UbiLab AirVisual API Assignment

Juan Pablo Mascaretti – 16/12/2020

Background

For this assignment, I was given documentation about the AirVisual API. Briefly, the AirVisual API allows users to access global air quality data that can be obtained for any location on Earth. From this data one can study air conditions such as the concentration of pollutants in the air, temperature, carbon dioxide levels (CO2), humidity, etc. and thus derive insightful trends from the obtained data.

I was also given a sample of raw data obtained from a real AirVisual Pro device as follows:

https://www.airvisual.com/api/v2/node/8CCaRCP4hA6cCjfwp

The two main goals of this assignment were:

- To visually describe the structure of the response data from the link above by drawing a chart.
- To find out the number of humidity measurements taken within one hour on 1am Dec 16th, 2020.

Below, the reader will be able to find an explanation to the followed approach, as well as the obtained results that demonstrate how this assignment was successfully solved.

Approach – Goal #1

As mentioned before, the raw sample data was given, and the first goal was to visually describe the structure of the response data from the link above by drawing a chart. The first step was to look at the data to understand the size and characteristics of the data. A snapshot is shown below:


```
{"ts":"2020-12-16T22:45:00.726Z","tp":3.2,"hm":81,"p2":25,"p1":25,"co":400,"errors":
{"hcho":-205,"voc":-202},"p01":25},{"ts":"2020-12-
16T22:40:00.726Z","tp":3.3,"hm":81,"p2":24,"p1":24,"co":402,"errors":{"hcho":-205,"voc":-202},"p01":24},
{"ts":"2020-12-16T22:35:01.726Z","tp":3.3,"hm":81,"p2":29,"p1":29,"co":402,"errors":
{"hcho":-205,"voc":-202},"p01":25},{"ts":"2020-12-
16T22:30:00.937Z","tp":3.4,"hm":81,"p2":30,"p1":30,"p01":24,"co":402,"errors":{"hcho":-205,"voc":-202}},
{"ts":"2020-12-16T22:25:01.032Z","tp":3.4,"hm":81,"p2":26,"p1":26,"p01":25,"co":403,"errors":
{"hcho":-205,"voc":-202}},{"ts":"2020-12-
16T22:20:00.832Z","tp":3.4,"hm":81,"p2":25,"p1":25,"p01":24,"co":401,"errors":{"hcho":-205,"voc":-202}},
{"ts":"2020-12-16T22:15:00.845Z","tp":3.4,"hm":81,"p2":24,"p1":24,"co":402,"errors":
{"hcho":-205,"voc":-202},"p01":23},{"ts":"2020-12-
16T22:10:00.845Z","tp":3.5,"hm":81,"p2":28,"p1":29,"co":401,"errors":{"hcho":-205,"voc":-202},"p01":23},
{"ts":"2020-12-16T22:05:01.006Z","tp":3.5,"hm":82,"p2":26,"p1":26,"p01":23,"co":402,"errors":
{"hcho":-205,"voc":-202}},{"ts":"2020-12-16T22:00:00.898Z","tp":3.5,"hm":81,"p2":25,"p1":25,"co":400,"errors":
{"hcho":-205,"voc":-202},"p01":25},{"ts":"2020-12-
16T21:55:00.898Z","tp":3.6,"hm":81,"p2":22,"p1":22,"co":401,"errors":{"hcho":-205,"voc":-202},"p01":22},
{"ts":"2020-12-16T21:50:01.038Z","tp":3.6,"hm":81,"p2":23,"p1":23,"p01":22,"co":400,"errors":
```


Figure 1. Snapshot of raw data.

As it can be observed in **Figure 1,** the raw data is very hard to read, so proper formatting is needed to facilitate the understanding of the structure and the fields. Luckily, the API references a tool called *Postman* that allows to easily get and format the data from the given API. Thus, I followed the instructions and pasted the link to get the formatted JSON object. This can be seen below:

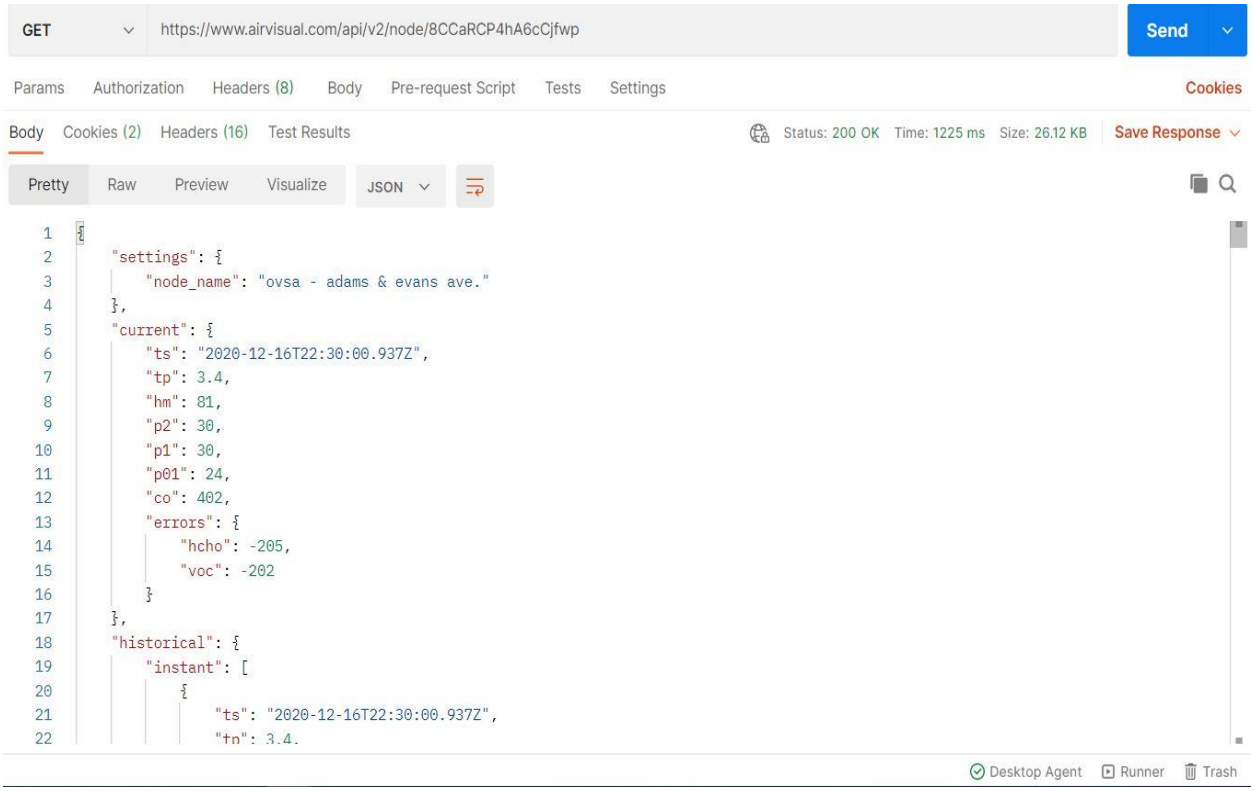

Figure 2. Snapshot of formatted data.

As it can be observed in **Figure 2,** the data looks much more readable, and can now be saved through the "Save Response" button as the formatted JSON type. To avoid complex code, I simply downloaded the file in a local directory and uploaded it to a local server to be analyzed. Thus, I saved the file as airvisual.json and stored it in a local folder.

Now that the data was successfully formatted, the next step was to understand the data structure. Thus, the file needed to be uploaded to a localhost server and then output in the console. To complete this step, I used the npm package along with the *lite server* npm module from Node.JS. Immediately after the server was running locally, I created an HTML file to visually display any data, as well as a JavaScript file to write the logic for any data processing and analysis.

Subsequently, I fetched the JSON data in JavaScript using promises and I logged the data in the console. This allowed me to understand the formatted data structure, which will be explored and explained in the results section below.

Results – Goal #1

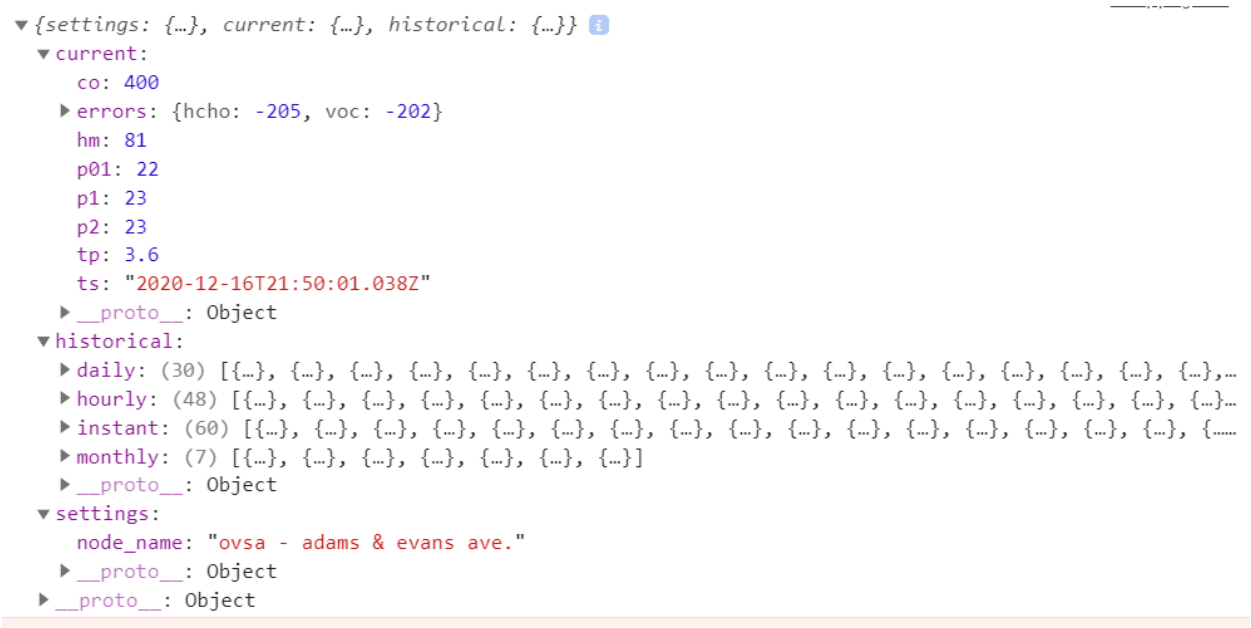

Figure 3. Console log of formatted data (JSON object).

As it can be seen in **Figure 3** above, a JSON object is returned by the API, mainly consisting of 3 objects with subfields. The first object seems to be the current set of measurements at a given time, the second object is a log of historical measurements by month, day, hour, and instant (measurements obtained every 5 minutes in the same day, since 4:55 pm). The third object contains the general settings of the data log.

The overall data structure is better represented visually in **Figure 4** below. Please note that the rectangles in the diagram represent an array, while an ellipse represents an object.

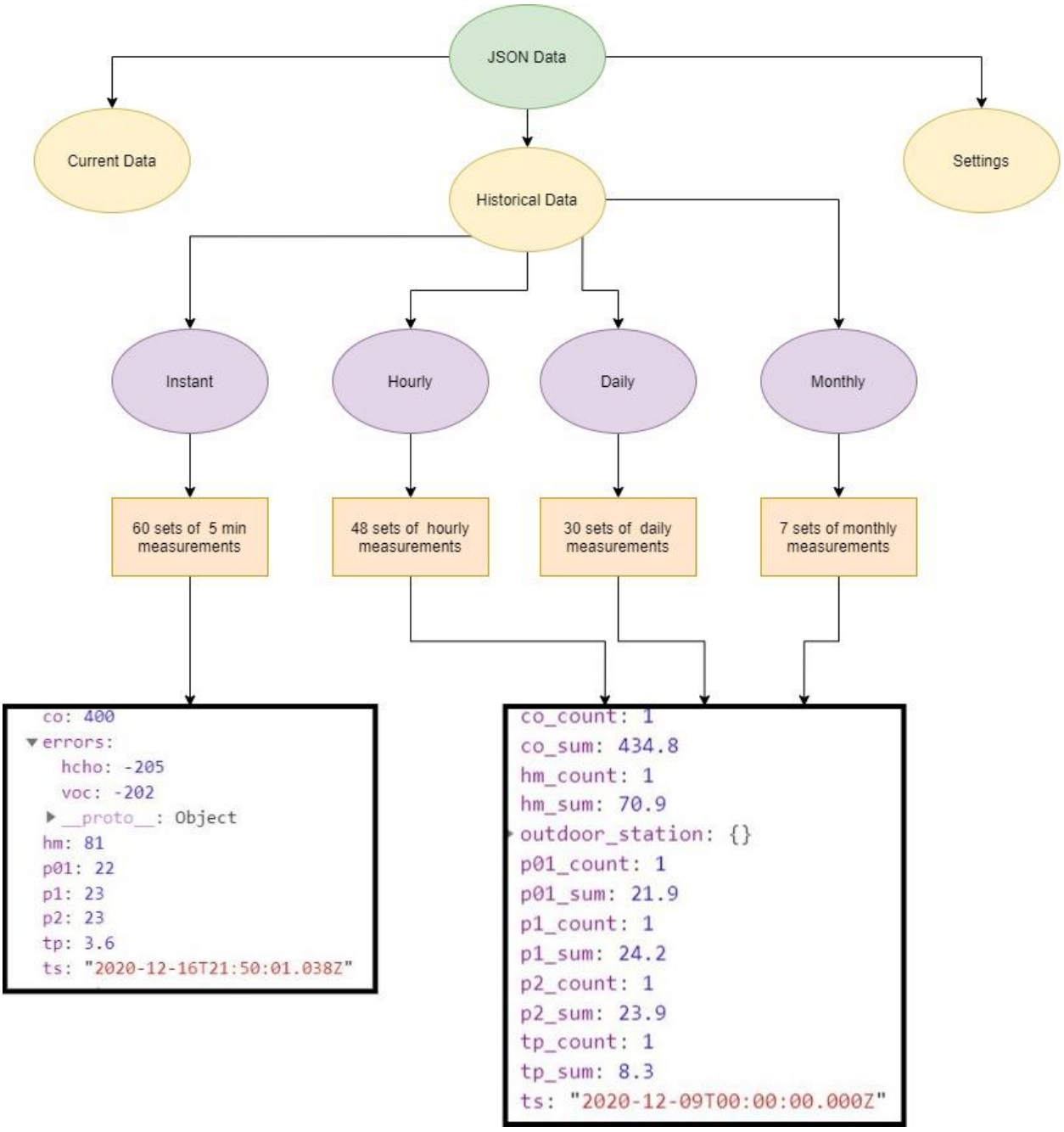

Figure 4. Diagram of data structure.

As it can be seen on Figure 4, the JSON data object has 3 objects, out of which the historical data is the only relevant one for our further analysis. This historical object has 4 sub-objects containing: instant measurements taken every 5 minutes, hourly measurements, daily measurements, and monthly measurements. These measurements are stored in arrays that contain sets of various climatic measurements. Each set in the array is an object with a key-value pair containing a detailed set of measurements for a given timestamp. I believe that with this understanding of the data, it is now worth looking at the approach and the results of the second goal, which are detailed below.

Approach – Goal #2

To recapitulate, the second goal was to find out the number of humidity measurements taken within one hour on 1am Dec 16th, 2020. Since the dataset is not very large, I manually looked at the data in the console. However, a better approach would have been to query everything by looping through the objects/arrays, while filtering for any desired timestamps.

The first approach was to look at the instant data, since the "current" object indicated that the data was taken on December 16th at `21:50:01.` Nonetheless, the instant measurements taken every 5 minutes only displayed data for that same day from `16:55:01 - 21:50:01` with no records of data sampled at 1 am every 5 minutes on that day. Since the day when the data was sampled was December 16th, the next step was to look at hourly data. By briefly looking into the daily and monthly fields I realized that I only needed to look at the hourly data, as I could locate the measurements that took place at 1 am. The results are shown below.

Results – Goal #2

Just by simple observation I realized that entry number 19 (20th element) in the hourly array displayed the set of measurements taken at 1 am on December 16th. This raw output is shown below:

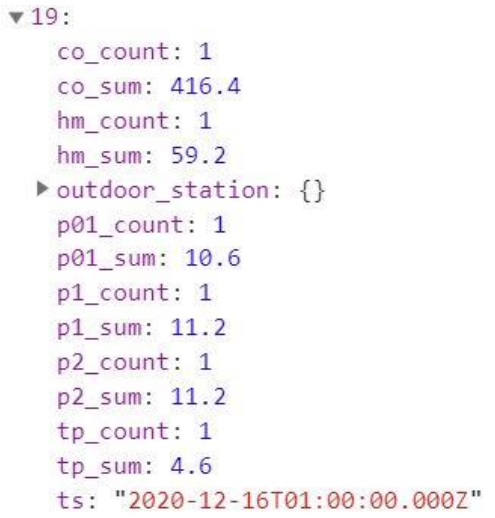

Figure 5. Raw desired output

By looking at the documentation, I realized that hm_count displays the number of humidity measurements for a given time. Thus, it is possible to conclude that only one humidity measurement was sampled at 1am on December 16th, 2020. For easy visualization of the output, I converted the JSON data for this log into a string by using JavaScript, and finally displayed it in the HTML file run by a local server. The final formatted output can be seen in **Figure 6** below:

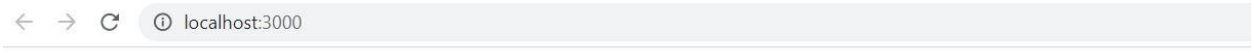

Figure 6. Formatted final output.

Conclusion

Both goals were achieved through the approach followed in this document. Better practices could be implemented for larger, real-world data, such as storing the data from the API in a secure database, while using more robust programming logic to query any desired results. A formatting document could also be created according to user specifications rather than using the Postman tool.

Finally, although the data was stored locally as a sample of data was given, a better approach for working with this type of atmospheric data in a real scenario would be to create various AirVisual API keys and to add them to the code attached to this document. This would have several advantages such as:

- Obtaining data simultaneously from multiple sources, reducing bias.
- Choosing the desired location to measure results and compare it to nearby locations for better insights.
- Sampling data every 5 minutes or less to get more measurements per day, potentially reducing measurement errors while allowing to accurately visualizing trends in the data.
- Comparing it with other API's to check the validity and the accuracy of the obtained results