Geolocalização em aplicações Web com Node.js

1 - Criando pasta para projeto “node-maps”

2 - Criando arquivo gerenciador de dependências : npm init

3 -Instalando as principais bibliotecas para nosso projeto

npm install –save restify knex mysql

ARQUIVO DE ENTRADA DO SERVIDOR

4 - Criando arquivo index.js

Principais configurações do RESTIFY.

```
const restify = require("restify")


const server = restify.createServer({
    name: "myapp",
    version: "1.0.0"
})
server.use(restify.plugins.acceptParser(server.acceptable))
server.use(restify.plugins.queryParser())
server.use(restify.plugins.bodyParser())


server.listen(8081, function(){
    console.log("%s listening at %s", server.name, server.url)
})
```

5 - Configurando a rota principal

```
server.get("/all", function (req,res,next){
})
```

6 - Configurando o banco de dados com Knex

```
const knex = require('knex')({
    client:'mysql',
    connection: {
        host:'127.0.0.1',
        user: 'root',
        password: '',
        database:'maps'
    }
})
```

7 – Refatorando a rota */all*

```
server.get("/all", function (req,res,next){
    knex('places').then((dados) => {
        res.send(dados)
    }, next)
    return next()
})
```

8 - Criar Banco de dados e a tabela Places

```
CREATE DATABASE maps;
use maps;
CREATE TABLE places(
   id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
   place_id VARCHAR(30),
   address TEXT,
   image TEXT
)
```

9 - Configurações API do Google para serviços de geolocalização

http://console.developers.google.com/

https://github.com/googlemaps/google-maps-services-js

10 - Instalar minha dependência:

```
npm install @google/maps –save
```

11 - configurando a api em seu back-end

```
const googleMapsClient = require('@google/maps').createClient({
    key: 'sua chave api aqui',
    Promise: Promise
  });
```

12 - Nova rota /geocode

```
server.get("/geocode", function (req,res,next){
    googleMapsClient.geocode({address: '1600 Amphitheatre Parkway, Mountain View, CA'}).asPromise()
    .then((response) => {
        res.send(response.json.results)
    })
    .catch((err) => {
        res.send(err)
    })

})
```

13 – Refatorando um pouco

```
    server.get("/geocode", function (req,res,next){

        googleMapsClient.geocode({address: '1600 Amphitheatre Parkway, Mountain
View, CA'}).asPromise()

        .then((response) => {

            const address = response.json.results[0].formatted_address

            const place_id = response.json.results[0].place_id



            res.send({place_id, address})

        })

        .catch((err) => {

            res.send(err)

        })



    })
```

14 – Codificando a Reverse Geocode

```
    server.post("/geocode", function (req,res,next){

        const {lat, lng} = req.body

        googleMapsClient.reverseGeocode({latlng: [lat, lng]}).asPromise()

        .then((response) => {

            const address = response.json.results[0].formatted_address

            const place_id = response.json.results[0].place_id



            res.send({place_id, address})

        })

        .catch((err) => {

            res.send(err)

        })

    })
```

15 - Criando imagens de Mapa a partir de geolocation

https://developers.google.com/maps/documentation/maps-static/intro

exemplo de coordenadas: lat: -24.0347898 lng: -52.3719842

https://maps.googleapis.com/maps/api/staticmap?center=-24.0347898,-52.3719842&zoom=13&size=300x300&sensor=false&key=AIzaSyA9jJUtPhlLRpfItW2xpbEACA_QrpIMiFw

```
server.post("/geocode", function (req,res,next){
    const {lat, lng} = req.body
    googleMapsClient.reverseGeocode({latlng: [lat, lng]}).asPromise()
    .then((response) => {
        const address = response.json.results[0].formatted_address
        const place_id = response.json.results[0].place_id

        const image =
`https://maps.googleapis.com/maps/api/staticmap?center=${lat},${lng}&zoom=13&s
ize=300x300&sensor=false&key=AIzaSyA9jJUtPhlLRpfItW2xpbEACA_QrpIMiFw`

        res.send({place_id, address, image})
    })
    .catch((err) => {
        res.send(err)
    })

})
```

16 – Salvando as informações no banco de dados

```
server.post("/geocode", function (req,res,next){
    const {lat, lng} = req.body
    googleMapsClient.reverseGeocode({latlng: [lat, lng]}).asPromise()
    .then((response) => {
        const address = response.json.results[0].formatted_address
        const place_id = response.json.results[0].place_id

        const image =
`https://maps.googleapis.com/maps/api/staticmap?center=${lat},${lng}&zoom=13&s
ize=300x300&sensor=false&key=AIzaSyA9jJUtPhlLRpfItW2xpbEACA_QrpIMiFw`

        knex('places')
        .insert({place_id, address, image})
        .then(()=>{
            res.send({address,image})
        }, next)

    })
    .catch((err) => {
        res.send(err)
    })

})
```

17 - Configurando o Front-End de nossa aplicação

Estrutura de pasta

- ./dist
  - assets/
    - app.js
    - style.css
  - index.html

18 – Existe um mundo sem bootstrap

Material Design https://getmdl.io

19 – Não podemos esquecer o Axios

```
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
    <link rel="stylesheet" href="https://code.getmdl.io/1.3.0/material.indigo-
pink.min.css">
    <script defer src="https://code.getmdl.io/1.3.0/material.min.js"></script>
    <script src="https://unpkg.com/axios/dist/axios.min.js"></script>
    <link rel="stylesheet" href="assets/style.css">
    <title>App My Location</title>
</head>
<body>


    <script src="assets/app.js"></script>
</body>
</html>
```

20 - Criando rota padrão para acesso de arquivos estáticos

Obs: Criando sempre no final!

```
server.get('*/',restify.plugins.serveStatic({
    directory: './dist',
    default: 'index.html',
  }));
```

21 – Voltando para o Material Design Lite

- Componentes → Cards
- Refatorando o Html e CSS do card
- Sistema de Grid
- Botao ADD

```
<!-- Colored FAB button with ripple -->
<button class="mdl-button mdl-js-button mdl-button--fab mdl-js-ripple-effect mdl-
button--colored">
  <i class="material-icons">add</i>
</button>
```

22 - Refatorando o botão add: criando estilo alterando a cor do botão para vermelho e que posição dele seja sempre no canto inferior direito.

```
.add {
      position: fixed;
      right: 40px;
      bottom: 25px;
      background: red !important;
      z-index: 3;
  }
```

23 - Adicionando Spinner, para nossa aplicação realizar o comando leva algum tempo, sendo necessário o usuário esperar, sendo assim é recomendo retornar algo para o usuário para ele saber que esta sendo processado.

```
<!-- MDL Spinner Component -->
<div class="mdl-spinner mdl-js-spinner is-active"></div>
```

24 - Adicionando a classe spinner-position

```
.spinner-position {
      position: fixed;
      right: 55px;
      bottom: 100px;
      z-index: 3;
  }
```

25 - Adicionando ID's para os elementos que vamos manipular com javascripts.

Grid, spinner e send.

Agora é no javascript...

Primeira parte do código

```
window.onload = () => {
    let grid = document.querySelector('#grid')
    let button = document.querySelector('#send')
```

```
    read()
}

function read() {
    axios
    .get('/all')
    .then(response => {

    })
    .catch(error => {

    })
}
```

26 – Refatorando a function read()

```
function read() {
    axios
    .get('/all')
    .then(response => {
        response.data.forEach(element => {
            let card = templateCard(element.address, element.image)
            grid.innerHTML += card
        })

    })
    .catch(error => {

    })
}
```

27 – Criando a function templateCard

```
function templateCard(address, image){
    return `
        <div class="demo-card-wide mdl-card mdl-shadow--2dp">
            <div class="mdl-card__title">
                <img src="${image}" alt="">
            </div>
            <div class="mdl-card__supporting-text">
            ${address}
            </div>
        </div>
    `
}
```

28 - Salvando novos dados de localização

```
function save(){
    if(!navigator.geolocation){
        alert("Seu browser não suporta a geolocalização! </p>")
        return
    }

    navigator.geolocation.getCurrentPosition(success, error, { //recebe por
paramentro duas funções de call-back a success e a error
        enableHighAccuracy: true //melhora a precisao, mas demora um pouco
mais para carregar.
    })
```

```
    function success(position){
        const lat = position.coords.latitude
        const lng = position.coords.longitude

        axios
            .post("/geocode",{lat, lng})
            .then(function(response){
                let card = templateCard(response.data.address,
response.data.image)
                grid.innerHTML +=
            })
            .catch(function(error){

            })
    }
    function error(err){
        alert(err)
    }

}
```

29 - Criando evento para botão add

```
button.addEventListener('click', save)
```

30 - loading ...

- remover class is-active
- Refatorando save()

```
function success(position){
        const lat = position.coords.latitude
        const lng = position.coords.longitude

        const spinner = document.querySelector('#spinner')
        spinner.classList.add('is-active')

        axios
            .post("/geocode",{lat, lng})
            .then(function(response){
                let card = templateCard(response.data.address,
response.data.image)
                grid.innerHTML += card

                spinner.classList.remove('is-active')
            })
            .catch(function(error){
                spinner.classList.remove('is-active')
            })
```

Pronto!