HiReach
innovative mobility solutions to cope with transport poverty

Documentation

Geofencing API

AUTHOR

Alexandru Coarna

September 2020

www.hireach-project.eu

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 769819.

TABLE OF CONTENTS

1.1 Introduction **1**

1.2 Technologies used **1**

1.2.1 Node.js *1*

1.2.2 Pm2 ..... *2*

1.2.3 Docker *2*

1.2.4 Typescript *2*

1.2.5 Tile38/Redis *2*

1.3 The geofence module **3**

1.3.1 Processing geofence data *3*

1.3.2 Geofence components and behaviour *4*

1.4 The push notification module **5**

1.4.1 Push notification configuration and behaviour *5*

1.4.2 Push notification messages *7*

1.5 API Configurable Parameters **7**

1.5.1 How it works *8*

LIST OF FIGURES

Figure 1. Aspect of a predefined route 3

Figure 2. Aspect of generated positions 3

Figure 3. Aspect of added radius areas 4

Figure 4. How the geofencing API works 8

1.1 Introduction

This is a NodeJs + Redis API designed to monitor travellers during a planned trip and check if the predefined route and all its aspects is respected and if something happens it will inform supervisors in real time about the issue.

The API needs a list of travellers to monitor alongside with route details for each one and optionally a list of supervisors for each traveller.

The API can detect and inform in real time, by using the push notification system, the following events:

- If the traveller does not respect the predefined route and goes off the specified road.
- If the traveller stays too much in the same spot.
- When a traveller reaches a specific location that was set in the route configuration.
- When a traveller does not reach a specific location at a specific time that was set in the route configuration.
- When a traveller reaches a specific location earlier than a specific time set in the route configuration and by how much earlier.
- When a traveller reaches a specific location later than a specific time set in the route configuration and by how much later.
- When a traveller initiates a new geofence session.
- When a traveller reaches the destination.

The API comes with a large list of configurable variables for each traveller, like controlling what notifications to be triggered, specific messages for each event with support for multiple languages or parameters for geofence accuracy. Also, it will return valuable information during the geofence session which can be used for tracking history, reports, or in-app custom notifications system.

1.2 Technologies used

1.2.1 Node.js

We choose Node.js because of its asynchronous behaviour and multi thread capabilities which improves the performance overall.

Asynchronous means that some actions like calling different external services, e.g. push servers, will not block the entire process and the result is returned way faster which leads to performance increase.

Usually calling external services will slow down the whole process because the performance is determined by the internet connection, server capabilities, the distance between your server and the 3rd party service, the optimization level of the 3rd party service etc.

Multi thread capabilities will let the Node application to take advantage of all the CPU core resources. If the server where the app runs has a multi core CPU then a separate instance of the application will start on each one of them. Basically, instead of one app, the server

will run multiple copies of it and that means more requests can be performed in a certain amount of time.

1.2.2 Pm2

This module is designed to start Node processes on a cluster mode, that means it runs a copy of the application for each core of the CPU on the server and all the processes run in parallel so they won't wait one for another.

1.2.3 Docker

Docker is a virtualization tool which wraps and isolates processes in so called containers. A docker container is an isolated process that runs independently to the server operating system and this is very helpful because you are able to run the same service on different platforms, you can instantiate an entire service in few minutes and you don't have to know how to manually set up everything because the docker container is already prepared with all the configuration.

Another useful thing is that docker will allow you to replicate an application, that means you can instantiate multiple copies of the application or stop them or migrate the whole application on another server within minutes.

1.2.4 Typescript

Typescript is an enchantment over Node.js, it can be qualified as a language, but its main role is to define models and interfaces for all the JavaScript code.

JavaScript does not support types natively so here is where Typescript comes in. This is very helpful for future changes in the project as the API is open source and anyone can change it, so Typescript will make it easier. It will be easier to understand the code and algorithms used in the API because you know exactly what function requires what parameters and the result type. This way is easier to modify the functionality without breaking anything that normally you will not take in consideration with plain JavaScript.

1.2.5 Tile38/Redis

This is a tool written in GO which uses Redis as storage and its main purpose is the geofence itself. Using this tool, the API can perform checks on target position relative to other positions / surfaces (https://tile38.com).

Redis is an In-memory Database and by default is used by the Tile38 API and because of this the entries passed to Tile38 must respect a certain format and will prevent invalid structures to be inserted in the database.

The main advantage of Redis is that it is In-memory. That means all the data is directly available in RAM memory which sometimes is 10x faster than the normal flash memory (storage SSD/HDD) and also it is persistent, the data being backed up in the device flash memory.

1.3 The geofence module

The minimum required data to start a geofence session is a Target with an id and a predefined road composed of a list of geolocations, pairs of latitude-longitude coordinates, all the other things being optional.

A complete geofence session is composed of the following:

- A predefined route.
- A list with places of interest, some custom areas.
- A list with places of interest alongside with specified arrival time.
- A list of configurable parameters that defines the geofence details. If this is not provided, then the API will use the default configuration.

1.3.1 Processing geofence data

The predefined route is a collection of linear geolocation positions in which two successive points can be linked with a straight line.

The distance between two consecutive points is not equal, that distance may be determined by many factors and in a slightly different way by using different services like Google Maps or Open Street Maps.

Figure 1. Aspect of a predefined route

Because of the difference in distance between points, the API will take that list of geolocation points and will generate additional geolocation positions at an equal distance one from another between each 2 consecutive points already existing in the list. If the distance between two already existing consecutive points is equal or less than the distance set in the configuration file or the distance defined for that specific target, those two points are automatically skipped, and nothing is generated between them.

Figure 2. Aspect of generated positions

Now the distance between each 2 consecutive points is equal or less than the defined distance during the processing stage. The distance between points can be set in the API configuration file and can be set differently from a target to another. The API will let the user choose the distance because depending on the type of road it might be higher or smaller.

The final step of data processing is to add a radius area around each point in the predefined route. The radius area value is configurable for each target and can be specified in the API configuration file. This is a kind of geolocation augmentation where a list of linear geolocation points will define in the end a road or street whose thickness is determined by the radius area around each point, which creates a border around the road.

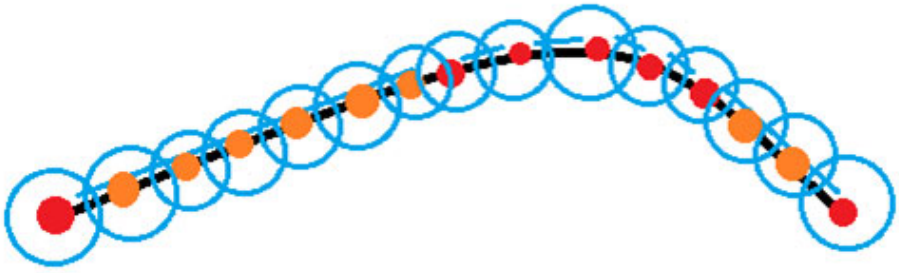

Figure 3. Aspect of added radius areas

For best accuracy, each two consecutive areas MUST intersect, and the distance between two consecutive points MUST be at maximum half of the radius value, and that will ensure the radius intersection. Also, the minimum radius values in meters is 20 and the maximum one is 150 and therefore the minimum distance between 2 consecutive points is 10 and the maximum one is 75.

1.3.2 Geofence components and behaviour

When a new session is started, the API puts all the data in the database and if another request is made to modify the fence specific data while the fence session is on, the API will throw an error as those data can't be changed while the fence is on. The current session must be stopped which means the tool will reset all fence and temporary data and then you can start another fence session with new data.

Beside the predefined route, points of interests can be defined, like subway stations or bus stations etc. These points of interests are known as Custom Areas in the API.

A custom aria can be a simple location which is determined by the configuration file in API. When a target reaches such a location an event occurs about the fact that the target reached a point of interest.

A custom area can also be linked to a time frame in which the target should reach that area, this is called Timetable Custom Area. The API will detect 3 different events for this kind of custom area.

- When a target reaches the area earlier and by how much time.
- When a target did not reach the area in the specified time frame and it should have.
- When a target finally reached the area but later and by how much time.

The time frame defined for a Timetable Custom Area is composed of a specific arrival time + an extra time defined in minutes. By default, the extra time is set to 10 minutes in the

configuration of the API, but this can be changed and set differently from a target to another.

All the fence values, the predefined route and all points of interest are validated by the tool to make sure that the correct distance between points and radius value is set and also to check if all interest points are located on the predefined route and there is nothing outside the fence area.

All this data is set, processed and validated when a new geofence session starts, then, whenever the target location is updated, that location must be fed to the API and it will perform all the checks based on targets current location.

The API will detect when and if a point of interest is reached but also it detects if the target is not anymore on the specified route and this check is performed with a retry system and also the notifications will be sent in a specified interval of time.

The retry system is necessary because sometimes mobile devices may send invalid locations and we must be sure that this is not an error and if the target will send a location outside the predefined route multiple times in a row then we know something is really wrong.

Both the retry count number and time interval can be set differently from a target to another and by default the API will set them to 5 tries and respectively 3 minutes. If a target is off the fence area and a notification is sent, then for another 3 minutes the notification won't be sent anymore.

The API detects a new geofence session and will send a notification related to it and also when the target reaches the destination. At this point the fence is automatically stopped and all temporary data are cleaned from the database to keep a low storage usage.

When a target advances on the route, the positions behind will be removed from the list, basically if the target will turn around and goes backwards on the route the API will detect that the target is off the predefined route, because the already travelled route is no longer considered valid.

If a target stays in the same location for more than a specified time, which can be defined in the API configuration or separately for each target, the API will detect this and will send a notification accordingly.

1.4 The push notification module

The API can send push notifications for the most used platforms and operating systems, for Web based applications, Android / IOS based applications, any mobile or fixed device.

1.4.1 Push notification configuration and behaviour

To cover most used operating systems and platforms, the API will use both standard Web Push Notifications, which is a service provided by the browser, and Firebase Cloud Messaging API.

The credentials for both push systems can be set in the API configuration file. Note that each application has its own credentials.

This module is optional because the API will also return information about events, if any occurred or not, on each request. This information can be further used for custom in-app notification or messaging, reports, tracking history etc.

The API will check for any of those push system methods and it will try to initiate push notifications only if the method was set, otherwise the push module will be inactive.

Each target may have multiple supervisors. A supervisor is basically defined by an id and a subscription. The subscription can be for either Web Push or Firebase or for both. Also, each supervisor may have multiple targets to supervise, the relation between target and supervisor is a many to many. The API will let a supervisor change its status, that means you can choose to be a supervisor or not and then become a supervisor again.

If there is no supervisor set for a target, the notification is simply skipped but the returned result will still contain information about the event if any occurred.

The API will let supervisors to choose if some of the events that occurred should or should not trigger a notification.

In the API configuration there are 4 events that may skip the notification part:

- New fence session Event.
- Reached destination Event.
- Late arrival for a point of interest Event.
- Early arrival for a point of interest Event.
- Target stays too much in the same spot event.

The other 3 main events cannot be skipped:

- When a target goes off fence.
- When a target reached a point of interest that does not define a time frame (Custom Area).
- When a target did not reach a point of interest in the defined time frame (Timetable Custom Area).

If an event occurs for a point of interest twice because the targets position may be updated in each second, that means the tool will detect that the target is in the same area of interest 5 times in 5 seconds. After the first notification there will not be another one as that point of interest is marked as notified and the tool will skip checks for already verified and notified points of interest.

If a target will arrive earlier than the specified time frame in a point of interest, there will not be a check anymore for no arrival event because basically the point was reached and furthermore it was earlier than expected.

If any of the skippable notifications are set to false so the supervisor chooses not to be notified about those events, the API will totally skip checks and this way it leads to a performance improvement. Also, the API will not detect the event at all and neither the standard request response will not contain information related to the event.

The notification module is totally asynchronous because the geofence module should not be slowed down by it.

If an error occurs while sending a notification this might disturb the entire process, so therefore these two modules are independent of each other.

Because of the asynchronous behaviour of the notification module, all the errors that might occur inside it are logged in files on the server. The API will not put those logs in the Redis database because this will lead to irrelevant usage of RAM memory, as Redis is an In-memory database. The files are generated for each day and they can be accessed either one at a time by defining the file name in the request, name which is composed of the current date in format day-month-year (e.g. 1-12-2020 / 10-1-2020), or they can be accessed all at once and the result will be merged from all the files.

1.4.2 Push notification messages

The API has a dedicated file in which messages for any event may be defined and also the messages can be defined for multiple languages.

The API has as a default setting the English (en) language, but this can be changed for each target in the configuration, or for all targets in the configuration file.

The API checks if there are messages set for English language and also for the language set in the default config, if this is different from English.

If the target has defined a language which doesn't have any defined message, the tool will automatically fallback to the default language (en).

The message system provides a list of placeholders which may be used while defining messages for each event, placeholders which are replaced with important values generated or detected by the API during the check process:

- Target name or identifier.
- The point of interest name or coordinates.
- The difference in time for late and early arrival.
- The time for how much the target stood in the same spot.

1.5 API Configurable Parameters

The default configuration can be modified, and the API offers an option to set those parameters for each target specifically.

If a target does not have a custom configuration then the default values are used. The tool will merge the default values with the target's custom values because the tool will let you set only specific parameters or multiple specific parameters or all of them but they are optional, so any can be skipped.

The target custom configuration can be changed anytime even if a geofence session is on or not, through the dedicated endpoint. The API will validate all the parameters to ensure a good functionality.

List of parameters/options that can be configured:

- The extra amount of time for the Timetable Custom Areas if this is not defined for each area when the fence session starts.
- The interval in minutes for how often the off-route notification should be sent.
- The retry count for the off-route check.

- The radius value that is generated for each position in the route position list.
- The distance between 2 consecutive points in the route position list.
- The radius value for each point of interest area.
- The default language for notification messages.
- A target alias / name; if this is not set, the target id is used in messages.
- The amount of time for how long a target should stay in the same spot, for the same location event to occur.
- If there should be a notification when a new fence session starts.
- If there should be a notification when the target reaches the destination.
- If there should be a notification when the target reaches a point of interest earlier than the specified time frame.
- If there should be a notification when the target reaches a point of interest later than the specified time frame.
- If there should be a notification when the target stays too much time in the same spot.

1.5.1 How it works

This API should be used as a micro service, a separate dedicated service for geofence in addition for other mobility / tracking applications. The API should be fed with data generated by the mobility Indie App and the target location must be updated frequently (sent by the Indie app to the API) so the API can detect any event that may occur.

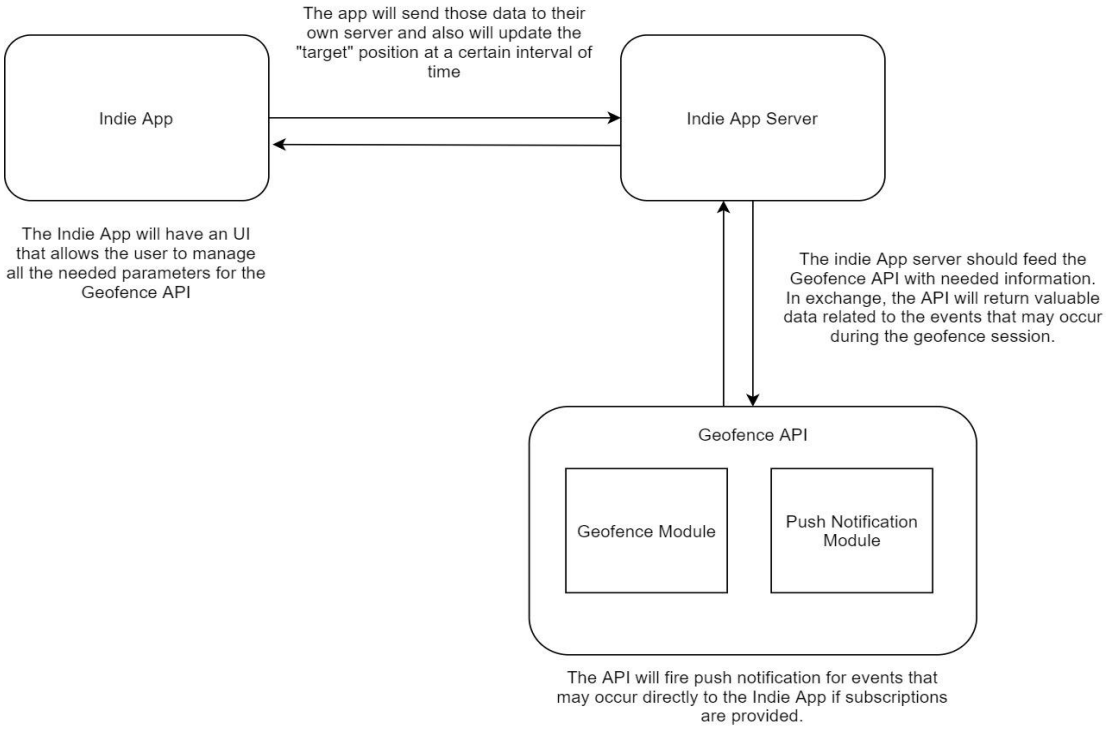

Figure 4. How the geofencing API works