Universidad Politécnica Salesiana

Nombre: Gabriel Cacuango.

Grupo: 1 **Docente:** Ing. Rodrigo Tufiño

Fecha: 6/16/2020

Practica 05 APP Clima

En esta práctica se procede al consumo de una API para la obtención de datos para mostrarlos en consola en base a una ciudad, opcionalmente podremos obtener dos datos extras con respecto a la misma ciudad los cuales son temperatura y humedad.

Herramientas:

- Yargs
- Openweathermap
- Axios
- Postman

Primeros pasos:

Lo primero que se procede es a crear una cuenta en openweathermap para que se nos asigne una API key que nos permitirá realizar 60 solicitudes por minuto, en la siguiente pagina

https://openweathermap.org/

Luego procedemos a usar postman para ver un ejemplo de como se ejecuta un llamado con nuestro usuario y los parámetros necesarios detallados a continuación:

Para nuestro caso utilizaremos el llamado mediante la ciudad, pero existen otros métodos de llamado como por ejemplo por las coordenadas geográficas, un zZIP code entro otras.

By city ID

Description:

You can call by city ID. API responds with exact result.
We recommend to call API by city ID to get unambiguous result for your city.

List of city ID city.list.json.gz can be downloaded here http://bulk.openweathermap.org/sample/

API call:

`api.openweathermap.org/data/2.5/weather?id={city id}&appid={your api key}`

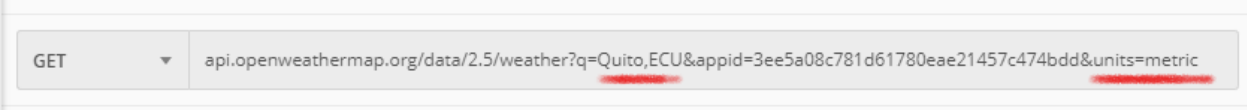
Parameters:

id City ID

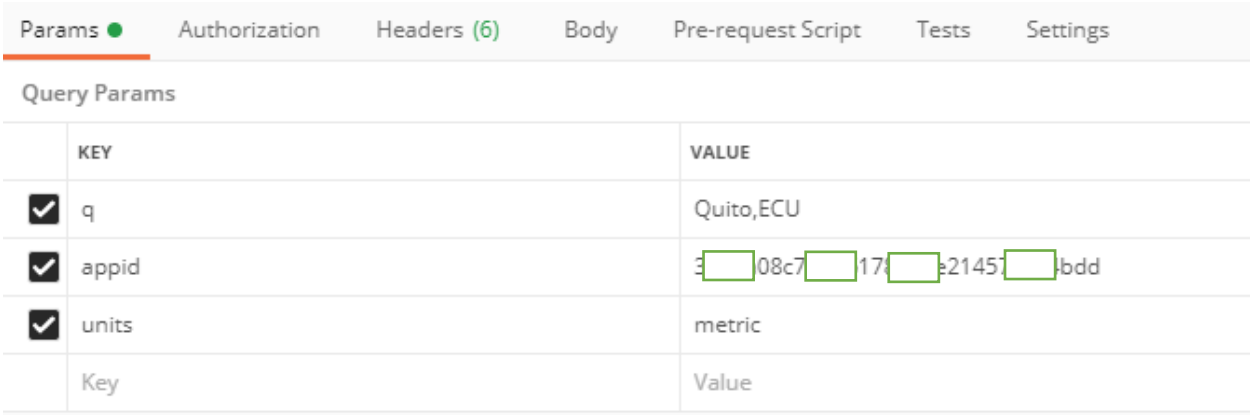
Examples of API calls:

api.openweathermap.org/data/2.5/weather?id=2172797

Ingresamos la petición a nuestro postman reemplazando y colocando la ciudad y el API key que se nos otorga al momento e realizar la inscripción, como parámetro adicional modificamos las unidades en las que nos entrega los datos para que la temperatura sea recibida en grados centígrados.



En los parámetros podemos visualizar los parámetros que son utilizados y enviamos para verificar la respuesta:



Nos regresa un JSON con la inforacion acerca de la ciudad como respuesta a nuestra solicitud:


```json
{
    "coord": {
        "lon": -78.52,
        "lat": -0.23
    },
    "weather": [
        {
            "id": 802,
            "main": "Clouds",
            "description": "scattered clouds",
            "icon": "03d"
        }
    ],
    "base": "stations",
    "main": {
        "temp": 19,
        "feels_like": 15.99,
        "temp_min": 19,
        "temp_max": 19,
        "pressure": 1025,
        "humidity": 63
    },
    "visibility": 10000,
    "wind": {
        "speed": 5.1,
        "deg": 10
```


Donde nos intera el dato de temp = Temperatura , pressure= presion, humidity=humedad y por consiguiente comenzamos a realizar nuestro programa.

Abrimos nuestro Visual Studio Code donde creamos un proyecto inicializamos nuestro proyecto con npm init para que se instalen los paquetes iniciales, luego creamos un app.js con lo siguiente:

```
const argv = require('yargs').options({
        ciudad: {
            alias: 'c',
            desc: 'Nombre para obtener el clima',
            demand: true
        }
    })
    //options capture others arguments opcional from the principal ciudad
    .option('humedad', { alias: 'h', demandOption: false, describe: 'Parametro para obtener la humedd' })
    .option('temperatura', { alias: 't', demandOption: false, describe: 'Parametro para obtemer la temperatura' }).
```

Utilizaremos la librería de Yargs para lo cual en la consola la instalamos

npm install yargs –save

luego se establecen los parametrps necesarios que unicamente se necesita la ciudad la cual vamos a consultar y como valores opcionales se establecen los argumentos humedad y temperatura.

En nuestro explorador creamos una carpeta controlador y dentro creamos una clase llamada clima.app con los siguientes componentes:

```
//import library axios for consume API external openweathermap
const axios = require('axios');
```

Primero usaremos axios para el consumo de la API externa por lo cual también procedemos a instalarla con el comando

npm install axios –save

```
const getClima = async(argv) => {
    //encodeURI complete the spaces whith & in http peticions example "buenos&aires"
    const ciudadURL = encodeURI(argv.ciudad);
    //define argument options for parametres humidity and pressure
    if (argv.o == undefined) {
        const resp = await axios.get(`https://api.openweathermap.org/data/2.5/weather?q=${ciudadURL},ECU&appid=3ee5a08c781d61780eae21457c474b
        return `Temperatura : ${resp.data.main.temp}`;

    } else {
        const resp = await axios.get(`https://api.openweathermap.org/data/2.5/weather?q=${ciudadURL},ECU&appid=3ee5a08c781d61780eae21457c474b
        let a = argv.o.length;
        if (a == 1) {
            if (argv.o[0] == 'p') {
                console.log(resp.data.main.pressure);
                return `Temperatura : ${resp.data.main.temp} , presion: ${resp.data.main.pressure}`;
            }
            if (argv.o[0] == 'h') {
                return `Temperatura : ${resp.data.main.temp} , humedad: ${resp.data.main.humidity}`
            } else {
                return ` parametro :${argv.o[0]}, no valido intente nuevamente`
            }
        } else {
            return `Temperatura : ${resp.data.main.temp} , presion: ${resp.data.main.pressure}, humedad: ${resp.data.main.humidity}`;
        }
```

Comenzamos creando una función en la cual recibirá el argv con ayuda de nuestra yargs para recibir los parámetros que son enviados por consola y establecemos si el argumento -o esta vacío en caso de esta vacío realizara una petición con ayuda de axion donde almacena el JSON recibido y luego tomamos los datos de nuestro interés que es la temperatura y luego retornamos el valor de la temperatura para mostrarla en consola con el nombre de la ciudad enviada como parámetro.

En caso de no estar vacío verifica los argumentos enviados para proceder a bien consultar solo la humedad o solo la temperatura o ambas dependiendo el caso nos regresa el resultado para luego mostrarlo en consola junto a la ciudad de consulta.

```
    //exports modeule get clima to use un app principal class
module.exports = {
    getClima
}
```

Exportamos el modulo de getClima para poder utilizarlo en nuestra clase principal.

Una vez realizado eso regresamos a nuestra app.js donde importamos la clase clima para poder llamar a nuestro método:

```
// import class clima from controller folder
const clima = require('./controlador/clima');
```

Luego de esto creamos una función para obtener la información necesaria de la ciudad a consultar aquí hay que recalcar que utilizamos una función async en la función getClima por lo cual debemos usar un await para manejarla como una promesa.

```
const getInformacion = async(ciudad) => {
    try {
        //use always await in a async function
        const temp = await clima.getClima(argv);
        return `Datos de la ciudad: ${ciudad} :  ${temp}`
    } catch (e) {
        return `No sew pudo obtener el clima de la ciudad ${ciudad}`
    }
}
```

Eso lo controlamos dentro de un try/catch para el manejo de excepciones y el respectivo retorno para cada caso, de esta forma controlamos en caso de existir error en nuestro código.

Luego llamamos a la función recordemos como es un async usamos un then y el catch para captura de errores, enviándole como parámetro la ciudad que se almacena en nuestro argv.ciudad.

```
getInformacion(argv.ciudad).then(console.log).catch(console.log);
```

Resultados obtenidos:

Solo enviando la ciudad a consultar

```
Gabriel@LAPTOP-RS81EFGL MINGW64 ~/Documents/Gabriel/05-Clima/node-05-Clima (master)
$ node app.js -c "Quito"
Datos de la ciudad: Quito :  Temperatura : 16
```

Enviando parámetro opcional con humedad:

```
Gabriel@LAPTOP-RS81EFGL MINGW64 ~/Documents/Gabriel/05-Clima/node-05-Clima (master)
$ node app.js -c "Quito" -o h
Datos de la ciudad: Quito :  Temperatura : 16 , humedad: 77
```

Enviando parámetro opcional con presión:

```
Gabriel@LAPTOP-RS81EFGL MINGW64 ~/Documents/Gabriel/05-Clima/node-05-Clima (master)
$ node app.js -c "Quito" -o p
1025
Datos de la ciudad: Quito :  Temperatura : 16 , presion: 1025
```

Enviando los dos parámetros opcionales humedad y presión:

```
Gabriel@LAPTOP-RS81EFGL MINGW64 ~/Documents/Gabriel/05-Clima/node-05-Clima (master)
$ node app.js -c "Quito" -o p -o h
Datos de la ciudad: Quito :  Temperatura : 16 , presion: 1025, humedad: 77
```

Conclusión:

- NodeS nos permite el uso de herramientas poderosas como una de ellas es axios que nos permite el consumo de una API externa y podemos realizar una aplicación actual consumiendo datos reales alojados en la web.
- Podemos hacer uso de varias API's para realizar trabajos no únicamente limitarnos a un tipo ya que existen un sinfín de este tipo.

Enlace del código a Github: https://github.com/GabrielCacuango07/node-05-Clima