D2D Vehicle Tracker LLD

# Requirement

<https://github.com/door2door-io/d2d-code-challenges/tree/master/fullstack>

# Solution at a glance

* RESTful API end points are created as per the requirement using express library for a Node js environment to listen to location update from Alligator Shuttles
* Mongo DB is being used to store every location updates received.
* Live visualization of Alligator shuttle locations is created,
  + the static contents(html, js, css, images etc) are delivered using Node.js
  + Live visualization App is merged with REST API App for easier deployment and can be decoupled.

# RESTful API

Code repo: <https://github.com/shakkirptb/vehicle-tracker>

## Dependencies

1. Server Env: Node.js v10.16.0
2. MongoDB v4.20

The RESTful API end-points are created using express library in for a Node.js server environment.

## Design and Pseudocode

### 1. Start of execution, index.js:

>”npm start” triggers <https://github.com/shakkirptb/vehicle-tracker/blob/master/index.js>

Express route (lib/route.js) to handle inbound requests is imported to Index.js and the API will listen to port provided in the properties file (properties.js). Default port: 3001.

### Properties,

Constance and default properties are stored at <https://github.com/shakkirptb/vehicle-tracker/blob/master/lib/properties.js>

### Express route

REST end-points for register, update-location unregister vehicles and get-updates (new) are configured at <https://github.com/shakkirptb/vehicle-tracker/blob/master/lib/route.js>

Data from the requests are passed to updateVehicleStats method for processing.

### Business logic

The object activeVehicles keeps track of all the vehicles to registered and running inside the city. And inactiveVehicles tracks all the vehicles outside the city, together they helps to figure-out vehicles that has to be accounted for visualization. This is to reduce response json size.

updateVehicleStats method determines active vehicles and inactive vehicles and updates the respective constants. Also each location update is passed to insert method of <https://github.com/shakkirptb/vehicle-tracker/blob/master/lib/database.js> for storing to db.

### Synchronization of data

Since 100s of request are received at every moment sending DB update requests of each DB wouldn’t increase network traffic as well as load on application. Hence, the data received by the insert method is added to a locationBuffer array and the data is inserted to DB in a bunch at regular intervals configured in properties file (default 5 sec). The timer will starts only on a request and it will wait for a fixed time, invokes insertBatch method, and clears the buffer.

### REST End-Point for Visualization

In addition to the end-points register, update, delete, a new end-point has been created that will be consumed by the visualization app in order to receive all vehicles’ update.

The express route code for this end-point sends activeVehicles objects as response

End point : “GET /locations”

Query params:

{"center": {"lat": 52.523, "lng": 13.424 }, "range": 3.5 }

Using this Query parameter, the visualization app can modify the center and the boundaries of the city to be visualized.

Sample response:

{"f616c6c8-33f2-4c98-9694-4abac3d46199":{"lat":52.53339,"lng":13.43917,"at":"2019-09-22T12:07:19.448Z"},"872e0bdb-4552-40b1-ba3a-9285b37659db":{"lat":52.55506,"lng":13.4148,"at":"2019-09-22T12:07:19.451Z"}}

# 2. Database

MongoDB is being used as DB for its rich query features, high availability and compatibility with JSON data. Given below are the DB details to be configured at properties.js file

MongoDB database name : d2d

Collection: locations

Sample document:

{

"vid" : "2e69d162-4f3e-489e-9239-c6090e3f0650",

"location" : {

"lat" : 52.43926,

"lng" : 13.40752,

"at" : "2019-09-22T11:57:08.852Z"

},

"action" : 3

}

Action represents the update given by a vehicle

{ UNREGISTER: 0, REGISTER: 1, UPDATE: 2, AWAY: 3 }

# 3. Visualization App

The code for visualization front-end app is merged with the REST API server as of now for the easiness of deployment and infra limitations.

URL : <http://domain/>

Demo: <http://d2d.shakkir.com/>

## Design and Pseudocode

Express route at <https://github.com/shakkirptb/vehicle-tracker/blob/master/lib/route.js> delivers index.html also exposes static content via <http://domain/> end-point.

External dependencies

1. JQuery
2. Google maps javaScript API
3. Google map’s Marker plugin
4. JQuery Easing plugin
5. Marker Animate plugin
6. Sliding Marker plugin

Google map’s API KEY and call back method (initMap) has to be mentioned at index.html while including google map’s plugin script.

Execution begin at <https://github.com/shakkirptb/vehicle-tracker/blob/master/web/res/main.js> as the map plugin invokes iniMap() method.

The method initialize maps object and tracker object (VehicleTracker constructor from <https://github.com/shakkirptb/vehicle-tracker/blob/master/web/res/tracker.js>)

Then invokes goLive() method.

goLive() invokes requestLocations() method immediately also starts timer to continuously invoke requestLocations() method for every 3 seconds (configurable at constants.js)