

Solución del trabajo PDF ACTIVIDAD GUIADA 1

Se utilizó el Sitio WEB de open Data de Valencia. La fuente de datos que se consumió es de la estación de contaminación atmosférica de Boulevard Sur (7A). La url del servicio es: <http://mapas.valencia.es/WebsMunicipales/uploads/atmosferica/7A.csv>

La URL del repositorio de la solución del ejercicio es:

https://github.com/AlexRivas11/Actividad_1_Atmosferica

Se utilizó en la solución del ejercicio como estrategia de programación el motor NodeJS y paquetes externos como:

- fs propios de nodejs: para la creación de archivos
- request paquete de npm: para la comunicación http con el servicio WEB.
- csv2json: Este paquete se utilizó para convertir el csv a JSON
- mongoose: Este paquete se utilizó para la conexión a mongodb y las consultas.

EJERCICIO:

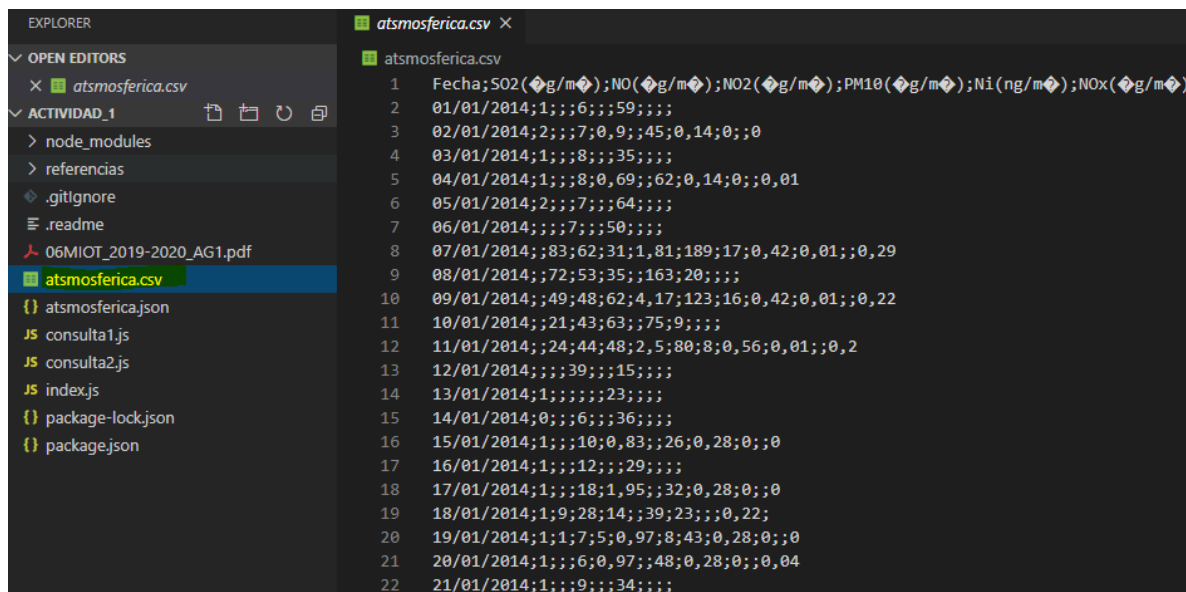
- Hacer una descarga de un conjunto de datos del Open Data de: Valencia, Madrid o Barcelona. Hablar sobre la fuente de los datos y la calidad de los datos. (Mínimo del conjunto de datos 100 registros o filas).

Respuesta:

```
const request = require('request')
const fs = require('fs')

request.get('http://mapas.valencia.es/WebsMunicipales/uploads/atmosferica/7A.csv').pipe(fs.createWriteStream('atmosferica.csv'))
```

Con este código se consiguió consumir el servicio del URL y crear el archivo csv. El resultado fue el siguiente:



Es una temática de medio ambiente por lo tanto la calidad de datos supera los 100 registros y estos con llevan valores que se pueden utilizar en queries.

- Generar un JSON de los datos en un fichero en local.

Respuesta:

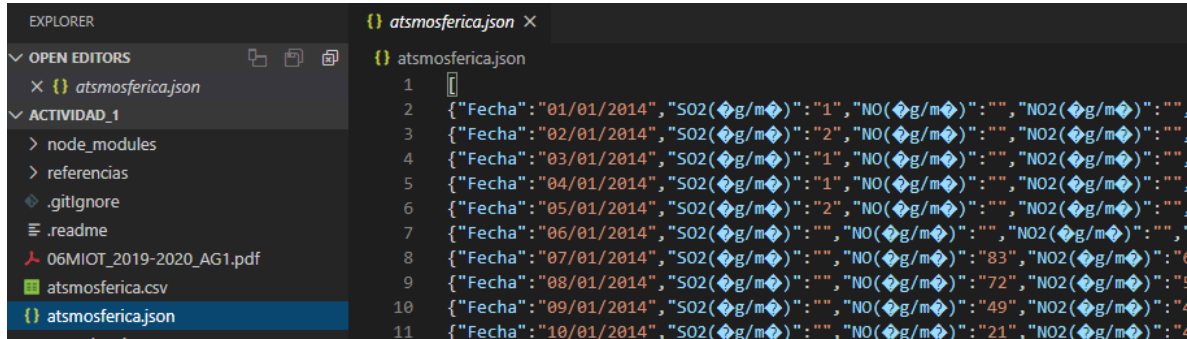
```
const fs = require('fs')
const csvFilePath='./atmosfera.csv'
const csv2json = require('csv2json');

setTimeout(() => {

    fs.createReadStream(csvFilePath)
        .pipe(csv2json({
            // Defaults to comma.
            separator: ','
        }))

        .pipe(fs.createWriteStream('atmosfera.json'));
```

Con este código se logró crear un archivo `atmosferica.json` con todo los datos del csv, del sitio web de Valencia. Para ello se utilizó los paquetes externos `fs` y `csv2json`. El resultado fue:



```

1  [
2  {
3    "Fecha": "01/01/2014",
4    "SO2(g/m³)": "1",
5    "NO(g/m³)": "",
6    "NO2(g/m³)": "",
7    "PM10(g/m³)": "83",
8    "NI(g/m³)": "72",
9    "NOX(g/m³)": "49",
10   "OZONO(g/m³)": "21",
11   "AS(g/m³)": "21",
12   "PB(g/m³)": "21"
13 },
14 {
15   "Fecha": "02/01/2014",
16   "SO2(g/m³)": "2",
17   "NO(g/m³)": "",
18   "NO2(g/m³)": "",
19   "PM10(g/m³)": "83",
20   "NI(g/m³)": "72",
21   "NOX(g/m³)": "49",
22   "OZONO(g/m³)": "21",
23   "AS(g/m³)": "21",
24   "PB(g/m³)": "21"
25 },
26 {
27   "Fecha": "03/01/2014",
28   "SO2(g/m³)": "1",
29   "NO(g/m³)": "",
30   "NO2(g/m³)": "",
31   "PM10(g/m³)": "83",
32   "NI(g/m³)": "72",
33   "NOX(g/m³)": "49",
34   "OZONO(g/m³)": "21",
35   "AS(g/m³)": "21",
36   "PB(g/m³)": "21"
37 },
38 {
39   "Fecha": "04/01/2014",
40   "SO2(g/m³)": "1",
41   "NO(g/m³)": "",
42   "NO2(g/m³)": "",
43   "PM10(g/m³)": "83",
44   "NI(g/m³)": "72",
45   "NOX(g/m³)": "49",
46   "OZONO(g/m³)": "21",
47   "AS(g/m³)": "21",
48   "PB(g/m³)": "21"
49 },
50 {
51   "Fecha": "05/01/2014",
52   "SO2(g/m³)": "2",
53   "NO(g/m³)": "",
54   "NO2(g/m³)": "",
55   "PM10(g/m³)": "83",
56   "NI(g/m³)": "72",
57   "NOX(g/m³)": "49",
58   "OZONO(g/m³)": "21",
59   "AS(g/m³)": "21",
60   "PB(g/m³)": "21"
61 },
62 {
63   "Fecha": "06/01/2014",
64   "SO2(g/m³)": "",
65   "NO(g/m³)": "",
66   "NO2(g/m³)": "",
67   "PM10(g/m³)": "83",
68   "NI(g/m³)": "72",
69   "NOX(g/m³)": "49",
70   "OZONO(g/m³)": "21",
71   "AS(g/m³)": "21",
72   "PB(g/m³)": "21"
73 },
74 {
75   "Fecha": "07/01/2014",
76   "SO2(g/m³)": "",
77   "NO(g/m³)": "83",
78   "NO2(g/m³)": "72",
79   "PM10(g/m³)": "49",
80   "NI(g/m³)": "21",
81   "NOX(g/m³)": "21",
82   "OZONO(g/m³)": "21",
83   "AS(g/m³)": "21",
84   "PB(g/m³)": "21"
85 },
86 {
87   "Fecha": "08/01/2014",
88   "SO2(g/m³)": "",
89   "NO(g/m³)": "72",
90   "NO2(g/m³)": "49",
91   "PM10(g/m³)": "21",
92   "NI(g/m³)": "21",
93   "NOX(g/m³)": "21",
94   "OZONO(g/m³)": "21",
95   "AS(g/m³)": "21",
96   "PB(g/m³)": "21"
97 },
98 {
99   "Fecha": "09/01/2014",
100  "SO2(g/m³)": "",
101  "NO(g/m³)": "49",
102  "NO2(g/m³)": "21",
103  "PM10(g/m³)": "21",
104  "NI(g/m³)": "21",
105  "NOX(g/m³)": "21",
106  "OZONO(g/m³)": "21",
107  "AS(g/m³)": "21",
108  "PB(g/m³)": "21"
109 },
110 {
111   "Fecha": "10/01/2014",
112   "SO2(g/m³)": "",
113   "NO(g/m³)": "21",
114   "NO2(g/m³)": "21",
115   "PM10(g/m³)": "21",
116   "NI(g/m³)": "21",
117   "NOX(g/m³)": "21",
118   "OZONO(g/m³)": "21",
119   "AS(g/m³)": "21",
120   "PB(g/m³)": "21"
121 }
122 ]

```

- Insertar en MongoDB todos los datos sin utilizar un bucle. Definir colección y código.

Respuesta:

```

const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/atmosferica', {useNewUrlParser: true}, ((err) => {
  if(err) {
    console.log(err)
  } else {
    console.log('Se conecto correctamente')
  }
})))

var post_schema = mongoose.Schema({
  Fecha : String,
  SO2: String,
  NO: String,
  NO2: String,
  PM10: String,
  NI: String,
  NOX: String,
  OZONO: String,
  AS: String,
  PB: String,

```

```

    PBAP: String,
    CD: String
  });
var post_model = mongoose.model('datos_atmosfericos', post_schema);

setTimeout(() => {

  let rawdata = fs.readFileSync('atmosferica.json');
  let data_atmosferica = JSON.parse(rawdata);
  // console.log(data_atmosferica)

  // var newData = new post_model({data_atmosferica});

  // saving json schema to mongodb

  // newData.save(function(err){
  //   if (err) {
  //     throw err;
  //   }
  //   console.log('INSERTED!');
  // });

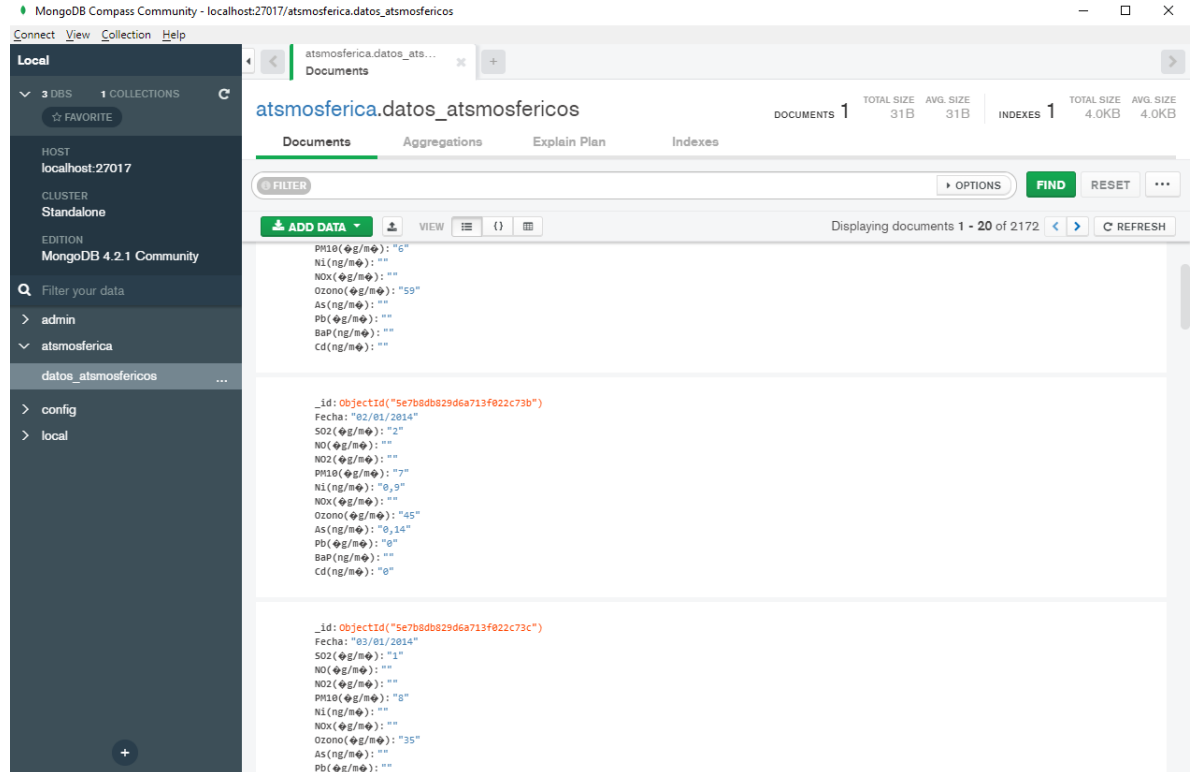
  post_model.collection.insertMany(data_atmosferica, (err, result) =>
{
  console.log('SE HAN INSERTADO LOS DATOS');
})

}, 3000);

```

Con ayuda del paquete mongoose se logró la conexión con la base de datos, posterior a ello se declara una variable con el fin de cargar los datos del archivo json para luego de nuevo con el paquete mongoose insertarlos en la base de datos. El resultado fue el siguiente:

El resultado en la bd es el siguiente:



- Mediante una consulta a Mongo obtener unos campos concretos de una consulta buscando en un atributo un valor concreto. El valor tiene que ser un valor numérico.

Respuesta:

En un script aparte denominado consulta1.js se crea toda la lógica necesaria para la consulta utilizando el paquete mongoose y fs para crear un archivo txt con la información consulta, la consulta a realizar será sobre el ozono con la cifra de 45. El código que se uso para la consulta fue el siguiente:

```
const mongoose = require('mongoose');
const fs = require('fs');

mongoose.connect('mongodb://localhost/atmosferica', {useNewUrlParser: true}, ((err) => {
  if(err) {
    console.log(err)
```

```

    } else {
      console.log('Se conecto correctamente')
    }
  })
})

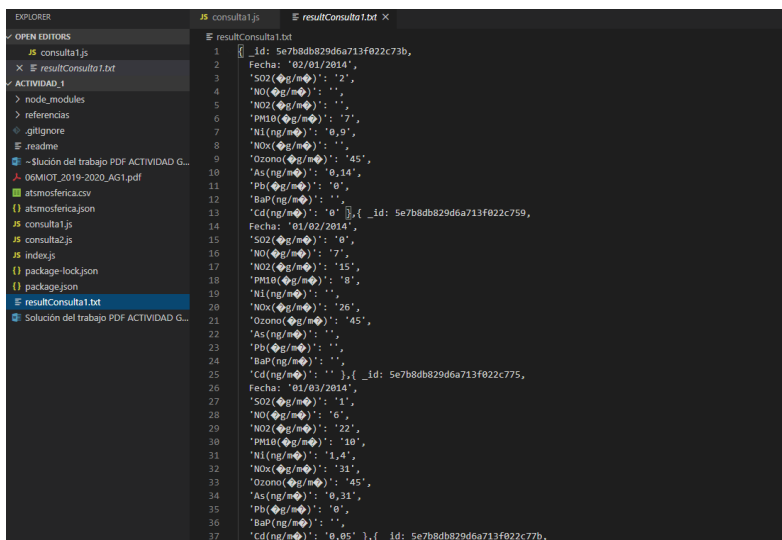
var post_schema = mongoose.Schema({data : JSON});
var post_model = mongoose.model('datos_atmosfericos', post_schema);

post_model.find( { 'Ozono(g/m³)': '45' }, (err, res) => {
  if(err) {
    // console.log(err)
    return;
  } else {

    console.log(JSON.stringify(res))

    fs.writeFile('resultConsulta1.txt', res, (err) => {
      if (err) throw err;
      console.log('File is created successfully.');
```

El resultado en el archivo txt mostro la siguiente información:



```

1  { "_id": "5e7b8db829d6a713f022c73b",
2    "Fecha": "02/01/2014",
3    "SO2(g/m³)": "12",
4    "NO(g/m³)": "7",
5    "NO2(g/m³)": "15",
6    "PM10(g/m³)": "7",
7    "NI(g/m³)": "0.9",
8    "NOx(g/m³)": "1",
9    "Ozono(g/m³)": "45",
10   "As(g/m³)": "0.14",
11   "Pb(g/m³)": "0",
12   "BaP(g/m³)": "0",
13   "Cd(g/m³)": "0", { "_id": "5e7b8db829d6a713f022c759",
14     "Fecha": "01/02/2014",
15     "SO2(g/m³)": "0",
16     "NO(g/m³)": "7",
17     "NO2(g/m³)": "15",
18     "PM10(g/m³)": "0",
19     "NI(g/m³)": "0",
20     "NOx(g/m³)": "26",
21     "Ozono(g/m³)": "45",
22     "As(g/m³)": "0",
23     "Pb(g/m³)": "0",
24     "BaP(g/m³)": "0",
25     "Cd(g/m³)": "0", { "_id": "5e7b8db829d6a713f022c775",
26     "Fecha": "01/03/2014",
27     "SO2(g/m³)": "1",
28     "NO(g/m³)": "6",
29     "NO2(g/m³)": "22",
30     "PM10(g/m³)": "10",
31     "NI(g/m³)": "1.4",
32     "NOx(g/m³)": "31",
33     "Ozono(g/m³)": "45",
34     "As(g/m³)": "0.31",
35     "Pb(g/m³)": "0",
36     "BaP(g/m³)": "0",
37     "Cd(g/m³)": "0.05", { "_id": "5e7b8db829d6a713f022c77b",

```

- Hacer una consulta de Mongo de forma que nos devuelva, en función de un atributo, el top 3 ordenado. ¿Existe el atributo “coordinates” en el conjunto de datos? ¿Cómo se consulta?

Resultado: El atributo coordinate no existe ya que se cambió el servicio planteado por la guía por el hecho de que al estar el país en cuarentena no mostraba registros, se optó por un servicio de medio ambiente. Para sacar el top 3 se utilizara las funciones de mongoose de la siguiente forma:

```
const mongoose = require('mongoose');
const fs = require('fs');

mongoose.connect('mongodb://localhost/atmosferica', {useNewUrlParser: true}, ((err) => {
  if(err) {
    console.log(err)
  } else {
    console.log('Se conecto correctamente')
  }
})))

var post_schema = mongoose.Schema({data : JSON});
var post_model = mongoose.model('datos_atmosfericos', post_schema);

post_model.find( { }).sort('Ozono(g/m)').exec((err, res) => {
  if(err) {
    // console.log(err)
    return;
  } else {

    // console.log(JSON.stringify(res))
    let top = [];
    top.push(res[res.length - 1]);
    top.push(res[res.length - 2]);
    top.push(res[res.length - 3]);
    fs.writeFile('resultConsulta2.txt', top, (err) => {
      if (err) throw err;
    });
  }
});
```

```

        console.log('File is created successfully.');
```

```

    })
  }
})
```

Se utilizó con el paquete mongoose una función de búsqueda con otra subfunción de sort que le permite retornar una promesa y con el resultado de la consulta los últimos tres elementos son el top 3 de los niveles mas altos de ozono de valencia. El archivo creado a partir de esta consulta es:

```

1 { _id: '5e7b8db29d6a713f022ceb6',
2   Fecha: '08/08/2018',
3   'SO2(ng/m³)': '4',
4   'NO2(ng/m³)': '4',
5   'PM10(ng/m³)': '28',
6   'PM10(ng/m³)': '28',
7   'NOx(ng/m³)': '26',
8   'Ozono(ng/m³)': '98',
9   'As(ng/m³)': '1',
10  'Pb(ng/m³)': '1',
11  'BaP(ng/m³)': '1',
12  'Cd(ng/m³)': '1',
13  'cd(ng/m³)': '1',
14  Fecha: '24/04/2019',
15  'SO2(ng/m³)': '4',
16  'NO2(ng/m³)': '3',
17  'PM10(ng/m³)': '13',
18  'PM10(ng/m³)': '13',
19  'NOx(ng/m³)': '18',
20  'Ozono(ng/m³)': '96',
21  'As(ng/m³)': '1',
22  'Pb(ng/m³)': '1',
23  'BaP(ng/m³)': '1',
24  'Cd(ng/m³)': '1',
25  'cd(ng/m³)': '1',
26  Fecha: '21/04/2019',
27  'SO2(ng/m³)': '3',
28  'NO2(ng/m³)': '2',
29  'PM10(ng/m³)': '5',
30  'PM10(ng/m³)': '24',
31  'NOx(ng/m³)': '1,47',
32  'Ozono(ng/m³)': '7',
33  'Ozono(ng/m³)': '95',
34  'As(ng/m³)': '0,42',
35  'Pb(ng/m³)': '0,01',
36  'BaP(ng/m³)': '1',
37  'Cd(ng/m³)': '0,07' }
```

```

$ node consulta2
(node:15632) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { unifiedTopology: true } to the MongoClient constructor.
Se conectó correctamente.
File is created successfully.
```

- Recorrer todos los registros de la colección y realizar una media en un atributo.

Respuesta: Se sacará la media del ozono en valencia con todos los registros que están el csv. Para ello se utilizó un ciclo for demostrado de la siguiente forma:

```

const mongoose = require('mongoose');
const fs = require('fs');

mongoose.connect('mongodb://localhost/atmosferica', {useNewUrlParser: true}, ((err) => {
```



```

    if(err) {
        console.log(err)
    } else {
        console.log('Se conecto correctamente')
    }
})))

var post_schema = mongoose.Schema({data : JSON});
var post_model = mongoose.model('datos_atmosfericos', post_schema);

post_model.find( { }).sort('Ozono(¿g/m¿)').exec((err, res) => {
    if(err) {
        // console.log(err)
        return;
    } else {

        let media = 0;
        let suma = 0;
        let arrayString = JSON.stringify(res);
        let arrayCurado = JSON.parse(arrayString);

        for (let index = 0; index < arrayCurado.length; index++) {

            suma = suma + Number(arrayCurado[index]["Ozono(¿g/m¿)"]);

        }

        console.log(suma);

        media = suma / Number(arrayCurado.length);

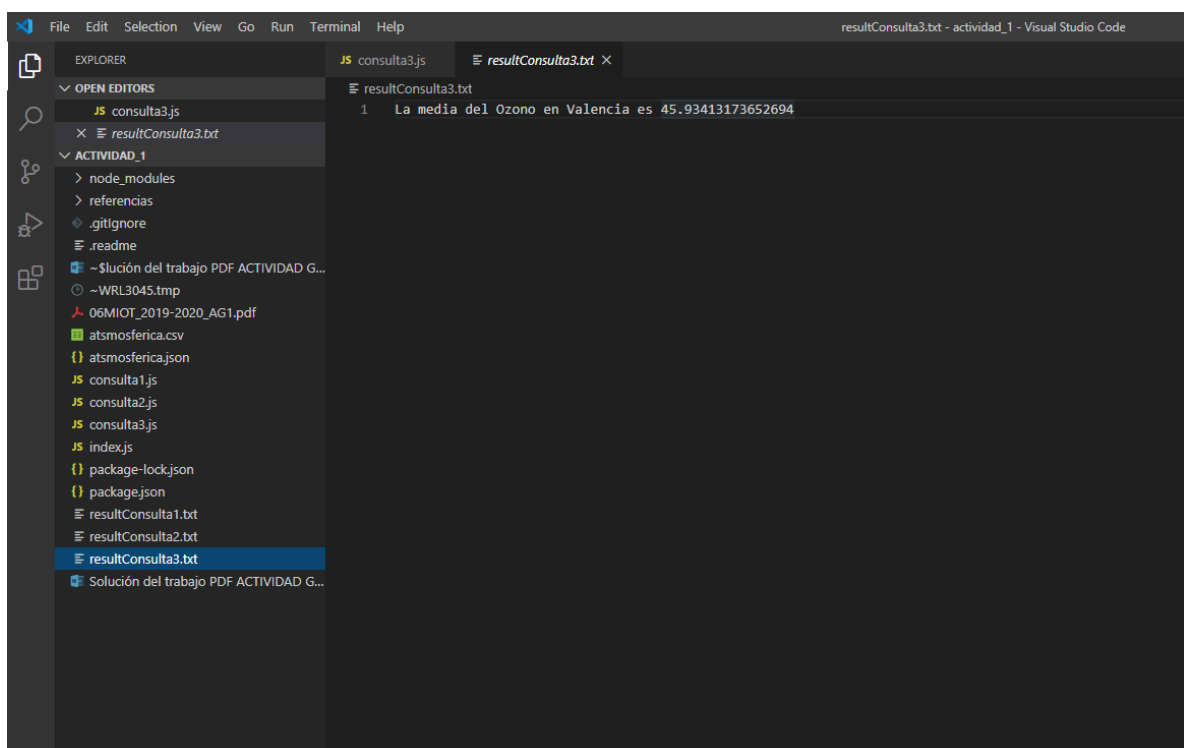
        resultado = 'La media del Ozono en Valencia es ' + String(media)
    }
})

```

```
fs.writeFile('resultConsulta3.txt', resultado, (err) => {  
    if (err) throw err;  
    console.log('File is created successfully.');
```

```
    })  
}  
})
```

EL resultado es



Conclusiones:

- Algunos Datos presentaron errores con los paquetes por caracteres especiales pero por conveniencia, JS es un lenguaje que permite parsear y curar ciertos datos problemáticos.
- El análisis de información maneja unos tiempos bastante ágiles con respecto a cálculos, claro que esto siempre depende de un buen hardware, cabe resaltar que estos algoritmos fueron probados en un portátil común.