Mobile Nanny v1.0 Report

Tutor: Pr. Marc SPANIOL

Abstract

This project deals with development of operating system that track mobile phones and visualize trails on web interface.
It's based on web platform and mobile platform that implement on the one hand, mobile registration and location track; and on the other hand, runtime perception of moving on personalized map.

This work is a concept for a Master degree in DNR2i computerization, located in Caen.

Document describes technologies in use which forge basic design of system. Furthermore, it is argued with a few specifications how the system works.

Synopsis

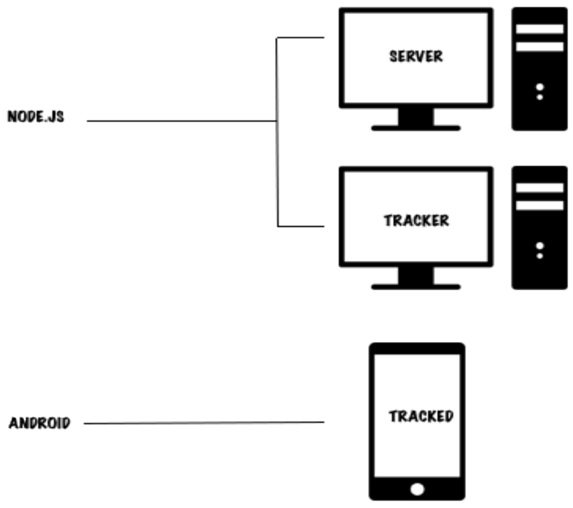
Getting Started

It's within the scope of my studies aiming at becoming a full-stack[1] web developer that I implement this teaching project named Mobile Nanny. This project is being entrusted to me on behalf M. Marc SPANIOL, teacher-researcher at the University of Caen in Normandy. Development phase was three months and I worked by myself.

The project involves carrying out a two-phase development plan. With respect to the first objective, it is an establishment of web server using Node.js[2], an easy to use Javascript[3] completion to perform some behaviors as API[4].
In second part, it is a question of offer intuitive graphical interfaces for web explorer and mobile phones.

System Requirements

In context of the study, macOS[5] and Android[6] are supported host development operating systems. For lack of resources, Node.js is used to achieve a more efficient use of instant message rather than Apache[7] server. Despite Android platform, iOS integration can be accomplished with ease.



[1]*full-stack: familiarity in layers (server, data modeling, user interface).*
[2]*Node.js: asynchronous event driven designed to build network applications.*
[3]*Javascript: interpreted programming language.*
[4]*API: Application Programming Language.*
[5]*macOS: Operating system marketed by Apple.*
[6]*Android: Google's mobile operating system.*
[7]*Apache: web server software.*

Server Deployment and Launching

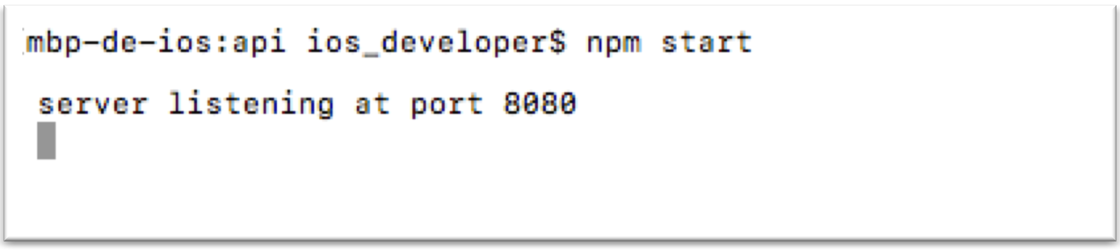
Server is a standalone shareable application. Due to several dependencies used on, npm[1] must be called to check modules updates before starting application. The best way to manage packages is to create a *package.json* file (@see documentation) and list all packages on ‘dependencies’ node.

```
{
        "name": "mobile_nanny",
        "dependencies": {
             "express": "4.12.x",
             "ejs": "2.5.x",

        }
```

When running **npm install** on terminal, npm looks at dependencies that are listed in *package.json* and downloads latest versions.

If all packages are updated, then we can run **npm start** on terminal to launch server.



```
mbp-de-ios:api ios_developer$ npm start

 server listening at port 8080
```

[1]*npm: Node Package Manager*

API Functionality

The following section explain more about operation that can be performed using API.

API Overview

Here are some resources that will help to understand basics of APIs. Major requests encountered are used with HTTP POST request as following:

```
POST <host>/<entities>/<method>/
{
        "key": <value>,
        ...
}
```

There are three “entities” in API: **Users**, **Phones** and **Locations**. Methods defined are only “**add**”, “**remove**” are “**list**”.

Request parameters are stored on JSON format.

Response status codes are defined on *config.js*. In most cases, response status code is as 6XX. Otherwise, error is unknown. Fortunately, a description of response is always available on JSON format.

User Properties

To reach tracking, an account is required. An HTTP POST request can be used as follow to registry credentials as user representation:

```
POST <host>/users/add/
{
        "email": <email>,
        "password": <password>
}
```

Note that if response code is 600, credentials are already being used. Available error describes reason of failure.

Phone Registration

A tracked phone is pinpointed by a generated token. Token is composed by email credential and a limited period expiring on seven days since registration requesting:

```
POST <host>/phones/add/
{
        "email": <email>,
        "password": <password>
}
```

Result of request is a JSON format response contained generated token.

Note that phone should be unregistered if token is remove using request as following:

```
POST <host>/phones/remove/
{
        "email": <email>,
        “password”: <password>,
        "token": <token>
}
```

Quite apart from that, it’s questionable whether there is a way to list phones tracked by identified tracker. All to have to do is:

```
POST <host>/phones/list/
{
        "email": <email>,
        "password": <password>
}
```

Notice that response is representative HTML format of phones.

Location Spying

This part shall apply to mobile phones. It consists of coordinates collecting as latitude and longitude.

How to recover phone latitude and longitude? On Android platform, we implement android.location.LocationListener class to retrieve location through onLocationChanged(Location location) method (@see LocationService).

To add location, use token registered on phone registration while adding phone:

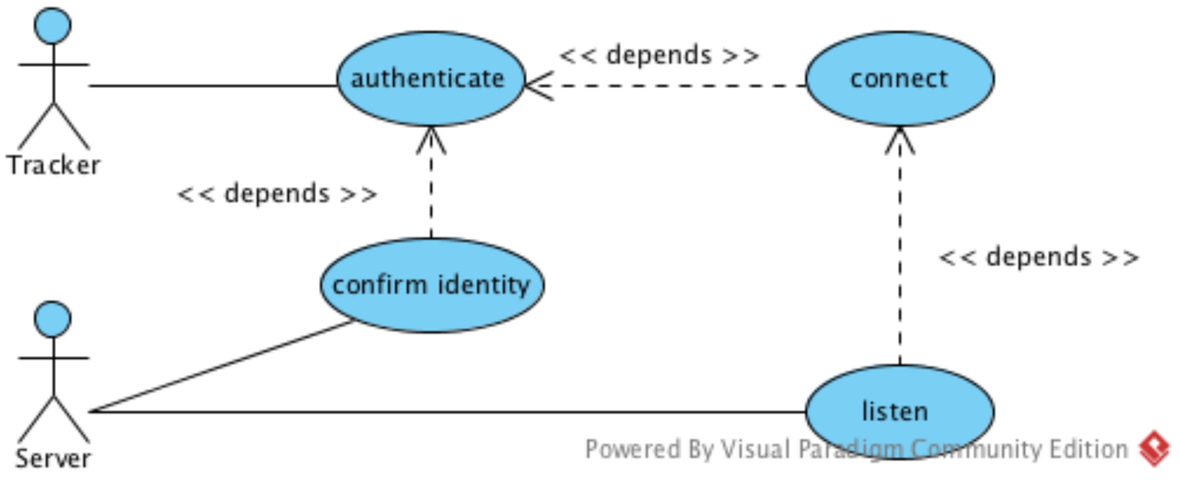
```
POST <host>/locations/add/
{
        "token": <token>,
        "latitude": <latitude>,
        "longitude": longitude>
}
```

As planned for phones, we can list locations about token:

```
POST <host>/locations/list/
{
        "token": <token>
}
```

Main Aspects

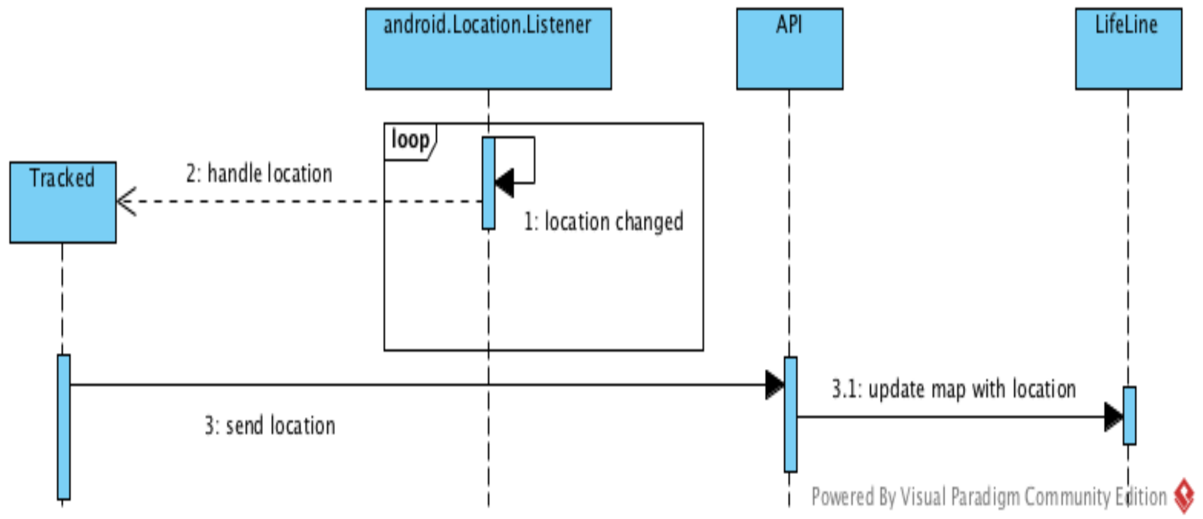
Authentication



When a tracker gets onto web interface at address <host>:8080, server is seeking to establish a “peer to peer” connection with explorer using sockets (@see Socket.io).

While tracker socket is connected, authentication process starts. Tracker emits credentials to server; the latter shall check data concerned. If identity exists, tracker has been allowed access to use API services (@see authentication).

Location Upgrading


android.Location.Listener
API
LifeLine
Tracked
loop
2: handle location
1: location changed
3: send location
3.1: update map with location
Powered By Visual Paradigm Community Edition

Location manager runs on background to loop on location retrieving. When a location is available, application will capture it and send it through API (@see LocationService).

If tracker is authenticated, server prevents tracker that location is available (@see update location).

Reviews

Results are mixed:
On the one hand, I produced all reasonable requests. As continuous process to update location of mobile phones or simultaneous display track on map we can enhance authentication.
On the other hand, because of lack of time, there are no alerts to prevent thresholds are missed. Moreover, we can enrich GUI.

Special Thanks

I would like to thank M. Marc SPANIOL offering me opportunity to highlight my mobile-developer skills and by what I learnt courses.

Annexes

Please, look at online project to study source code or to share your comments.