

Práctica 05 - Aplicación del clima

Wilson Aguilar
PLATAFORMAS WEB

16 de junio de 2020

1. Inicio del proyecto y dependencias

Esta aplicación se centra en el consumo de un servicio de internet y poder mostrar esos datos dentro de la aplicación.

En específico vamos a crear una aplicación que nos dara información del clima de alguna ciudad del mundo. Para ello nos vamos a conectar a una API que nos brinde esta información.

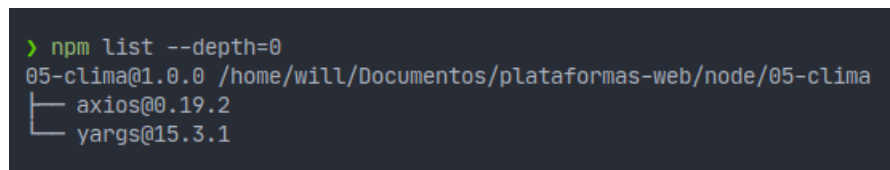
Para inicializar el proyecto ejecutamos el comando: `npm init -y`

Además, instalaremos las siguientes dependencias:

Yargs Nos ayudara con la creación de una app en consola.

Axios Nos permite realizar peticiones http.

Para ello ejecutamos `npm install --save yargs axios`



```
> npm list --depth=0
05-clima@1.0.0 /home/will/Documentos/plataformas-web/node/05-clima
├── axios@0.19.2
└── yargs@15.3.1
```

Figura 1: Listado de dependencias del proyecto

2. Configuración de yargs

Yargs es un paquete de node que nos ayuda a crear aplicaciones de consola de una manera mucho mas sencilla. En este caso nuestra aplicación contara de 2 opciones, una para ingresar la ciudad de donde vamos a consultar la información del clima y la otra para solicitar información extra (humedad y presion).

A screenshot of a code editor with a dark background and light-colored text. The code is written in JavaScript and configures the yargs module. It defines two options: 'city' and 'option'. The 'city' option has an alias 'c', a description 'Nombre de la ciudad', and is required. The 'option' option has an alias 'o', a description 'Devuelve la presion(p) o la humedad(h).', and has choices 'h' and 'p'. The code is as follows:

```
1  const argv = require('yargs')
2    .options({
3      city: {
4        alias: 'c',
5        desc: 'Nombre de la ciudad',
6        required: true
7      },
8      option: {
9        alias: 'o',
10       desc: 'Devuelve la presion(p) o la humedad(h).',
11       choices: ['h', 'p']
12     }
13   })
14   .argv;
```

Figura 2: Configuración del módulo yargs.

3. Consumo de servicios

3.1. Open Weather Map

Open Weather Map es un servicio en linea que nos proporciona datos meteorológicos. Nos conectaremos a esta API para recopilar la información que necesitamos. Para ello necesitamos crearnos una cuenta en este sitio para que nos proporcionen un token de acceso para poder solicitar datos al sitio.

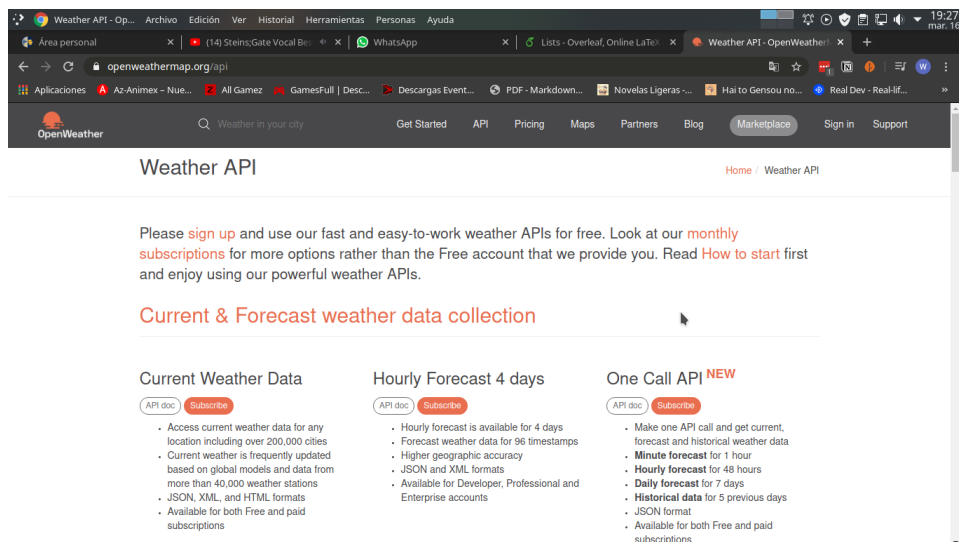


Figura 3: Sitio oficial de Open Weather Map.

Dentro de la documentación de Open Weather Map podemos observar como podemos realizar una petición de los datos que solicitamos.



Figura 4: Ejemplo de petición.

En este caso usaremos el enlace que nos proporciona la documentación para poder realizar pruebas y observar que datos nos devuelve.

El enlace que usaremos es el siguiente:

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

city Es el nombre de la ciudad que vamos a consultar.

key Es el token que nos da la api para poder realizar peticiones.

3.2. Solicitar datos

Para poder pedir los datos del clima que necesitamos debemos realizar una petición http de tipo GET hacia la url mostrada anteriormente. Los navegadores web por defecto cuando ingresan a un sitio realizan esta petición hacia el servidor solicitando la página, así que usaremos este método para que pedir la información por medio del navegador.

```
▼ {
  ▶ "coord": { ... }, // 2 items
  ▶ "weather": [ ... ], // 1 item
  "base": "stations",
  ▼ "main": {
    "temp": 287.15,
    "feels_like": 285.21,
    "temp_min": 287.15,
    "temp_max": 287.15,
    "pressure": 1026,
    "humidity": 87
  },
  "visibility": 10000,
  ▶ "wind": { ... }, // 2 items
  ▶ "clouds": { ... }, // 1 item
  "dt": 1592355207,
  ▶ "sys": { ... }, // 5 items
  "timezone": -18000,
  "id": 3652462,
  "name": "Quito",
  "cod": 200
}
```

Figura 5: Solicitando datos desde el navegador

Como observamos pedimos información acerca del clima de Quito y nos devolvió estos datos en un formato json. Cuando hacemos peticiones hacia una API, por lo general nos dan una respuesta en este formato.

También tenemos herramientas especializadas en realizar peticiones. Postman es una de ellas y muy famosa entre los desarrolladores. Postman no solo nos deja realizar peticiones GET, sino que todo tipo de peticiones, POST,

PUT, OPTION, etc. Además de que nos deja modificar varios aspectos de estas, es un programa especializado para probar API's y realizar peticiones.

Ahora para realizar una petición desde Postman solo debemos escoger el método a usar, en este caso GET, ingresamos la dirección y los parametros que son la q (query) y appid(key de api), esta vez añadimos un parametro mas que es metric y nos ayudara a que la temperatura que nos devuelve la API sea en grados celcius.

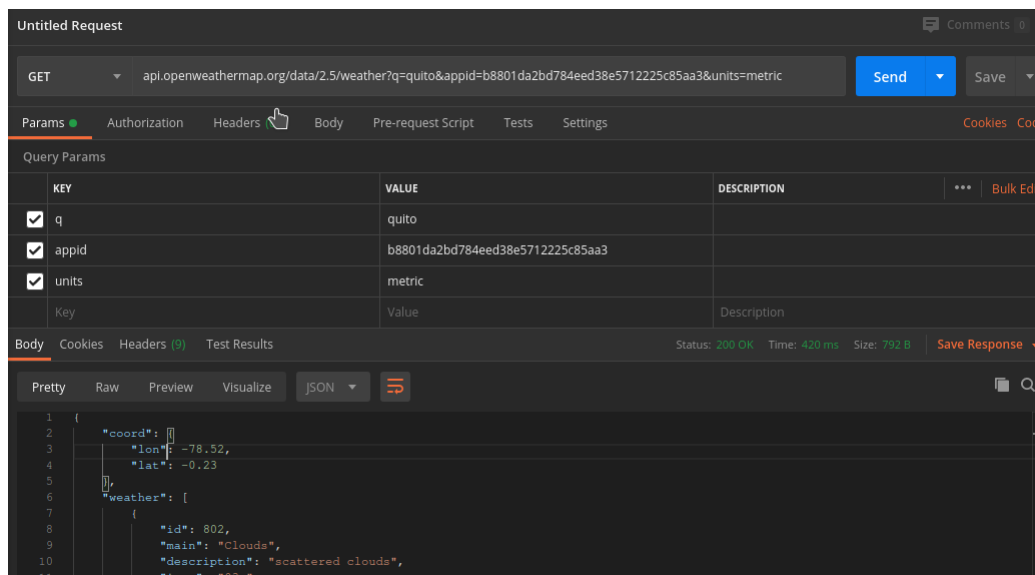


Figura 6: Peticion desde Postman

Como podemos observar postman es un poco mas complejo pero a cambio nos da mucha mas información acerca de como se realizo la petición, vemos que la petición fue exitosa con código de estado 200 ok y nos devolvio la misma información que el navegador.

4. Axios

Para poder realizar peticiones http desde algun lenguaje de programación nos apoyamos de librerias que nos permitan realizar esta acción. Para Nodejs y javascript Axios es una buena opción, es sencilla de utilizar y tiene buen soporte de la comunidad.

4.1. Configurando Axios

Para tener nuestro código mejor ordenado separamos la configuración de axios en un archivo aparte.



```
1  const axios = require('axios');
2
3  let instance = axios.create({
4    baseUrl: process.env.API_URI,
5  });
6
7
8  module.exports = {
9    instance
10 }
```

Figura 7: Configuración de Axios

Como observamos creamos una nueva instancia de axios con la url del sitio ya cargada en una variable de entorno.

De igual manera se cargo la API KEY en otro variable de entorno, esto para tener un mejor manejo de la seguridad.

5. Peticion desde axios

Creamos un archivo que nos sirve de controlador, aqui definimos un método que nos permitirá obtener los datos de la API. Utilizamos la instancia de axios creada anteriormente



```
1  const { instance: axios } = require('../config/axios')  
2  ;  
3  const getWeather = async (city, units = 'metric') => {  
4      let response = await axios  
5      .get('/', {  
6          params: {  
7              q: encodeURIComponent(city),  
8              appid: process.env.API_KEY,  
9              units  
10         }  
11     });  
12     return response;  
13 }
```

Figura 8: Peticion desde axios

Como esta instancia de axios esta configurada ya con la url basta con realizar la petición a la raíz, además de enviar los parametros necesarios que son la ciudad y la api key, además de las unidades para que obtenga la temperatura en grados celcius.

6. App

Ahora solo nos hace falta llamar el método y manejar los datos para que se muestren por pantalla.

```

1  getWeather(argv.city)
2    .then(res => {
3      res = res.data;
4
5      switch (argv.option) {
6        case 'h':
7          console.log(`La temperatura de ${res.name} es ${res.main.temp}°c.`);
8          console.log(`La humedad es ${res.main.humidity}%`);
9          break;
10       case 'p':
11         console.log(`La temperatura de ${res.name} es ${res.main.temp}°c.`);
12         console.log(`La presion es ${res.main.pressure}`);
13         break;
14
15       default:
16         console.log(`La temperatura de ${res.name} es ${res.main.temp}°c.`);
17         break;
18     }
19   })
20   .catch(err => {
21     console.log(`No se pudo encontrar el clima para ${argv.city}.`)
22   });
23
24

```

Figura 9: Lógica de app.js

Una vez ejecutamos el programa obtenemos los siguientes resultados:

```

> node app -c quito
La temperatura de Quito es 13°C.
> node app -c quito -o h
La temperatura de Quito es 13°C.
La humedad es 93%.
> node app -c quito -o p
La temperatura de Quito es 13°C.
La presion es 1027.

```

Figura 10: Ejemplo de ejecucion