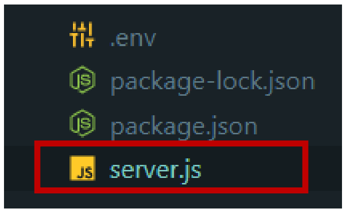
# Documentation

*Mini-Project **AIR QUALITY***

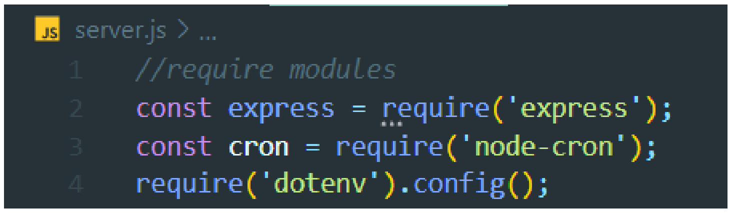
## 1.Build Node Js REST API:

After we initialize the project and install all the necessary packages,
we start the development by setting up the main file in our project which is **server.js**



```
.env
package-lock.json
package.json
server.js
```

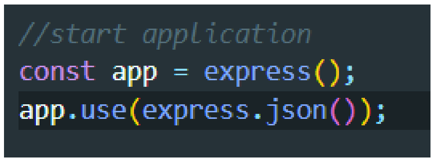
In this project we're using **express** framework, so we start by requiring this module along with **node-cron** module and **dotenv** module.



```
server.js > ...
//require modules
const express = require('express');
const cron = require('node-cron');
require('dotenv').config();
```

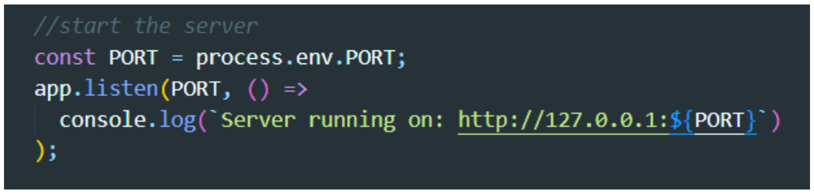
**node-cron** is to perform cron jobs in node, and **dotenv** is to require certain data stored in our env file, and the reason we store these data in this file is to add an extra layer of protection and security to our project.

Next, we start the application using express.



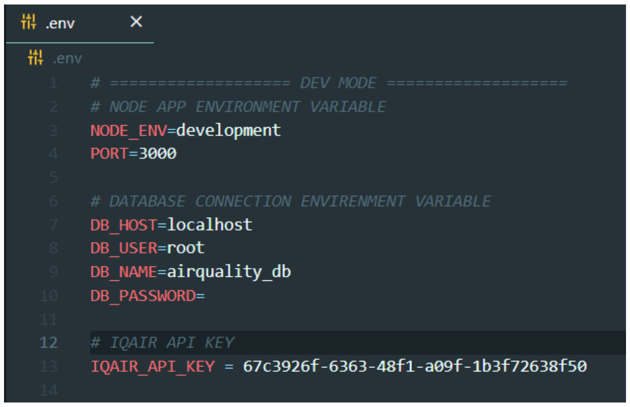
```
//start application
const app = express();
app.use(express.json());
```

And at last, we setup the project to listed on the given port stored in the **env** file.



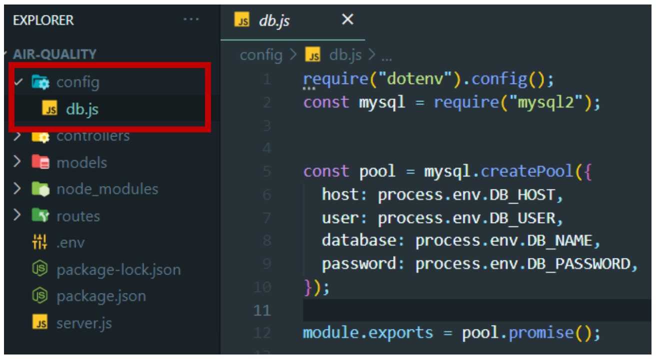
```
//start the server
const PORT = process.env.PORT;
app.listen(PORT, () =>
  console.log(`Server running on: http://127.0.0.1:${PORT}`)
);
```

*<u>Note 1:</u> the **env** file contain variables such as **PORT** which the app is listening to, database connection info, in order to connect to the database, and **IQAIR API key** so we can perform the requested tasks using their API.*



```
.env
# =================== DEV MODE ===================
# NODE APP ENVIRONMENT VARIABLE
NODE_ENV=development
PORT=3000

# DATABASE CONNECTION ENVIRENMENT VARIABLE
DB_HOST=localhost
DB_USER=root
DB_NAME=airquality_db
DB_PASSWORD=

# IQAIR API KEY
IQAIR_API_KEY = 67c3926f-6363-48f1-a09f-1b3f72638f50
```

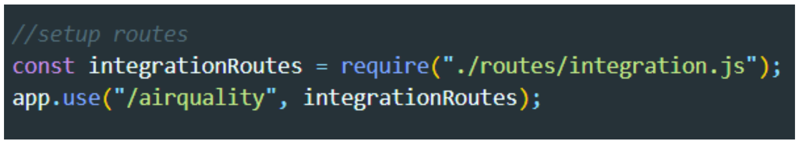
*<u>Note 2:</u> the config file contains a file called **db.js** which contains the default setup for our project to connect to the database using the data within our **env** file.*



```
config > db.js > ...
require("dotenv").config();
const mysql = require("mysql2");


const pool = mysql.createPool({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  database: process.env.DB_NAME,
  password: process.env.DB_PASSWORD,
});

module.exports = pool.promise();
```

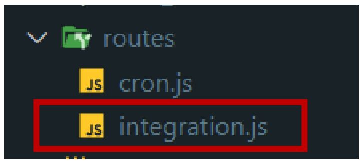
## 2. Get air quality for the given zone:

We start this increment by creating an endpoint Which is “**/airquality**” to make a call to **IQAIR API**,



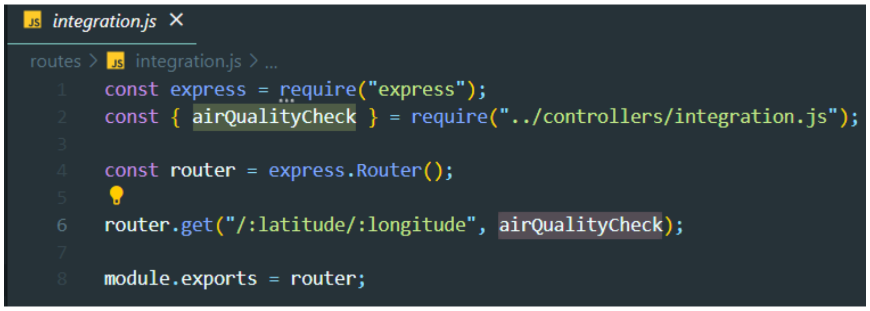
```
//setup routes
const integrationRoutes = require("./routes/integration.js");
app.use("/airquality", integrationRoutes);
```

So, the way it works is this URL is mapped to the route located in **routes/integration.js**



```
routes
  cron.js
  integration.js
```

This route is requiring two parameters **(Latitude, Longitude)**, when this route is called the function **airQualityCheck** will be executed which is located in **controllers/integration.js**

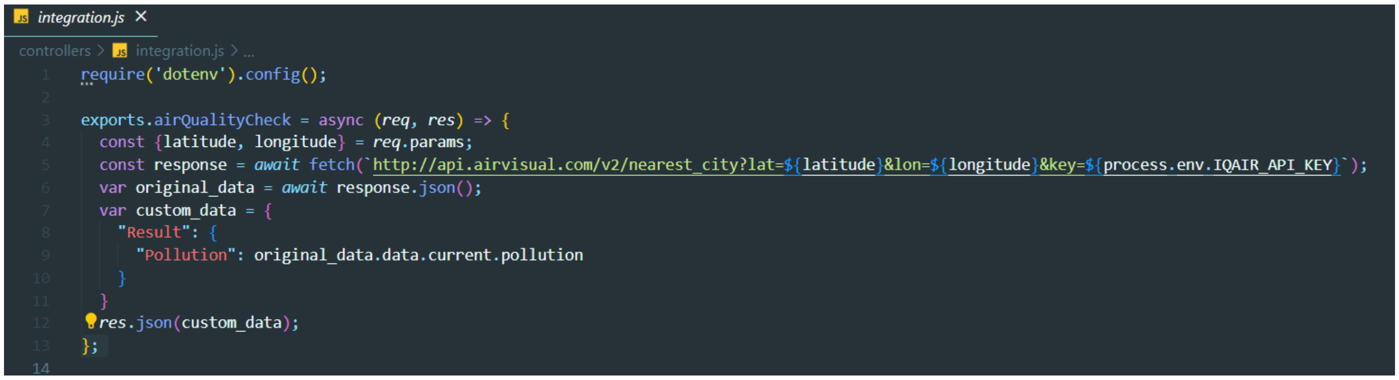


integration.js ×

routes > integration.js > ...

```
const express = require("express");
const { airQualityCheck } = require("../controllers/integration.js");

const router = express.Router();

router.get("/:latitude/:longitude", airQualityCheck);

module.exports = router;
```

The function retrieves the **latitude** and **longitude** from the request parameters (URL Variables), then we make a **GET Request** sing the **IQAIR API** for the **nearest city data (GPS coordinates),** then we store the response in a variable called **original_data.**



integration.js ×

controllers > integration.js > ...

```
require('dotenv').config();

exports.airQualityCheck = async (req, res) => {
  const {latitude, longitude} = req.params;
  const response = await fetch(`http://api.airvisual.com/v2/nearest_city?lat=${latitude}&lon=${longitude}&key=${process.env.IQAIR_API_KEY}`);
  var original_data = await response.json();
  var custom_data = {
    "Result": {
      "Pollution": original_data.data.current.pollution
    }
  }
  res.json(custom_data);
};
```

*Note: the API URL require the **latitude, longitude** and the **IQAIR API Key**.*

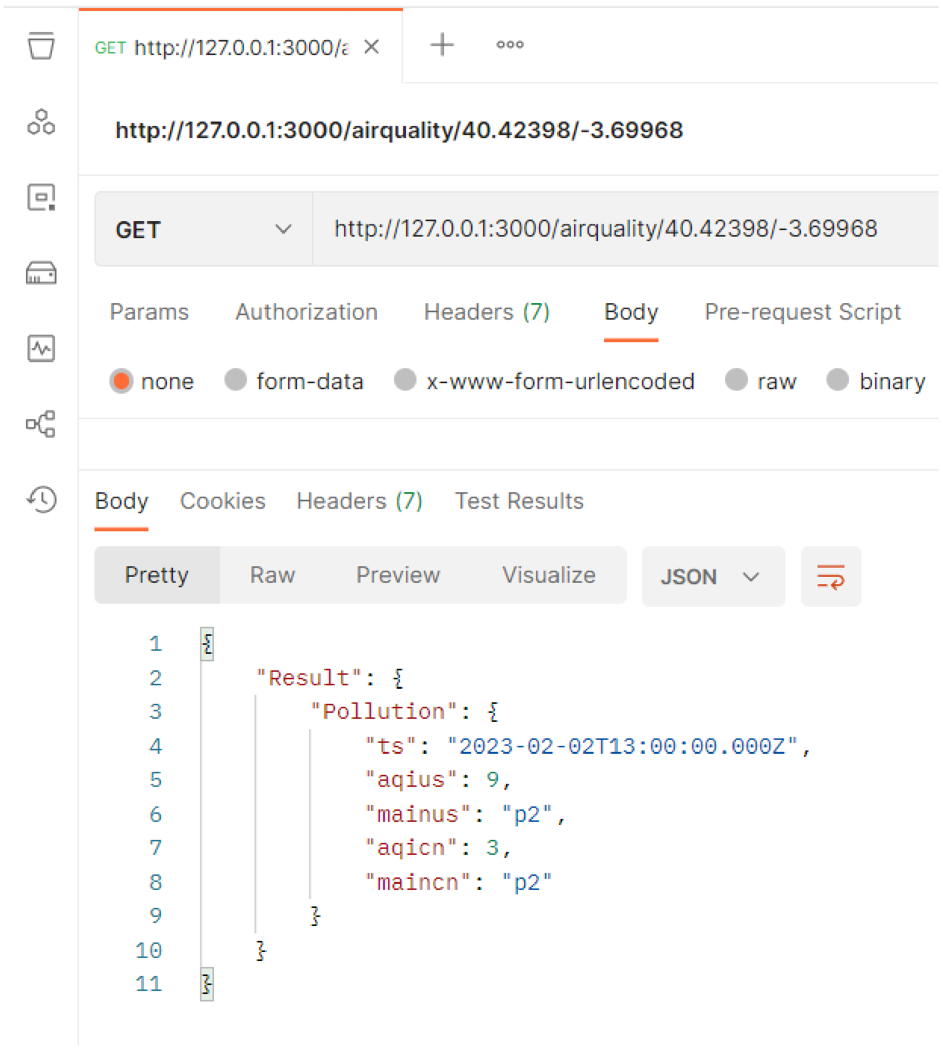
- At last, we customize the response for our project according to the challenge requirements, the image bellow demonstrate the result requested in the test.

***Result:***

**Request***:* http://127.0.0.1:3000/airquality/40.42398/-3.69968

**Response***:*

```
{
  "Result": {
    "Pollution": {
      "ts": "2023-02-02T13:00:00.000Z",
      "aqius": 9,
      "mainus": "p2",
      "aqicn": 3,
      "maincn": "p2"
    }
  }
}
```



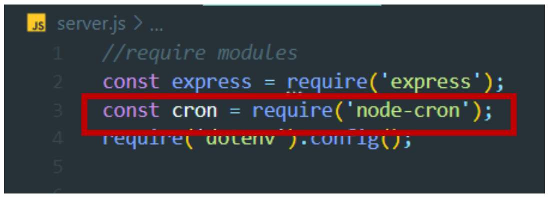
http://127.0.0.1:3000/airquality/40.42398/-3.69968

GET http://127.0.0.1:3000/airquality/40.42398/-3.69968

```
{
    "Result": {
        "Pollution": {
            "ts": "2023-02-02T13:00:00.000Z",
            "aqius": 9,
            "mainus": "p2",
            "aqicn": 3,
            "maincn": "p2"
        }
    }
}
```

## 3. Implement CRON JOB to check " air quality " for the Paris zone :

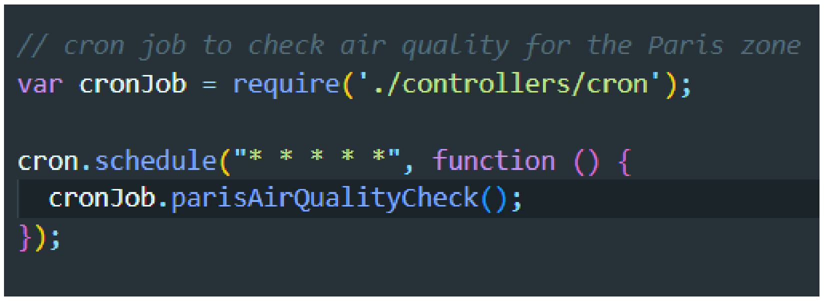
We mentioned earlier that we required the library node-cron, well that's what we're going to user to implement this task.



server.js > ...

```
//require modules
const express = require('express');
const cron = require('node-cron');
require('dotenv').config();
```

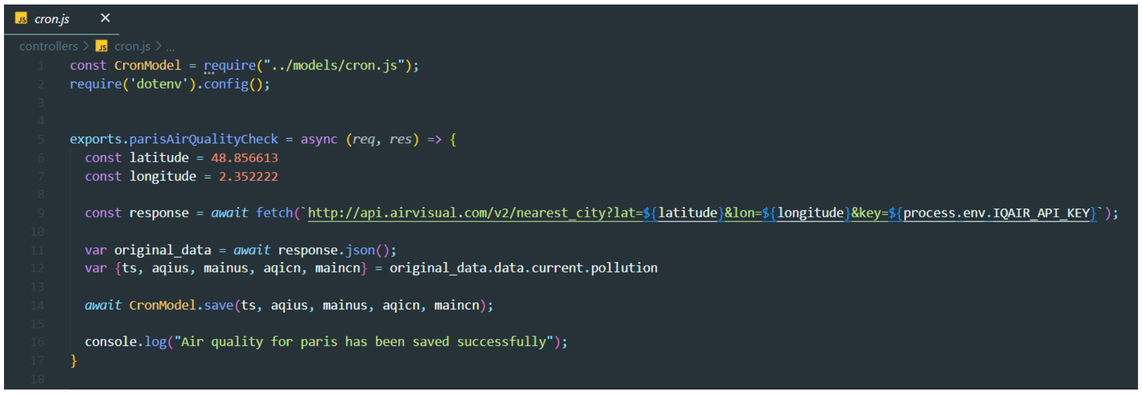
First, we grape our cron job script file which is located in **/controllers/cron.js** and we store it into a variable called **cronjob**, then we setup a cron job to execute a function within the script file **every minute**, the function called **parisAirQualityCheck**



```
// cron job to check air quality for the Paris zone
var cronJob = require('./controllers/cron');

cron.schedule("* * * * *", function () {
  cronJob.parisAirQualityCheck();
});
```

The function holds two constant values which they were given in the challenge, using the **IQAIR API** we fetch the data and we store it into a variable called **original_data**, then we create an object containing all pollution data and we call the **CronModel** model using the method **save** in order to store the data into our database, at last we print the phrase *"Air quality for paris has been saved successfully"* just so we can observe that the cron is engaging with the function every one minute.

/controllers/cron.js

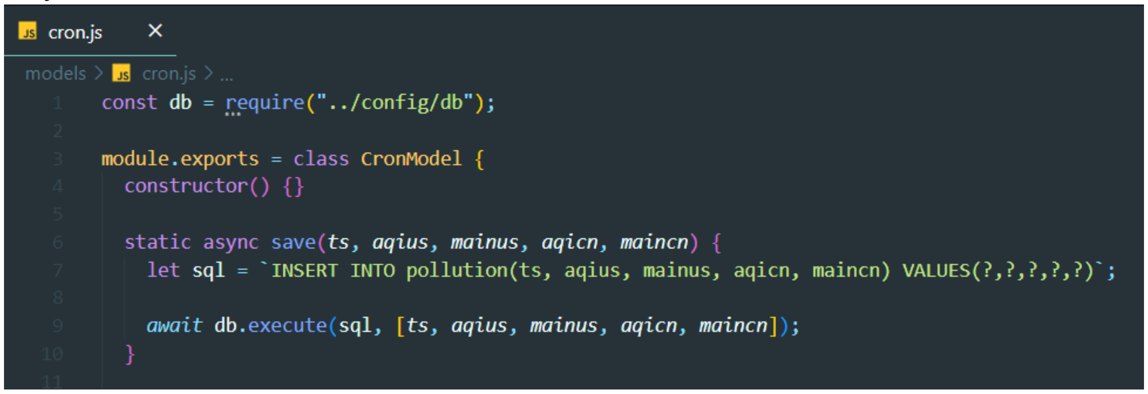


cron.js

controllers > cron.js > ...

```
const CronModel = require("../models/cron.js");
require('dotenv').config();


exports.parisAirQualityCheck = async (req, res) => {
  const latitude = 48.856613
  const longitude = 2.352222

  const response = await fetch(`http://api.airvisual.com/v2/nearest_city?lat=${latitude}&lon=${longitude}&key=${process.env.IQAIR_API_KEY}`);

  var original_data = await response.json();
  var {ts, aqius, mainus, aqicn, maincn} = original_data.data.current.pollution

  await CronModel.save(ts, aqius, mainus, aqicn, maincn);

  console.log("Air quality for paris has been saved successfully");
}
```

/models/cron.js

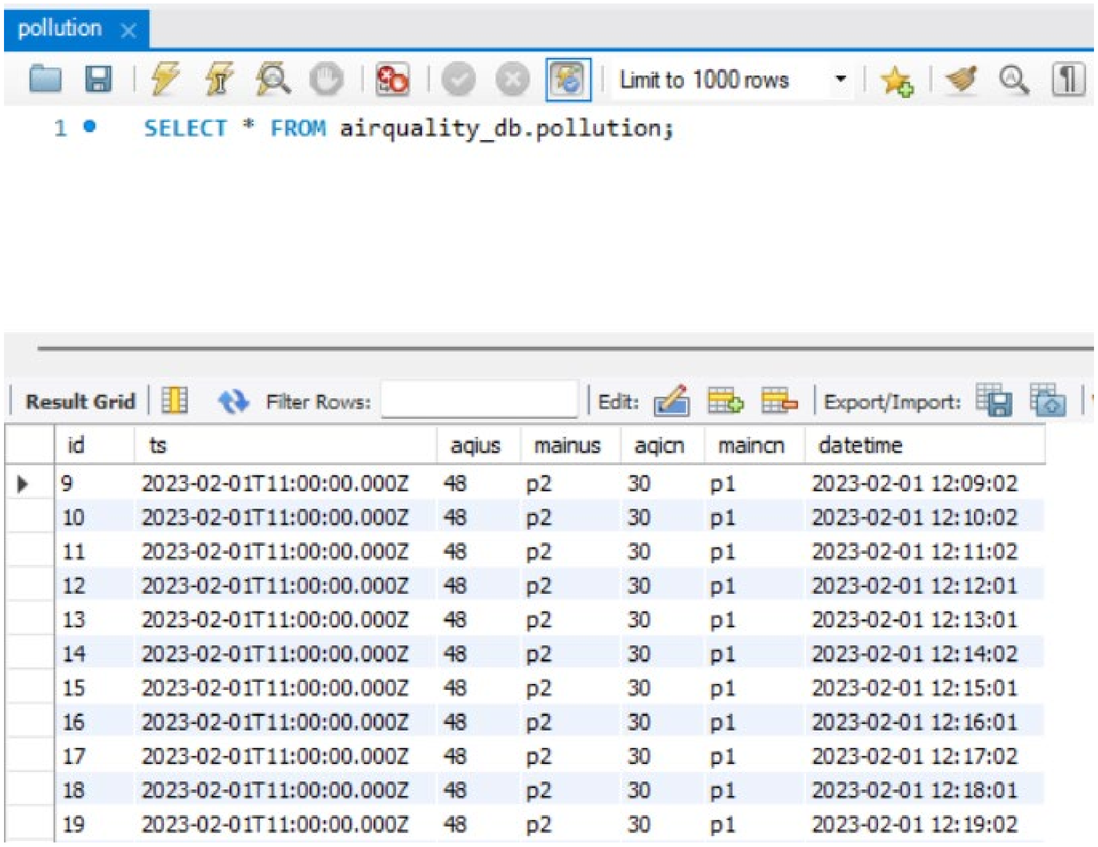


cron.js

models > cron.js > ...

```
const db = require("../config/db");

module.exports = class CronModel {
  constructor() {}

  static async save(ts, aqius, mainus, aqicn, maincn) {
    let sql = `INSERT INTO pollution(ts, aqius, mainus, aqicn, maincn) VALUES(?,?,?,?,?)`;

    await db.execute(sql, [ts, aqius, mainus, aqicn, maincn]);
  }
```

*NOTE: Regarding the database we're using* ***MySQL****, the database's name is* ***aiquality_db****, which contain a single table called* ***pollution****, the image bellow demonstrates some data saved during the development.*



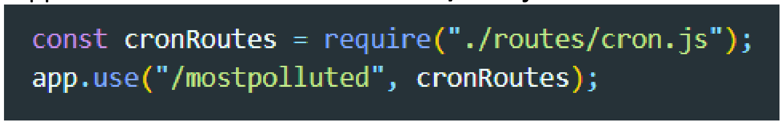
```
SELECT * FROM airquality_db.pollution;
```

| id | ts | aqius | mainus | aqicn | maincn | datetime |
|---|---|---|---|---|---|---|
| 9 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:09:02 |
| 10 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:10:02 |
| 11 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:11:02 |
| 12 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:12:01 |
| 13 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:13:01 |
| 14 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:14:02 |
| 15 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:15:01 |
| 16 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:16:01 |
| 17 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:17:02 |
| 18 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:18:01 |
| 19 | 2023-02-01T11:00:00.000Z | 48 | p2 | 30 | p1 | 2023-02-01 12:19:02 |

## 4. Get datetime(date and time ) where the Paris zone is the most polluted:

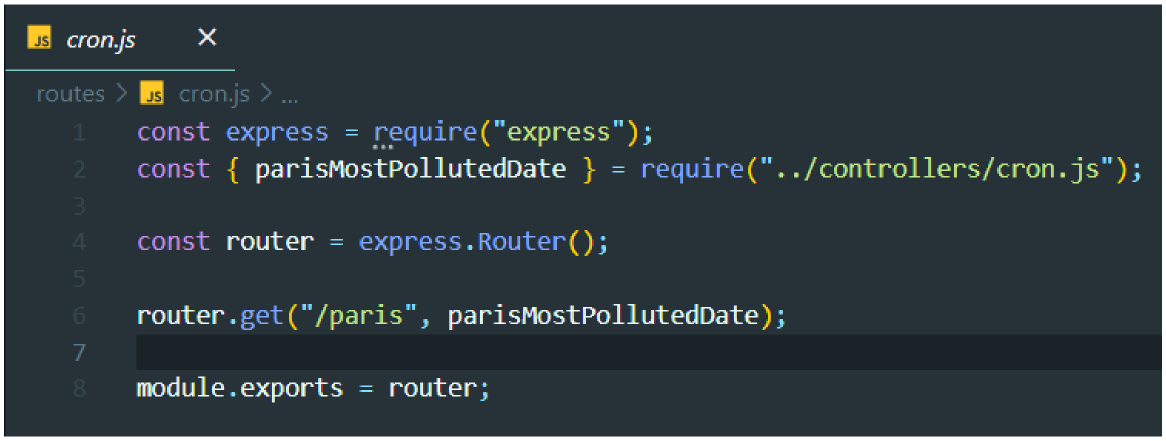
The second endpoint we created in our **server.js** is to get the date and the time where the Paris zone is the most polluted (based on the cron job results), the **URL is mapped to the route located in routes/cron.js**.

*/server.js*



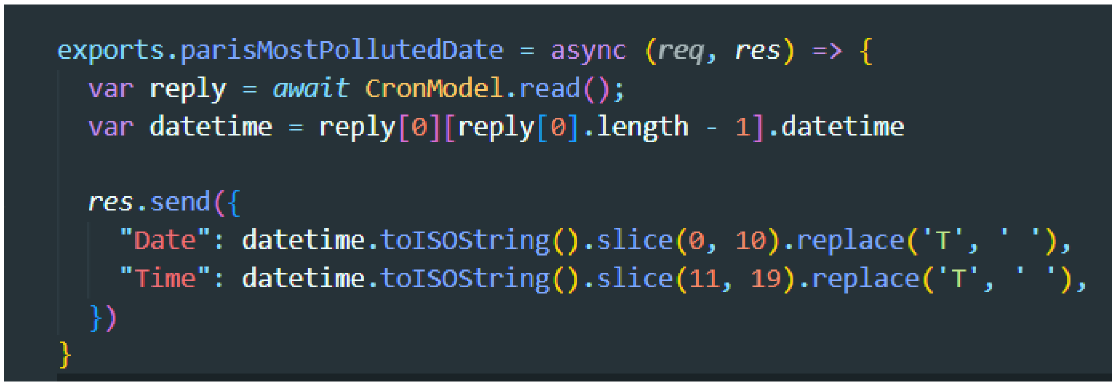
```
const cronRoutes = require("./routes/cron.js");
app.use("/mostpolluted", cronRoutes);
```

*/routes/cron.js*



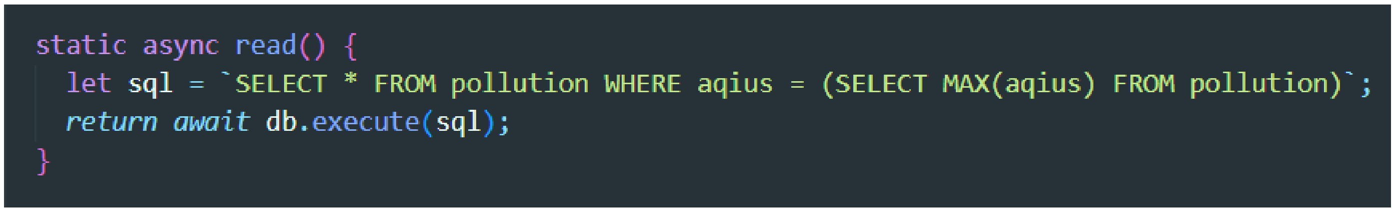
```
const express = require("express");
const { parisMostPollutedDate } = require("../controllers/cron.js");

const router = express.Router();

router.get("/paris", parisMostPollutedDate);

module.exports = router;
```

So the full URL will be **/mostpolluted/paris,** under this url we have a function called **parisMostPollutedDate** which is located in **/controllers/cron.js**



```
exports.parisMostPollutedDate = async (req, res) => {
  var reply = await CronModel.read();
  var datetime = reply[0][reply[0].length - 1].datetime

  res.send({
    "Date": datetime.toISOString().slice(0, 10).replace('T', ' '),
    "Time": datetime.toISOString().slice(11, 19).replace('T', ' '),
  })
}
```

In this function we call the **CronModel** model located in **/models/cron.js** using the method **read** which will connect to the database and return a list of objects filtered by the maximum value of **aqius,** which is an **US AQI (EPA)**



```
static async read() {
  let sql = `SELECT * FROM pollution WHERE aqius = (SELECT MAX(aqius) FROM pollution)`;
  return await db.execute(sql);
}
```

then we retrieve the **datatime** value and send it separately for easy read, the result is down below:

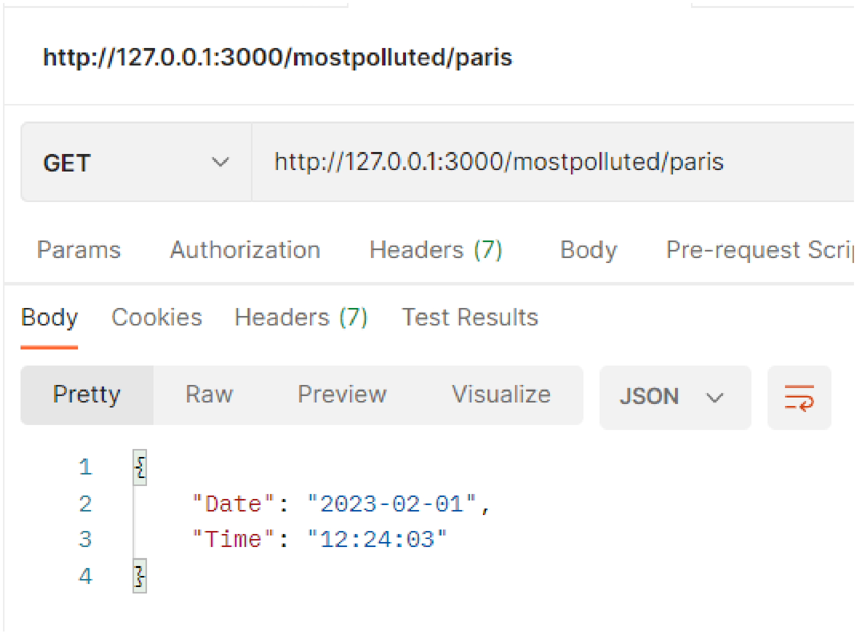
**Result:**

**Request:** http://127.0.0.1:3000/mostpolluted/paris

**Response:**

```
{
  "Date": "2023-02-01",
  "Time": "12:24:03"
}
```

***Exemple***



```
{
    "Date": "2023-02-01",
    "Time": "12:24:03"
}
```

Start server command : **npm start**