk *query* menambah dan mengubah data format data dan data apa saja yang ditambah atau diubah diproses di *api_controller* sebelum dilewatkan ke *port_model*.

```
public function get_all() {
  $query = $this->db->query("...");
  return $query->result();
}
```

Kode 3.6 *Query* data untuk semua port

```
public function
get_port_vp($lat1, $lat2, $lng1, $lng2){
  $query = $this->db->query("...");
  return $query->result();
}
```

Kode 3.7 *Query* data untuk port yang tampil di *viewport*

```
public function get_port($id) {
  $query = $this->db->query("...");
  return $query->result();
}
```

Kode 3.8 *Query* data untuk port berdasarkan ID-nya

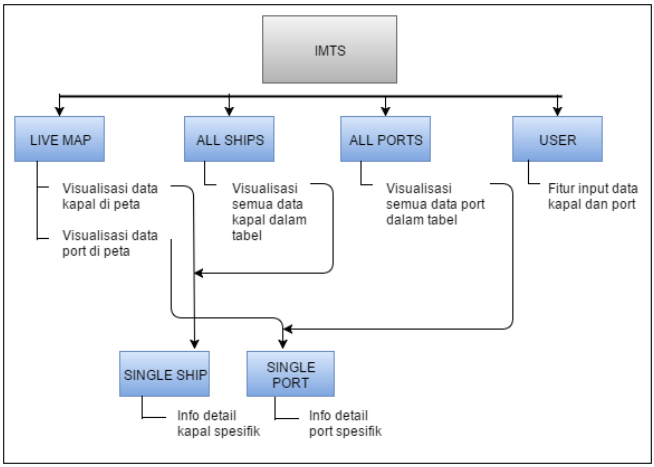
```
public function post_port($data) {
  $this->db->insert('port',$data);
}
public function delete_port($id) {
  $this->db->where('id_tmp',$id);
  $this->db->delete('port');
}
public function update_port($id, $data) {
  $this->db->where('id_tmp',$id);
  $this->db->update('port',$data);}
```

Kode 3.9 *Query* untuk menambah, menghapus dan mengubah data kapal

Sedangkan *user_model* bertanggung jawab mengolah data akun pengguna. Akun ini digunakan untuk masuk ke sistem dan memberikan pengguna hak akses untuk melakukan penambahan, pengubahan dan penghapusan data baik data kapal maupun data port.

3.2.2 Desain View

Halaman *live map* bertanggung jawab menampilkan data kapal dan pelabuhan dalam peta. Halaman *all ships* bertanggung jawab menampilkan data semua kapal dalam bentuk tabel untuk keperluan pencarian data kapal. Halaman *single ship* bertanggung jawab menampilkan data lengkap dari suatu kapal tertentu. Halaman *all ports* bertanggung jawab menampilkan data semua pelabuhan dalam bentuk tabel untuk keperluan pencarian pelabuhan. Halaman *single port* bertanggung jawab menampilkan data lengkap dari suatu pelabuhan tertentu dan daftar kapal yang sedang berada di pelabuhan, berangkat dari pelabuhan, baru saja tiba di pelabuhan, dan yang sedang menuju ke pelabuhan tersebut. Halaman *user* bertanggung jawab menampilkan halaman yang memberikan pengguna hak akses untuk melakukan penambahan atau pengubahan data kapal dan port setelah melakukan autentikasi akun. Struktur dari *view* ditunjukkan oleh gambar 3.4



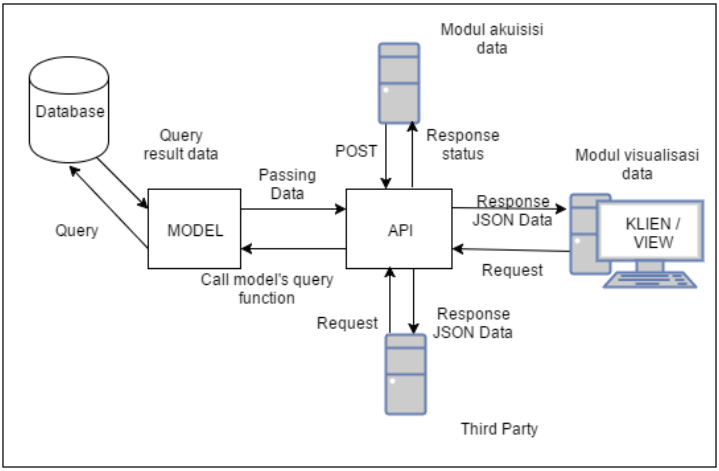
Gambar 3.4 Struktur dari *view*

3.2.3 Desain Controller

Controller bertanggung jawab sebagai jembatan antara *model* dan *view*. Pada aplikasi ini dibuat lima controller yaitu, *live_map_controller, ship_controller, port_controller, user_controller* dan *api_controller*. Disini *live_map_controller, ship_controller, dan port_controller* hanya berfungsi untuk menentukan *view* mana yang ditampilkan. Data yang ditampilkan pada *view* akan diatur oleh *api_controller*. Sedangkan *user_controller* berfungsi untuk melakukan autentikasi akun pengguna sebelum menampilkan halaman pengguna.

3.3 Desain REST API

REST API dibuat agar sistem bisa berkomunikasi dengan program lain. Pada dasarnya pemanggilan API ini sama saja dengan memanggil fungsi untuk melakukan *query* data yang terdapat pada model. Hasil dari *request* API ini adalah hasil *query* data yang diubah dalam bentuk JSON. Alur kerja API ditunjukkan oleh gambar 3.5. API dibagi menjadi 2 yaitu API untuk data kapal dan API untuk data pelabuhan. Masing-masing API ditentukan parameter yang diperlukan untuk mengambil data maupun menambah data.



Gambar 3.5 Alur *request/response* API IMTS

3.3.1 API Data Kapal

API untuk data kapal bisa dipanggil melalui URL dengan format seperti pada kode 3.10. Pemanggilan API ini akan memberikan respon berupa hasil *query* membaca data kapal pada *model_ship* sesuai parameternya dalam format JSON. *Request* GET pada API ini dibagi menjadi tiga, yaitu berdasarkan parameter *viewport*, ID, dan tanpa parameter. Sedangkan *Request* POST dibagi menjadi dua, yaitu untuk data spesifikasi kapal dan data riwayat pergerakan kapal.

```
http://server_address/api/ships
```

Kode 3.10 URL API untuk data kapal

Request GET dengan parameter *viewport* bisa dipanggil dengan format URL seperti kode 3.11 dan format parameter *viewport* yang ditunjukkan tabel 3.1. *Request* ini akan memberikan *response* berupa ID kapal mana saja yang tampil pada *viewport* yang dimaksud dengan format seperti kode 3.12.

Tabel 3.1 Parameter GET data kapal

Parameter	Nilai	Description
id	integer	ID unik kapal
lat1	float	Posisi *viewport* peta
lat2	float	
lng1	float	
lng2	float	

```
GET http://server_address/api/ships?
lat1=lat1&lng1=lng1&lat2=lat2&lng2=lng2
```

Kode 3.11 GET *request* untuk data kapal dengan parameter *viewport*

```
[
  {ship_id: integer},
  {ship_id: integer},
  ...
]
```

Kode 3.12 *Response* dari GET *request* data kapal dengan parameter *viewport*

Request GET dengan parameter ID bisa dipanggil dengan format URL seperti kode 3.13 dan format parameter ID yang ditunjukkan tabel 3.1. *Request* ini akan memberikan *response* berupa data spesifikasi kapal berserta riwayat pergerakannya dengan format seperti kode 3.14.

```
http://localhost/ci/imts/index.php/api/ships?id=id
```

Kode 3.13 GET *request* untuk data kapal dengan parameter ID

```
[
  {
     id_tmp: integer,
     name: string,
     imo: integer,
     mssi: integer,
     callsign: string
     flag: string,
     length: float,
     width: float,
     height: float,
     gross_tonnage: float
     deadweight: float,
     alpha_2: string,
     type: integer,
     type_name: string,
     lat: float
     lng: float,
     speed: float,
     status: string,
     time: time,
     last_known_port_id: integer,
     last_known_port_name: string,
     atd_time: time,
     current_port_name: string,
     destination_id: integer,
     destination_name: string,
     ata_time: integer
  },
  {
     //data kapal untuk waktu sebelumnya
  }
]
```

Kode 3.14 *Response* dari HTTP *request* data kapal dengan parameter ID

Request POST untuk data spesifikasi kapal bisa dipanggil dengan format parameter yang ditunjukkan tabel 3.2. Sedangkan untuk POST data riwayat pergerakan kapal memiliki format parameter yang ditunjukkan oleh tabel 3.3. *Request* POST akan memberikan status berupa *booelan* dimana akan bernilai *true* jika POST sukses dan bernilai *false* ketika ada parameter yang salah.

Tabel 3.2 Parameter POST data spesifikasi kapal

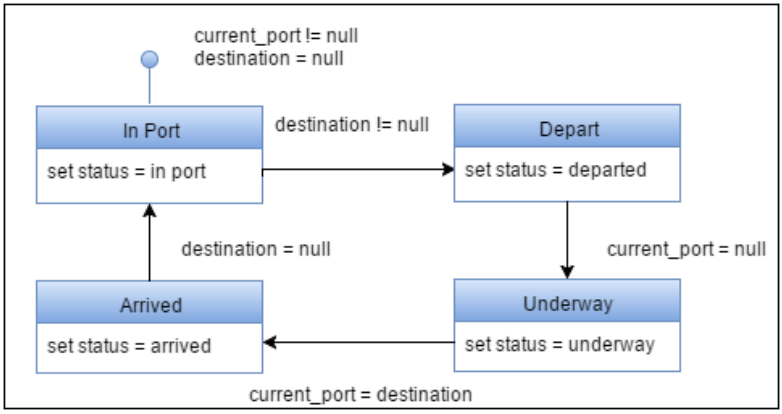
Parameter	Nilai	Description
data	1	POST untuk data spesifikasi kapal
name	string	Nama kapal
imo	integer	IMO
mssi	integer	MSSI
callsign	string	Callsign
flag	string	Kode negara
type	string	Tipe kapal
deadweight	float	Deadweight
gross_tonnage	float	Gross Tonnage

Tabel 3.3 Parameter POST data riwayat pergerakan kapal

Parameter	Nilai	Description
history	1	POST untuk data riwayat kapal
id	integer	ID unik kapal
lat	float	Latitude kapal
lng	float	Longtitude kapal
speed	float	Kecepatan kapal
cog	float	*Course on ground*
destination	integer	ID unik pelabuhan tujuan kapal
eta	timestamp	Estimasi waktu kedatangan

Pada saat POST data kapal beberapa data diproses terlebih dahulu sebelum disimpan ke *database*. Data tersebut adalah data status pergerakan kapal, dan status posisi kapal berada di pelabuhan mana (*current port*). Data status pergerakan kapal sudah ada dari data yang

dikirim AIS. Pengolahan ulang ini dimaksudkan sebagai antisipasi jika sensor AIS rusak dan tidak mengirim data tersebut maka data dari hasil perhitungan yang akan digunakan. Data status kapal didapat dari data tujuan kapal dan *current port*. *State diagram* dari status pergerakan kapal ditunjukkan oleh gambar 3.6 dimana status didefinisikan menjadi 4 *state* yaitu berada di pelabuhan, berangkat dari pelabuhan, sedang dalam perjalanan dan tiba di pelabuhan . Sedangkan *current port* didapat dari *query* data pelabuhan terdekat dengan parameter *viewport* dengan radius 1 km dimana titik tengah adalah posisi terbaru kapal.



Gambar 3.6 *State diagram* status kapal

3.3.2 API Data Pelabuhan

API untuk data pelabuhan bisa dipanggil melalui URL dengan format seperti pada kode 3.15 Pemanggilan API ini akan memberikan respon berupa hasil *query* membaca data pelabuhan pada *model_port* sesuai parameternya dalam format JSON. *Request* GET pada API ini dibagi menjadi empat, yaitu berdasarkan parameter *viewport*, ID, *portcall* dan tanpa parameter. Sedangkan *Request* POST hanya untuk data spesifikasi dari pelabuhan

```
http://server_address/api/ports
```

Kode 3.15 URL API data port

Request GET dengan parameter *viewport* bisa dipanggil dengan format URL seperti kode 3.16 dan format parameter *viewport* yang ditunjukkan tabel 3.4. *Request* ini akan memberikan *response* berupa data pelabuhan yang tampil pada *viewport* yang dimaksud beserta jumlah kapal yang ada di pelabuhan tersebut dan yang sedang perjalanan menuju pelabuhan tersebut dengan format seperti kode 3.17.

Tabel 3.4 Parameter GET data port

Parameter	Nilai	Description
id	integer	ID unik pelabuhan
lat1	float	Posisi *viewport* peta
lat2	float	
lng1	float	
lng2	float	
portcall	"ship_in", "arrivals", "departures", "expected_arrivals"	Status kapal terhadap pelabuhan

```
GET http://server address/api/ports?
lat1=lat1&lng1=lng1&lat2=lat2&lng2=lng2
```

Kode 3.16 GET *request* data pelabuhan berdasarkan parameter *viewport*

```
[
 {
  id_tmp: integer,
  port_name: string,
  country: string,
  unlocode: string,
  lat: float,
  lng: float,
  ships_in_port: integer,
  expected_arrivals: integer
 },
 {
   //data port lain
   ...
 }
]
```

Kode 3.17 *Response* dari GET *request* untuk data pelabuhan dengan parameter *viewport*

Request GET dengan parameter ID bisa dipanggil dengan format URL seperti kode 3.18 dan format parameter ID yang ditunjukkan tabel 3.4. *Request* ini akan memberikan *response* berupa data spesifikasi pelabuhan beserta data *portcall* untuk pelabuhan dengan ID terkait dengan format seperti kode 3.19.

```
GET http://server address/api/ports?id=id
```

Kode 3.18 GET *request* data pelabuhan berdasarkan parameter ID

```
[
  {
   port_name: string,
   country: string,
   unlocode: string,
   lat: float,
   lng: float
   ships_in_port: integer,
   departures: integer,
   arrivals: integer,
   expected_arrivals: integer,
   country_name: string
  }
]
```

Kode 3.19 *Response* dari GET *request* untuk data kapal dengan parameter ID

Request GET dengan parameter *portcall* bisa dipanggil dengan format URL seperti kode 3.20 dan format parameter *portcall* dan id yang ditunjukkan tabel 3.4. *Request* ini akan memberikan *response* berupa daftar kapal yang memiliki status *portcall* yang diminta untuk pelabuhan dengan ID terkait dengan format seperti kode 3.21.

```
http:// server address /api/ports?portcall=
portcall&id=id
```

Kode 3.20 GET *request* data pelabuhan berdasarkan parameter *portcall*

```
[
 {
   ship_id: integer,
   name: string,
```

```
    type: integer,
    alpha_2: string,
    time: time
  },
  {
    //data kapal lain
    ...
  }
]
```

Kode 3.21 *Response* dari HTTP *request* untuk data kapal dengan parameter portcall

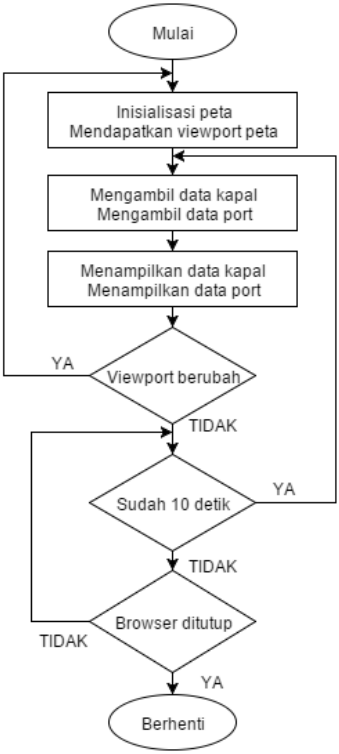
Request POST untuk data spesifikasi pelabuhan bisa dipanggil dengan format parameter yang ditunjukkan tabel 3.5. *Request* POST akan memberikan status berupa *booelan* dimana akan bernilai *true* jika POST sukses dan bernilai *false* ketika ada parameter yang salah.

Tabel 3.5 Parameter POST data port

Parameter	Nilai	Description
data	1	POST untuk data pelabuhan
name	string	Nama pelabuhan
unlocode	string	Kode pelabuhan
country	string	Kode negara

3.4 Implementasi Visualisasi pada Live Map

Implementasi visualisasi data pada live map terdiri dari visualisasi untuk data kapal dan data pelabuhan. Alur visualisasi ini ditunjukkan oleh gambar 3.7. Ketika peta dibuka akan dilakukan inisialisasi objek peta dari Google Maps yang dilakukan oleh NgMap.Js yang ditunjukkan oleh kode 3.22. Dari objek tersebut didapatkan *viewport* atau daerah pada yang tampil pada layar *browser*. Lalu dilakukan *query* data oleh Angular melalui *request* GET pada API berdasarkan parameter *viewport* tersebut untuk mendapatkan data kapal dan pelabuhan. Kemudian data ditampilkan pada peta. Ketika peta digeser maka viewport peta juga berubah, maka dilakukan *query* data ulang untuk mengambil data pada area peta yang baru. Data secara otomatis akan di muat ulang setiap 10 detik untuk mendapatkan data terbaru.



Gambar 3.7 Alur visualisasi live map

```
NgMap.getMap().then(function(map) {
 viewport={
   lat1 : map.getBounds().getNorthEast().lat(),
   lng1 : map.getBounds().getNorthEast().lng(),
   lat2 : map.getBounds().getSouthWest().lat(),
   lng2 : map.getBounds().getSouthWest().lng()}
 $rootScope.getShip();
 $rootScope.getPort();
 $interval(function(){
   $rootScope.refreshMap(map);
 }, 30000);
});
```

Kode 3.22 Fungsi inisialisasi peta pada halaman live map

Pada halaman *live map* ketika ada *marker* yang posisinya terlalu berdekatan, maka akan dilakukan *clustering* dimana *marker-marker* yang berdekatan akan diwakili oleh ikon *cluster* beserta angka yang menunjukkan ada berapa *marker* yang berdekatan yang diwakili oleh *cluster* tersebut seperti ditunjukkan pada gambar 3.8. Ketika ikon *cluster* tersebut diklik maka secara otomatis peta akan diperbesar sampai pada tingkat dimana posisi *marker-marker* tidak terlalu berdekatan. .

Gambar 3.8 *Clustering* pada peta

3.4.1 Visualisasi Kapal

Visualisasi kapal diwakili dengan ikon kapal dengan warna yang berbeda-beda sesuai jenis kapal yang ditunjukkan oleh gambar 3.9. Kapal ditampilkan pada peta sesuai posisi terbaru dan sudut arah pergerakannya ditunjukkan oleh kode 3.23. Sudut arah pergerakan kapal diukur dari arah utara sebagi titik 0 derajat.

```
<custom-marker
  ng-if="displayShip"
  ng-repeat="ship in ships"
  position="[{{ship.lat}}, {{ship.lng}}]
```

```
  on-mouseover="showTooltip(ship.id_tmp, ship.name,
    ship.time, ship.destination_name, ship.flag)">
    <div
      ng-class="{active: isActive(ship.id_tmp)}"
      ng-click="select(ship.id_tmp)">
        <div
          class="ship ship_type{{ship.type}}"
          uib-tooltip-html="shipTooltip"
          tooltip-placement="right"
          uib-popover-template=
            "shipPopover.templateUrl"
          popover-placement="right"
          popover-is-open=
            "shipPopover.isOpen == {{ship.id_tmp}}"
          ng-click="portPopover.open(ship.id_tmp)"
          style="transform:
            rotate({{ship.bearing}}deg)">
        </div>
    </div>
</custom-marker>
```

Kode 3.23 Menampilkan kapal di live map

Gambar 3.9 Ikon kapal pada *live map*

Pada live map data kapal diambil berdasarkan parameter *viewport* untuk menentukan ID kapal mana saja yang tampil pada *viewport*. Lalu dilakukan *query* data untuk masing-masing kapal berdasarkan ID kapal untuk mendapatkan riwayat yang ditunjukkan oleh kode 3.24.

```
$rootScope.getShip = function(){
  var _parameter     =
  'lat1='+viewport.lat1+'&lat2='+viewport.lat2+'&lng
  1='+viewport.lng1+'&lng2='+viewport.lng2;
  imtsData.getData("ships", _parameter)
    .then(function(ship){
        for(i = 0; i < ship.data.length; i++){
          var id = JSON.parse(ship.data[i].ship_id);
          $rootScope.getPos(id);
        }
    })
}
```

Kode 3.24 Mendapatkan ID kapal yang tampil di *viewport*

Ketika data yang diambil tidak terdapat data sudut kapal dikarenakan sensor AIS yang rusak atau karena permasalahan yang lain maka dilakukan perhitungan arah pergerakan kapal yang ditunjukkan oleh kode 3.25. Pada pengambilan riwayat posisi inilah dilakukan penghitungan arah pergerakan kapal bersasarkan posisi terbaru dengan posisi sebelumnya. Sebelum data-data ini ditampilkan, akan dicek terlebih dahulu apakah data untuk kapal tersebut sudah ada atau belum. Jika belum data kapal akan ditambahkan, sedangkan jika sudah data kapal akan diganti dengan data yang baru. Hal ini dimaksudkan agar data yang tampil pada *live map* hanya data terbaru dari setiap kapal sehingga tidak mungkin akan ada 1 kapal yang sama pada beberapa posisi yang berbeda yang ditunjukkan dengan kode 3.26.

```
service.getShipBrng = function(route){
  var _curLat, _curLon, _prevLat, _prevLon,
    _curLatR, _prevlatR, _dLat, _dLon, _R, _a, _c,
    _distance;

  _curLat      = route[0].lat;
  _curLng      = route[0].lng;
  _prevLat     = route[1].lat;
  _prevLng     = route[1].lng;

  _curLatR     = _curLat * Math.PI / 180;
  _prevLatR    = _prevLat * Math.PI / 180;

  _dLng        = (_curLng-_prevLng) * Math.PI / 180;
```

```
  _y    = Math.sin(_dLng)*Math.cos(_curLatR);
  _x    = Math.cos(_prevLatR) *Math.sin(_curLatR) -
          Math.sin(_prevLatR) * Math.cos(_curLatR) *
          Math.cos(_dLng);

  _shipBrng   = ((Math.atan2(_y, _x) * 180 /
                Math.PI) + 360) % 360;
  return _shipBrng;
}
```

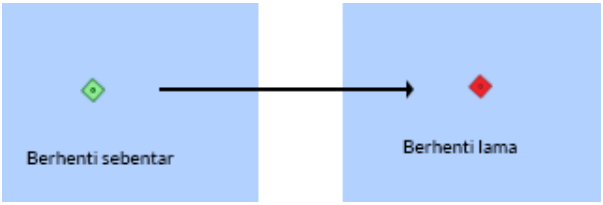
Kode 3.25 Servis untuk mendapatkan sudut arah pergerakan kapal

```
$rootScope.getPos = function(id) {
  var _parameter = "id="+id;
  imtsData.getData("ships", _parameter)
    .then(function(ship){
      var _theShip    = ship.data[0];
      var _positions  = [];
      for(i = 0; i < ship.data.length; i++){
        var _pos = {
          lat: JSON.parse(ship.data[i].lat),
          lng: JSON.parse(ship.data[i].lng)}
          _positions.push(_pos);
      }

      if(_positions.length > 1){
        var _currentRoute =
          [_positions[0],_positions[1]];
      } else {
        var _currentRoute =
          [_positions[0], _positions[0]];
      }

      var _bearing =
        imtsData.getShipBrng(_currentRoute);

      _theShip.positions     = _positions;
      _theShip.bearing       = _bearing;

      if($rootScope.listShips.indexOf(id) === -1) {
        $rootScope.listShips.push(id);
        $rootScope.ships.push(_theShip);
      } else {
```

```
        var _idx = $rootScope.listShips.indexOf(id);
        $rootScope.ships[_idx] = _theShip;
      }
    })
  }
```

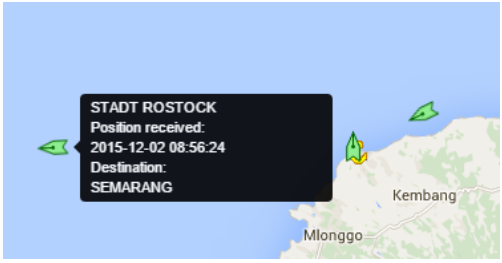
Kode 3.26 Mendapatkan data dan riwayat posisi kapal

Ketika kapal sedang berhenti atau sedang melakukan *anchoring* ikon kapal ditampilkan menjadi bujur sangkar. Ikon akan berubah warna menjadi merah seperti pada gambar 3.10 dan berkedip-kedip jika kapal tersebut berhenti untuk waktu yang lama untuk tanda peringatan bahwa kapal tersebut sedang melakukan sesuatu yang bisa saja adalah sesuatu yang ilegal.



Gambar 3.10 Ikon kapal saat berhenti

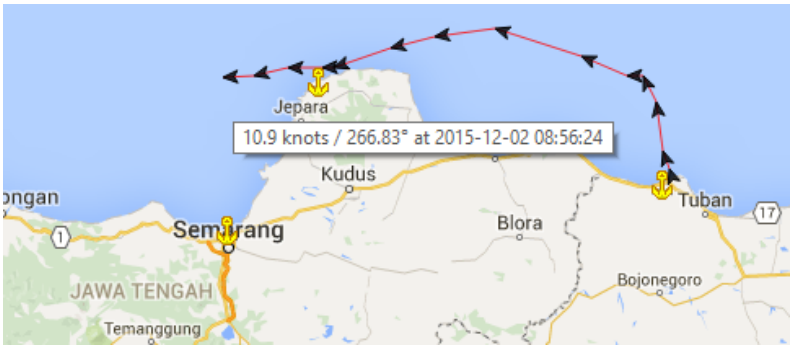
Pada gambar 3.11 ketika ikon kapal di-*hover* akan muncul info singkat berisi nama kapal, waktu terakhir data diterima, dan pelabuhan tujuan. Pada gambar 3.12 ketika ikon kapal diklik akan muncul *popover* berisi info yang sedikit lengkap dengan tambahan data jenis kapal, ukuran, kecepatan dan arah pergerakan kapal serta menu untuk melihat riwayat posisi, mencari jarak dan menu untuk melihat informasi kapal lebih detail.



Gambar 3.11 Ikon kapal pada peta ketika di-*hover*

Gambar 3.12 Ikon kapal pada peta ketika diklik

Ketika menu '*show vessel track*' diklik maka akan ditampilkan riwayat posisi dari kapal tersebut menggunakan *script* yang ditunjukkan oleh kode 3.27. Ikon kapal lain akan dihilangkan untuk sementara agar *live map* fokus pada riwayat posisi kapal. Pada tampilan ini ikon kapal divisualisasikan dengan ikon segitiga berwarna hitam yang diletakkan sesuai posisi dan waktunya. Jalur kapal divisualisasikan sebagai garis yang menghubungkan ikon-ikon kapal yang ditunjukkan pada gambar 3.13. Ketika ikon kapal di-*hover* maka akan tampil info kecepatan, sudut arah pergerakan dan waktu ketika data tersebut diterima.



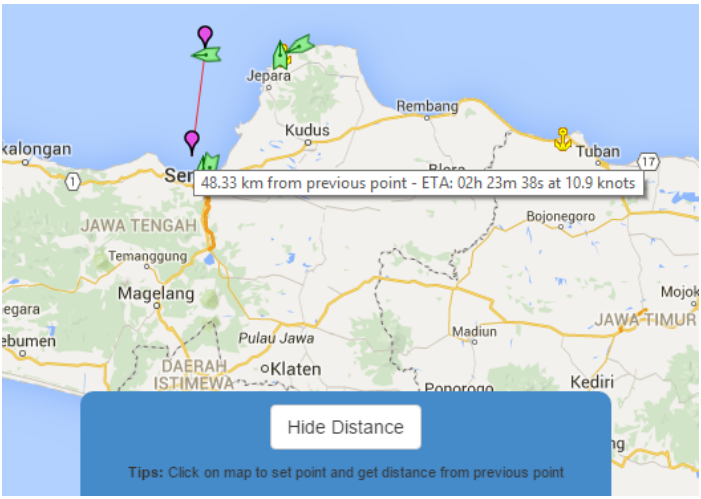
Gambar 3.13 Riwayat posisi kapal

```
$rootScope.getTrack = function(id){
  $rootScope.shipPopover.isOpen = false;
  $rootScope.displayShip = false;
  $rootScope.trackActive = true;
  var _parameter = "id="+id;

  imtsData.getData("ships", _parameter)
   .then(function(ship){
      var _positions= [];
      var _time = [];
      var _speed = [];
      for(i = 0; i < ship.data.length; i++){
       var _pos = {
         lat: JSON.parse(ship.data[i].lat),
         lng: JSON.parse(ship.data[i].lng)}
       var _t = ship.data[i].time;
       var _s = ship.data[i].speed
       _positions.push(_pos);
       _time.push(_t);
       _speed.push(_s);
      }

      var _trackLine = new google.maps.Polyline({
       path : _positions,
       geodesic : true,
       strokeColor : '#FF0000',
       strokeOpacity: 0.8,
       strokeWeight : 1
      });

      NgMap.getMap().then(function(map) {
        _trackLine.setMap(map);
      });

      var _markers = [];
      var _symbol = {};
      var _brng = "";
      var _tracks = [];
      _markers = imtsData.getShipTrack(_positions);
      NgMap.getMap().then(function(map) {
       for(i = 0; i < _markers.length; i++){
         if(i != _markers.length - 1){
           _markers[i].symbol =
            imtsData.getShipSymbol("track",
```

```
            imtsData.getShipBrng(_markers[i].route))
        } else {
          _markers[i].symbol =
            imtsData.getShipSymbol("track",
            imtsData.getShipBrng(
              _markers[i-1].route));
        }

      _brng =_markers[i].symbol.rotation.toFixed(2);
      _tracks[i] = new google.maps.Marker({
         map : map,
         icon : _markers[i].symbol,
         position : {lat:_markers[i].route[0].lat,
          lng:_markers[i].route[0].lng},
         title : _speed[i]+" knots / "+_brng+"° at
          "+_time[i]
         .....
```

Kode 3.27 Menampilkan riwayat posisi kapal pada *live map*

Pada gambar 3.14 ketika menu '*Distance to'* diklik maka akan ditampilkan ikon penanda berwarna ungu sebagai titik awal pada posisi dari kapal tersebut. Ketika *live map* diklik akan dimunculkan ikon penanda baru pada titik lokasi yang diklik. Titik penanda baru dihubungkan dengan garis dengan titik penanda sebelumnya. Ketika titik penanda baru di-*hover* akan ditampilkan jarak antara titik tersebut dengan titik tersebut dalam satuan kilometer dan estimasi waktu untuk sampai titik tersebut dengan asumsi kecepatan kapal konstan. Setiap titik penanda bisa ditarik atau digeser untuk mengganti lokasi titik tersebut. Ini dilakukan meggunakan *script* yang ditunjukkan oleh kode 3.28.



Gambar 3.14 Mencari jarak pada *live map*

```
service.getDistance = function(route){
 var _curLat, _curLon, _prevLat, _prevLon,
   _curLatR, _prevlatR, _dLat, _dLon, _R, _a, _c,
   _distance;
 _curLat     = route[0].lat;
 _curLng     = route[0].lng;
 _prevLat    = route[1].lat;
 _prevLng    = route[1].lng;
 _curLatR    = _curLat * Math.PI / 180;
 _prevLatR   = _prevLat * Math.PI / 180;
 _dLat       = (_curLat-_prevLat) * Math.PI / 180;
 _dLng       = (_curLng-_prevLng) * Math.PI / 180;
 _a = Math.sin(_dLat /2) * Math.sin(_dLat /2) +
      Math.cos(_curLatR) * Math.cos(_prevLatR) *
      Math.sin(_dLon/2) * Math.sin(_dLon/2);
 _c = 2 * Math.atan2(
      Math.sqrt(_a), Math.sqrt(1-_a));
 _R = 6371; // km
 _distance = _R * _c;

 return _distance;
}
```

Kode 3.28 Servis untuk mendapatkan jarak antara dua titik posisi

3.4.2 Visualisasi Port

Visualisasi port diwakili dengan ikon jangkar berwarna kuning seperti ditunjukkan pada gambar 3.15 menggunakan *script* yang ditunjukkan oleh kode 3.29. Port ditampilkan pada peta sesuai koordinat posisinya. Pada gambar 3.16 ketika port di-*hover* akan muncul info singkat berisi nama port tersebut. Pada gambar 3.17 ketika ikon port diklik akan muncul *popover* berisi info yang sedikit lengkap dengan tambahan data unlocode, jumlah kapal yang ada di port, jumlah kapal yang sedang menuju port tersebut dan menu untuk melihat informasi port lebih detail.

```
<custom-marker
 ng-repeat="port in ports"
 position="[{{port.lat}}, {{port.lng}}]">
  <div
   ng-class="{active: isActive(port)}"
   ng-click="select(port)">
    <div
     class="port"
     uib-popover-template=
      "portPopover.templateUrl"
     popover-placement="right"
     popover-is-open=
      "portPopover.isOpen == {{port.id_tmp}}"
     ng-click="portPopover.open(port.id_tmp)"
     ng-show="displayPort">
    </div>
  </div>
</custom-marker>
```

Kode 3.29 Menampilkan kapal di *live map*

Gambar 3.15 Ikon port pada *live map*

Gambar 3.16 Ikon port pada *live map* ketika di-*hover*

Gambar 3. 17 Ikon port pada *live map* ketika diklik

Pada *live map* data port diambil berdasarkan parameter *viewport* untuk menentukan port mana saja yang tampil pada *viewport* yang ditunjukkan oleh kode 3.30. Lalu dilakukan *query* data untuk setiap port yang berada pada *viewport*. Sebelum data-data ini ditampilkan, akan dicek terlebih dahulu apakah data untuk port tersebut sudah ada atau belum. Jika belum data kapal akan ditambahkan, sedangkan jika sudah data kapal akan diganti dengan data yang baru. Hal ini dimaksudkan agar data yang tampil pada *live map* hanya data terbaru dari setiap port sehingga tidak terjadi penumpukan ikon port pada lokasi yang sama.

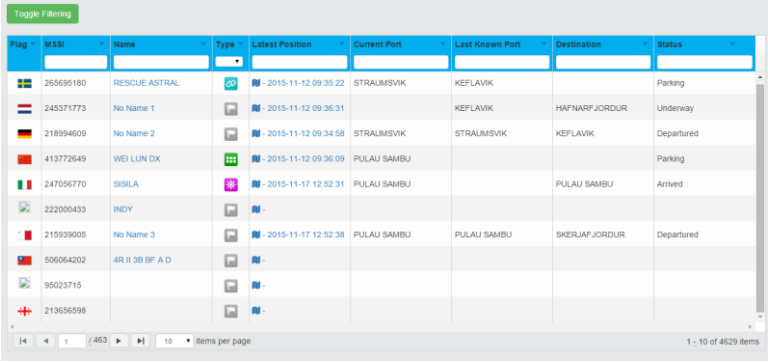
```
$rootScope.getPort = function(){
  var _parameter =
    'lat1='+viewport.lat1+'&lat2='+viewport.lat2+
    '&lng1='+viewport.lng1+'&lng2='+viewport.lng2;
  imtsData.getData("ports", _parameter)
  .then(function(port){
    for(i = 0; i < port.data.length; i++){
      if(
```

```
        $rootScope.listPorts.indexOf(
          port.data[i].id_tmp) === -1) {
          $rootScope.listPorts.push(
            port.data[i].id_tmp);
          $rootScope.ports.push(port.data[i])
      } else {
          var _idx =
             $rootScope.listPorts.indexOf(
              port.data[i].id_tmp);
          $rootScope.ports[_idx] = port.data[i];
      }
    }
  })
}
```

Kode 3.30 Mendapatkan data port yang tampil di *viewport*

3.5 Implementasi Visualisasi *All Ships* dan *All Ports*

Implementasi visualisasi data pada *all ships* merupakan visualisasi untuk data semua kapal dalam bentuk tabel dengan *pagination* menggunakan modul Angular UI Grid seperti pada gambar 3.18. Tabel terdiri dari kolom negara, MSSI, nama, tipe, posisi terbaru, pelabuhan terdekat, pelabuhan terakhir kali kapal terlihat, tujuan kapal serta status kapal. Tabel dibuat memiliki fitur untuk pencarian dan penyaringan berdasarkan kolom yang ada. Pada gambar 3.19 ditunjukan hasil pencarian pada kolom MSSI dan menghasilkan 1 kapal dengan yang sesuai kriteria pencarian.

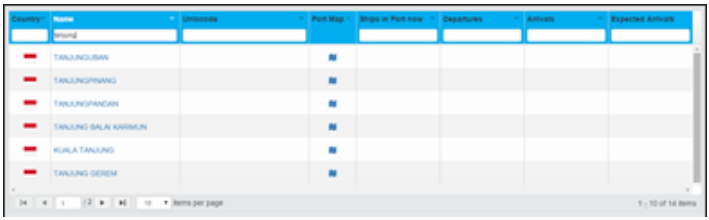


Gambar 3.18 Tabel *All Ship*

Gambar 3.19 Tabel *All Ship* ketika data difilter

Implementasi visualisasi data pada *all port* merupakan visualisasi untuk data semua pelabuhan dalam bentuk tabel dengan *pagination* seperti pada gambar 3.20. Tabel terdiri dari kolom negara, MSSI, nama, tipe, lokasi, jumlah kapal yang ada di pelabuhan, jumlah kapal yang sedang berangkat, jumlah kapal yang baru tiba dan jumlah kapal yang sedang menuju pelabuhan. Tabel dibuat memiliki fitur untuk pencarian dan penyaringan berdasarkan kolom yang ada. Pada gambar 3.21 ditunjukan hasil pencarian pada kolom nama dan menghasilkan 14 pelabuhan yang sesuai kriteria pecarian yang dibagi menjadi 2 *pagination*.

Gambar 3.20 Tabel *All Port* ketika data difilter



Gambar 3.21 Tabel *All Ports* ketika data difilter

Halaman ini sengaja dikosongkan

BAB IV
PENGUJIAN DAN ANALISIS

Pada bab ini akan dijelaskan mengenai pengujian aplikasi yang telah diimplementasikan apakah telah sesuai dengan desain sistem yang telah dirancang. Adapun pengujian yang akan dilakukan sebagai berikut:

1. Pengujian API
2. Pengujian Visualisasi

4.1 Pengujian API

Pengujian ini dilakukan untuk mengetahui apakah API yang dirancang memberikan *response* yang diinginkan untuk setiap *request* yang dikirim. Pengujian dilakukan dengan memanggil API untuk *request* data yang berbeda untuk setiap parameter dan mengetahui besar ukuran data yang dimuat setiap mengambil data untuk ditampilkan pada *live map* seperti yang ditunjukkan gambar 4.1.

4.1.1 Pengujian API untuk Data Kapal

Pengujian API untuk data kapal dilakukan dengan melakukan *request* data kapal untuk setiap parameter. Pengujian *request* GET data kapal berdasarkan parameter *viewport* memberikan contoh *response* seperti ditunjukkan oleh kode 4.1. Hasil untuk pengujian ini selengkapnya ditunjukkan oleh tabel 4.1. Pengujian *request* GET data kapal berdasarkan parameter ID memberikan contoh *response* seperti ditunjukkan oleh kode 4.2. Hasil untuk pengujian ini selengkapnya ditunjukkan oleh tabel 4.2. Sedangkan pengujian *request* POST data riwayat pergerakan kapal ditunjukkan oleh tabel 4.3. Pengujian ini dilakukan dengan mengubah-ubah parameter latitude dan longtitude kapal menggunakan *dummy* data untuk posisi.

```
[
  {
    ship_id: "1536"
  },
  {
    ship_id: "2771"
  },
```

```
  {
    ship_id: "3380"
  },
  {
    ship_id: "1529"
  },
  {
    ship_id: "4629"
  },
  {
    ship_id: "4311"
  }
]
```

Kode 4.1 Contoh *response* dari *request* data kapal berdasarkan parameter *viewport*

Tabel 4.1 API *request* untuk GET data kapal berdasarkan parameter *viewport*

No	*Parameter*	*Response*
1	lat1 = -5.6350 lng1 = 114.7518 lat2 = -8.3609 lng2 = 107.2481	Menghasilkan respon 6 data kapal.
2	lat1 = 2.0554 lng1 = 116.3256 lat2 = 0.4353 lng2 = 112.573	Menghasilkan respon *not found* untuk peta yang menampilkan daerah daratan di tengah pulau Kalimantan
3	lat1 = null lng1 = 104.7379 lat2 = 0.8567 lng2 = 102.8620	Menghasilkan respon status *error* karena ada parameter *viewport* yang kosong (lat1).
4	lat1 = 5.0825 lng1 = 121.5848 lat2 = abcd lng2 = 114.0811	Menghasilkan respon status *error* karena ada parameter *viewport* yang diisi string (lat2).
5	lat1 = 4.3654 lng1 = 190.6540 lat2 = -2.1117 lng2 = 115.6467	Menghasilkan respon status *error* karena ada parameter *viewport* yang melebihi jangkauan nilai latitude dan longtitude (lng1).

```
[
  {
   id_tmp: "3380",
   name: "KRI RIGEL",
   imo: null,
   mssi: "525014077",
   callsign: "PLJJ",
   flag: "Indonesia",
   length: "60",
   width: "12",
   height: "4",
   gross_tonnage: null,
   deadweight: null,
   alpha_2: "id",
   type: "5",
   type_name: "Tugs and Special Craft",
   lat: "-6.90529",
   lng: "110.437",
   speed: "1.2",
   status: "Arrived",
   time: "2015-12-02 05:22:03",
   last_known_port_id: "3688",
   last_known_port_name: "Tanjung Jati",
   atd_time: "2015-12-02 05:21:37",
   current_port_id: "3689",
   current_port_name: "SEMARANG",
   destination_id: "3689",
   destination_name: "SEMARANG",
   ata_time: "2015-12-02 05:22:03"
  },
  {
   id_tmp: "3380",
   name: "KRI RIGEL",
   imo: null,
   mssi: "525014077",
   callsign: "PLJJ",
   flag: "Indonesia",
   length: "60",
   width: "12",
   height: "4",
   gross_tonnage: null,
```

```
    deadweight: null,
    alpha_2: "id",
    type: "5",
    type_name: "Tugs and Special Craft",
    lat: "-6.55261",
    lng: "110.51",
    speed: "10.2",
    status: "Underway",
    time: "2015-12-02 05:21:42",
    last_known_port_id: "3688",
    last_known_port_name: "Tanjung Jati",
    atd_time: "2015-12-02 05:21:37",
    current_port_id: null,
    current_port_name: null,
    destination_id: "3689",
    destination_name: "SEMARANG",
    ata_time: null
  },
  {
   //Data kapal untuk riwayat sebelumnya
  }
]
```

Kode 4.2 Contoh *response* dari *request* data kapal untuk ID tertentu

Tabel 4.2 API *request* untuk GET data kapal berdasarkan ID

No	ID	*Response*
1	3380	Menampilkan data riwayat kapal ‘KRI Rigel’
2	KRI	Menghasilkan respon status error karena parameter id diisi string
3	null	Menghasilkan status error karena parameter id kosong
4	1536	Menampilkan data riwayat kapal ‘MV Suryawati’
5	2771	Menampilkan data riwayat kapal ‘KM DHARMA KENCANA 2’
6	1496	Menampilkan data riwayat kapal ‘Kartini Samudra’
7	4311	Menampilkan data riwayat kapal ‘Tangkas’

Tabel 4.3 API *request* untuk POST data riwayat kapal

No	Parameter	*Response*
1	ID = 4311 lat = -5.939 lng = 110.6518	Kapal dengan ID 4311 berubah posisi pada peta menjadi semakin ke arah timur laut.
2	ID = 3380 lat = -6.912 lng = 110.415	Kapal dengan ID 3380 berubah posisi pada peta menjadi semakin ke arah barat daya.

4.1.2 Pengujian API untuk Data Port

Pengujian API untuk data kapal dilakukan dengan melakukan *request* data pelabuhan untuk setiap parameter. Pengujian *request* GET data pelabuhan berdasarkan parameter *viewport* memberikan contoh *response* seperti ditunjukkan oleh kode 4.3. Hasil untuk pengujian ini selengkapnya ditunjukkan oleh tabel 4.4. Pengujian *request* GET data pelabuhan berdasarkan parameter ID memberikan contoh *response* seperti ditunjukkan oleh kode 4.4. Hasil untuk pengujian ini selengkapnya ditunjukkan oleh tabel 4.5. Sedangkan pengujian *request* GET data *portcall* memberikan contoh *response* seperti ditunjukkan oleh kode 4.5. Hasil untuk pengujian ini selengkapnya ditunjukkan oleh tabel 4.6.

```
[
 {
 id_tmp: "3688",
 port_name: "Tanjung Jati",
 country: "ID",
 unlocode: "",
 lat: "-6.4412",
 lng: "110.736",
 ships_in_port: "2",
 expected_arrivals: "1"
 },
 {...},
 {...}
]
```

Kode 4.3 Contoh *response* yang benar dari *request* data port berdasarkan parameter *viewport*

Tabel 4.4 API *request* untuk data kapal berdasarkan *viewport*

No	*Viewport*	*Response*
1	lat1 = -5.6350 lng1 = 114.7518 lat2 = -8.3609 lng2 = 107.2481	Menghasilkan respon 3 data port yaitu Tanjung Jati, Semarang dan Tuban.
2	lat1 = 2.0554 lng1 = 116.3256 lat2 = 0.4353 lng2 = 112.573	Menghasilkan respon *not found* untuk peta yang menampilkan daerah daratan di tengah pulau Kalimantan
3	lat1 = null lng1 = 104.7379 lat2 = 0.8567 lng2 = 102.8620	Menghasilkan respon status *error* karena ada parameter *viewport* yang kosong (lat1).
4	lat1 = 5.0825 lng1 = 121.5848 lat2 = abcd lng2 = 114.0811	Menghasilkan respon status *error* karena ada parameter *viewport* yang diisi string (lat2).
5	lat1 = 4.3654 lng1 = 190.6540 lat2 = -2.1117 lng2 = 115.6467	Menghasilkan respon status *error* karena ada parameter *viewport* yang melebihi jangkauan nilai latitude dan longtitude (lng1).

```
[
  {
   port_name: "SEMARANG",
   country: "ID",
   unlocode: "IDSRG",
   lat: "-6.93529",
   lng: "110.427",
   ships_in_port: "1",
   departures: null,
   arrivals: "1",
   expected_arrivals: "2",
   country_name: "Indonesia"
  }]
```

Kode 4.4 Contoh *response* yang benar dari *request* data port berdasarkan parameter ID

Tabel 4.5 API *request* untuk data pelabuhan berdasarkan ID

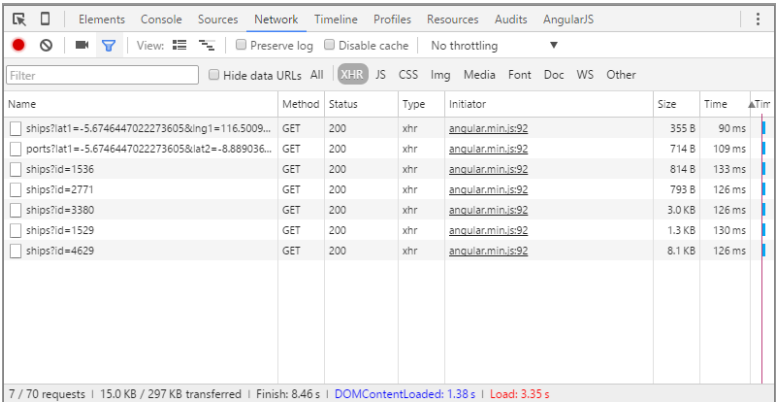
No	ID	*Response*
1	2955	Menampilkan data port ‘Jakarta’
2	Jakarta	Menghasilkan respon status error karena parameter id diisi string
3	null	Menghasilkan status error karena parameter id kosong
4	2917	Menampilkan data port ‘Sekupang’
5	2940	Menampilkan data port ‘Sabang’
6	2943	Menampilkan data port ‘Pangkalansusu’
7	3689	Menampilkan data port ‘Semarang’

```
[
  {
   ship_id: "1529",
   name: "KARTINI SAMUDRA",
   type: "1",
   alpha_2: "id",
   time: "2015-12-05 08:00:34"
  },
  {
   ship_id: "3380",
   name: "KRI RIGEL",
   type: "5",
   alpha_2: "id",
   time: "2015-12-02 05:21:30"
  },
  {
   ...
  }
]
```

Kode 4.5 Contoh *response* yang benar dari *request* data port berdasarkan parameter *portcall*

Tabel 4.6 API *request* untuk data pelabuhan berdasarkan *portcall*

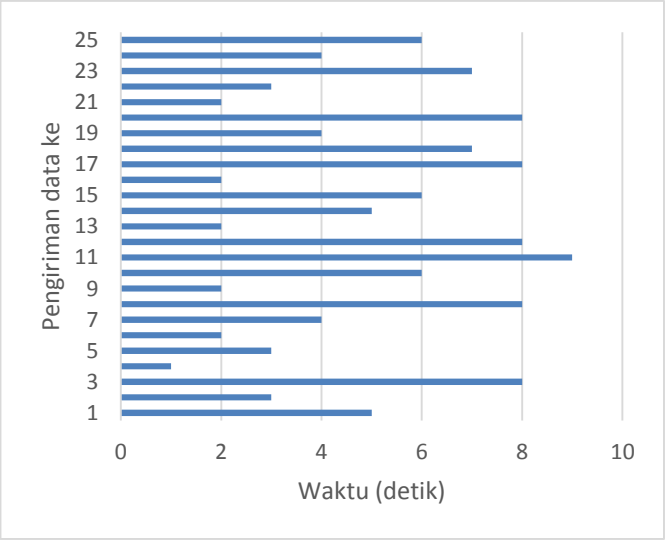
No	ID	*Portcall*	*Response*
1	2918	ship_in	Menampilkan respon 3 data kapal
2	3688	arrivals	Menampilkan respon 3 data kapal
3	2932	departures	Menampilkan respon 2 data kapal
4	2917	ship_in	Menampilkan respon 4 data kapal
5	2940	expected_arrivals	Menampilkan respon 4 data kapal
6	SMG	departures	Menghasilkan respon status *error* karena id diisi string
7	3007	kedatangan	Menghasilkan respon status *error* karena tidak ada parameter *portcall* dengan nilai ‘kedatangan’
8	3025	null	Menghasilkan respon status *error* karena parameter *portcall* kosong



Gambar 4.1 Besar data *response* dari API

4.2 Pengujian Visualisasi

Aplikasi ini didesain untuk mengambil data terbaru setiap 10 detik. Dengan begitu ada kemungkinan ketika data terbaru diterima, akan ada jeda penampilan data terbaru antara 1 – 10 detik. Pengujian ini dilakukan untuk mengetahui jeda rata-rata penampilan data terbaru pada peta daring. Pengujian dilakukan dengan melakukan simulasi POST data riwayat posisi kapal melalui API dengan mengubah-ubah data latitude dan longtitude kapal. Waktu yang dicatat adalah kapan data terbaru dikirim melalui API, kapan data terbaru diterima oleh client (*web browser*), dan kapan data terbaru ditampilkan pada peta. Hasil pengujian untuk 25 kali pengiriman data ditunjukkan pada gambar 4.2.



Gambar 4.1 Jeda waktu penampilan data terbaru

Halaman ini sengaja dikosongkan

BAB V
PENUTUP

5.1 Kesimpulan

Dari hasil implementasi dan pengujian dapat ditarik beberapa kesimpulan sebagai berikut :

1. REST API memberikan *response* seperti yang didesain untuk setiap *request.* Rata-rata data yang dimuat untuk menampilkan halaman adalah 15K dimana *request* data pelabuhan rata-rata hanya 714B sedangkan data kapal 4,13KB.
2. Terdapat jeda waktu dengan rata-rata 4,92 detik untuk *refresh rate* 10 detik dalam menampilkan data terbaru ketika ada data baru yang diterima.

5.2 Saran

Demi pengembangan lebih lanjut mengenai tugas akhir ini, disarankan beberapa langkah lanjutan sebagai berikut :

1. Untuk mengurangi jeda waktu dalam menampilkan data terbaru, aplikasi bisa diatur untuk melakukan muat ulang data terbaru secara otomatis menjadi lebih cepat dari 10 detik. Namun hal ini akan memberatkan pengolahan disisi *client* sehingga diperlukan optimasi dibagian proses penampilan *view*.
2. Antena yang digunakan pada sistem penerima menggunakan memiliki jangkauan penerimaan dengan tipe *omnidirectional* dimana hampir setengah area tersebut adalah daratan. Untuk memperbesar area visualisasi disarankan menggunakan antena dengan *reflector* yang dapat memperbesar area jangkauan ke arah laut dimana kapal berada.

Halaman ini sengaja dikosongkan

DAFTAR PUSTAKA

[1] Eddy Ketut. Pengembangan Intelligent Maritime Transportation System untuk Penegakan Kedaulatan Maritim Indonesia. Kompetitif Nasional – Pengembangan IPTEK, 2015, Indonesia.

[2] Shunfu Hu, Ting Dai, Online Map Application Development Using Google Maps API, SQL Database, and ASP.NET, ICT Journal, 2013 International Journal of Information and Communication Technology Research on, vol. 3, no. 3, Mar. 2013.

[3] Muzammil Khan, Sarwar Shah Khan, Data and Information Visualization Methods, and Interactive Mechanisms: A Survey, International Journal of Computer Applications (0975 – 8887) on, vol. 34, no. 1, Nov. 2011.

[4] Jianghui Ying, Denis Graþanin, Chang-Tien Lu, Web Visualization of Geo-Spatial Data using SVG and VRML/X3D, Proceedings of the Third International Conference on Image and Graphics (ICIG'04), 2004, Hong Kong.

[5] Udell Sterling, Beginning Google Maps Mashups with Mapplets, KML, and GeoRSS, Apress, 2009, New York.

[6] U.S. Coast Guard Navigation Center, 2015., AIS Messages. http://www.navcen.uscg.gov/?pageName=AISMessages. Diakses pada tanggal 23 Januari 2016.

[7] Pusat Data dan Informasi - Sekretariat Jenderal Kementerian Perhubungan - Republik Indonesia, 2010., Sistem Informasi Geografis Prasarana Transportasi. http://gis.dephub.go.id/mapping/Prasarana/PelabuhanList.aspx. Diakses pada tanggal 23 Januari 2016.

Halaman ini sengaja dikosongkan

BIOGRAFI PENULIS

Shidqon Famulaqih lahir di Pati pada tanggal 17 April 1993. Ia menyelesaikan pendidikan SD di SDN Dengkek 01 pada tahun 2005, kemudian melanjutkan pendidikan di SMP N 1 Pati menyelesaikannya pada tahun 2008, kemudian pendidikan SMA dilanjutkan di SMA N 2 Pati dan diselesaikan pada tahun 2011. Setelah lulus SMA, ia memilih untuk melanjutkan pendidikan di Institut Teknologi Sepuluh Nopember (ITS) Surabaya dengan jurusan Teknik Elektro dan mengambil konsentrasi bidang studi Teknik Komputer dan Telematika. Hal ini tidak lepas dari ketertarikannya dalam bidang teknologi. Selama menempuh pendidikan kuliah, penulis juga menjadi *freelancer* di bidang desain *website*. Penulis beberapa kali memberi pelatihan tentang *web blogging* di komunitas *web* Surabaya dan acara kampus.