**hartapoluarebrasov**

Harta Poluare Brasov - Proiect Licenta UnitBV - IESC – Calculatoare – Iancu George-Alexandru

# BACKEND SIDE

# Harta Poluare Brasov Middleware Server

### Features

* Rest API for searching in AWS - ElasticSearch

### Modules

HartaPoluareBrasov uses a number of open source projects to work properly:

* [aws-sdk] - Amazon web services SDK for nodeJS
* [cors] - CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.
* [dotenv] - Dotenv is a zero-dependency module that loads environment variables from a .env file into process.env.
* [elasticsearch] - The official Node.js client for Elasticsearch.
* [express] - evented I/O for the backend
* [cookie-parser] - Parse Cookie header and populate req.cookies with an object keyed by the cookie names. Optionally you may enable signed cookie support by passing a secret string, which assigns req.secret so it may be used by other middleware.
* [jsonwebtoken] - This was developed against draft-ietf-oauth-json-web-token-08. It makes use of node-jws
* [bcryptjs] - Optimized bcrypt in JavaScript with zero dependencies. Compatible to the C++ bcrypt binding on node.js and also working in the browser.

### Installation

Actual developments for HartaPoluareBrasov requires

* [Node.js](https://nodejs.org/) v12.13.1
* [NPM](https://www.npmjs.com/get-npm) v6.12.1

Create the ".env" file in the /server folder:

AWS\_USER=<USER FOR AWS>

AWS\_PASS=<PASS FOR USER>

DEV\_PATH=https://localhost:3001

JWT\_SECRET=<JWT SECRET PASSWORD>

COOKIE\_SECRET=<COOKIE SECRET PASSWORD>

Install the modules and start the server.

$ git clone https://github.com/georgealexandruiancu/hartapoluarebrasov

$ cd server

$ npm install --save

$ npm start

# API

## ROUTES

### 1. /users/<params> (users.js)

### 2. /data/<params> (data.js)

# 1.1. Create Users Tabel -- just once

Method: GET

/users/create-users-index

-- Add in ElasticSearch tabel with index: "users"

# 1.2. Add Users

Method: POST

/users/add-user

* 1. Body Object to Post:

{

"name": <STRING>,

"email": <STRING>,

"password": <STRING>

}

-- Add in "users" table an user. -- This method will create the user and the device table for the user

# 1.3. Login Users

Method: POST

/users/login

* 1. Body Object to Post:

{

"email": <STRING>,

"password": <STRING>

}

-- This will interogate the ElastiSearch DB and find and match if user exists. -- Will compare using bycrypt the password from request and the password from DB -- If all is OK, the method will create a signed token using jwt and send to a cookie

# 1.4. Logout

Method: GET

/users/logout

-- This will clear the cookies created in login part.

# 1.5. Who am I ?

Method: GET

/users/who-am-i

-- Recieve the user data if is logged in. -- It's a test method

# 1.6. Add data to User

Method: POST \*\* AUTH REQUIRED \*\*

/users/add-data-user/<hash>

-- 1. it's required (this will be the id from device)

* Body Object to Post:

{

"MQ135": {

"value": <NUMBER>

},

"PM25": {

"dustDensity": <NUMBER>,

"value": <NUMBER>

},

"gpslocation": {

"location": {

"lat": <NUMBER>,

"lon": <NUMBER>

}

},

"humidity": <NUMBER>,

"name": <TEXT>,

"temperature": <NUMBER>

}

# 1.7. Get Data from a user

Method: GET

/get-user-data/<hash>

* 1. Recieve data from user with id

# 2.1. Recive all data

Method: GET

/data/get-all

-- Return all data from DB elasticsearch with limit of 1000 entries

# 2.2. Recive specific data from sensors

Method: GET

/data/get-all/<sensor>/<limit>

* 1. If we specific only the , api will get us the 1000 entries from that sensor <sensor: mq135 || mq1>
  2. If we specific the with we will have the data from sensor with a limit for entries <limit: 0 -> 10000>
* E.g.: /get-all/mq135/100

# 2.3. Get Air Quality from radius point

Method: GET

/data/get-data-by-radius/<radius>/<lat>/<lng>/<hashUser>

* 1. Recieve all data from a checkpoint radius
  2. <radius> - MANDATORY - eg: 1km, 100m, 100km, 1000miles - string
  3. <lat> - MANDATORY -point latitude - float
  4. <lng> - MANDATORY - point longitude - float
  5. <hashUser> - OPTIONAL - user device hash
  6. ?limit=<number> - PARAMETER - recieve max data from 0 to - number

# 2.4. Add Data to Sensor

Method: POST

/data/post-data/<sensor>

* 1. ? "MQ135" : "PM2.5"
  2. Body Object to Post:

{

"MQ135": {

"type": "long"

},

"heatIndex": {

"type": "float"

},

"humidity": {

"type": "float"

},

"name": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword",

"ignore\_above": 256

}

}

},

"query": {

"properties": {

"match\_all": {

"type": "object"

}

}

},

"quote": {

"type": "text",

"fields": {

"keyword": {

"type": "keyword",

"ignore\_above": 256

}

}

},

"temperature": {

"type": "float"

},

"timestamp": {

"type": "date"

}

}

# 2.5. Get All Data from all users

Method: GET

/data/get-all-data?limit=<NUMBER>

* 1. Recieve data from all users
  2. ?limit= - is an optional query

### TODO REMAINING

* get data with filters
* make users active on platform to setting a index: PENDING / ACTIVE
* create a sys for mailing
* update users confidentials eg. pass, name
* ~~get data from range of time~~
* ~~get data only for a day~~
* ~~get data for charts~~
* create administrator database to update users

# FRONTEND SIDE

In the project directory, you can run:

### npm install --save

### npm start

Runs the app in the development mode.  
Open [http://localhost:3000](http://localhost:3000/) to view it in the browser.

**Proiectul a fost creat pentru react-create-app (react.js)**React este o bibliotecă JavaScript open-source pentru construirea de interfețe de utilizator. Este întreținută de Facebook și de o comunitate de dezvoltatori și companii individuale. React poate fi folosită ca bază pentru dezvoltarea aplicațiilor mobile cu o singură pagină sau mobile.

**Components:**

*- Login:*

a. UserLogin.js – autentificarea userilor

b. UserRegister.js – inregistrarea userilor

c. UserRegisterConfirmation.js – confirmarea userului  
  
*- BottomCharts.js* – include partea de tabele si vizualizari din partea jos a site-ului

*- Header.js* – include partea de navigatie si informatii despre aplicatie

*- SideLeft.js* – meniul principal pentru utilizator

*- MapContainer.js* – include harta google

*- Middleware*

a. Marker.js – obiect ce contine pin-ul din hartile google si heats map

b.\_mapStyles.js – obiect cu stilizari google

c. Chart.js – obiectul (templateul) pentru tabele

**Functions:**

*- Iancu.openModal.js  
- Iancu.spalshScreen.js  
- Iancu.switchTabs.js  
- Iancu.toggle.js*

**Styles – s-a folosit metodologia BEM si post-procesorul de CSS numit SCSS**

**App.js – este root-ul aplicatiei**

**Folosire javascript: ES6**

**Proiectul foloseste clase si obiecte de JS compilate prin webpack (folosind react.js)**