Monitoring our city
VehiCloud

Baptiste Lerat (AE) - Ewan Mackay (AE) - Grégoire Hebras (MsIoT)
Abir Benazzouz (IR) - Florian Convert (AE)

INSA

I - Problem Statement

II - Objectives

III - Data collection

IV - Data presentation

V - Casing

VI - Conclusion

I-PROBLEM STATEMENT

Atmospheric pollution in France

Pech David - Toulouse

48,000 premature deaths per year, 9% of all deaths in France*.

A total annual health cost of 100 billion euros**.

30% of the population affected by a respiratory allergy***.

*Santé Publique France ** Report 610 of the Sénat ***R.N.S.A

Solving this problem takes planning, time and money.

Our idea

Accelerate change by making the issue visible & understandable

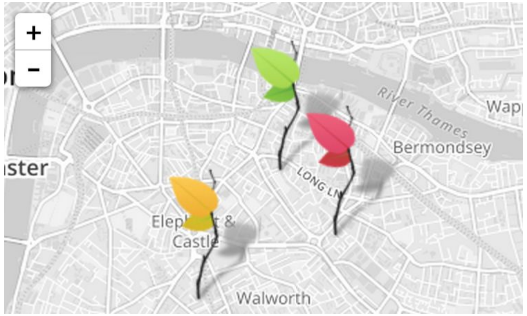
We want to show decision makers where this pollution is

II-OBJECTIVES

The solution

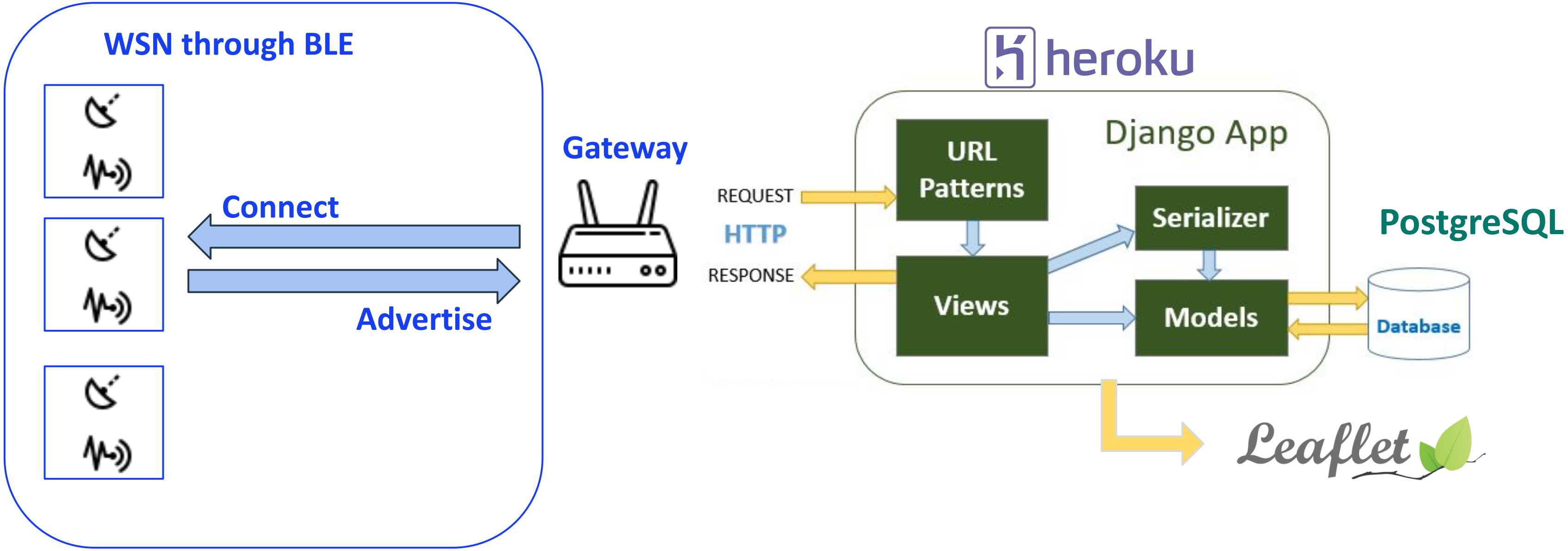
Moving sensors on the bike (ESP32)

Autonomous fixed stations as gateway (Raspberry Pi)



Web application (RestAPI|SQL|Leaflet)

The IoT architecture



III-DATA COLLECTION

Specifications of sensor node

Device composed of 3 sensors

Temperature:

- DHT11
- Digital
- 0 to 50°C

Humidity:

- DHT11
- Digital
- 20 to 90%

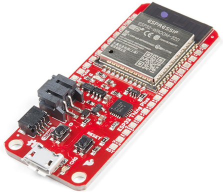
Gas:

- multichannel gas sensor
- I2C
- CO, NO2, SO2

GPS:

- Adafruit Ultimate GPS
- UART
- < 5s meters

Controlled by a microcontroller (ESP32)

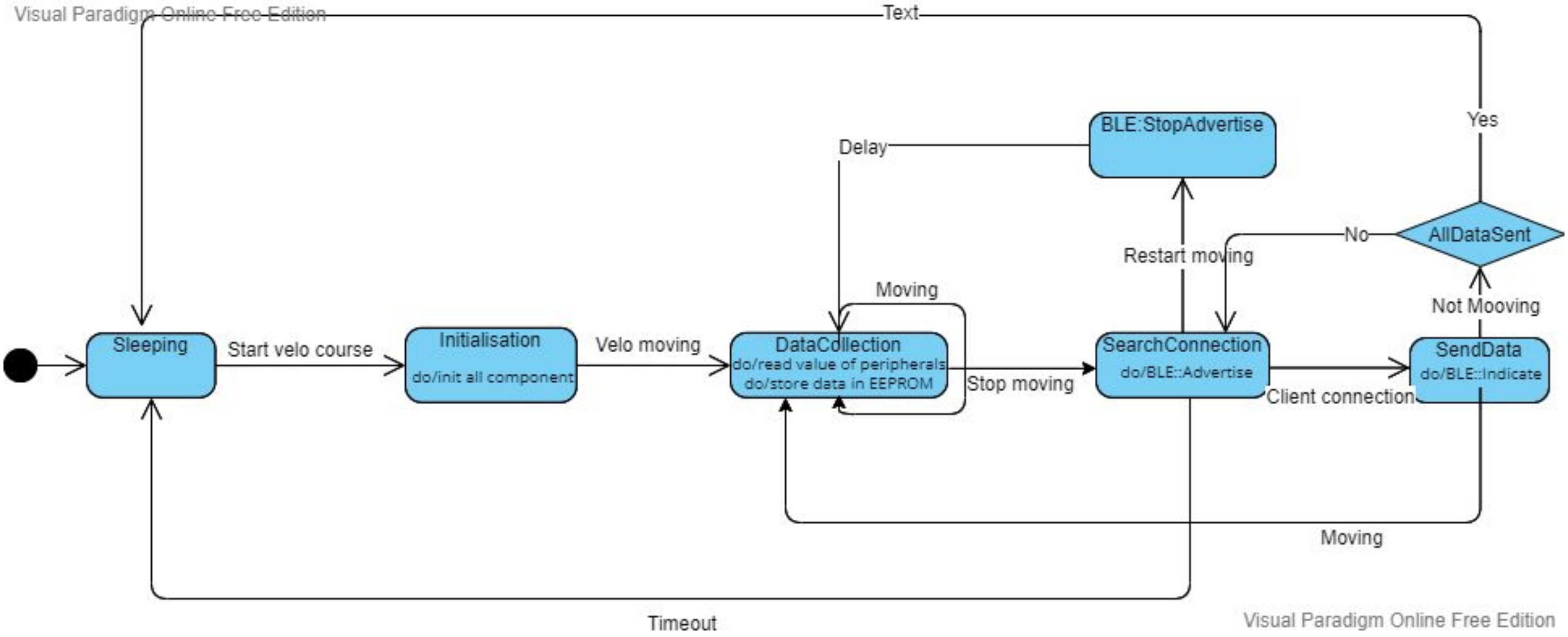


- Actuator: button
- EEPROM as storage

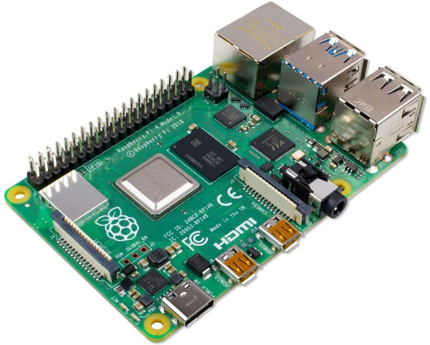
Communication with gateway through BLE
ESP as a BLE Server

- Advertise
- 1 characteristic with all data
- Send all stored data at connection
- Restart after transmitting data

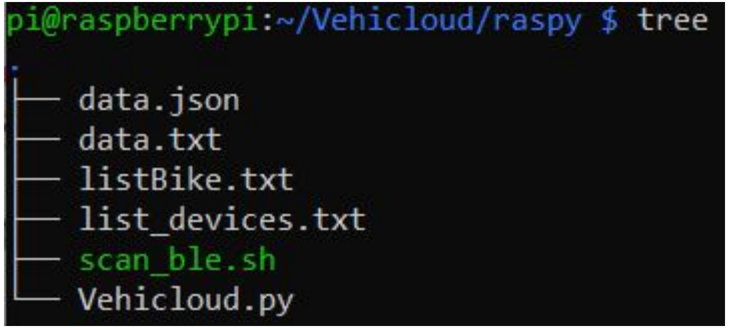
Firmware state diagram



III The gateway



Raspberry Pi4



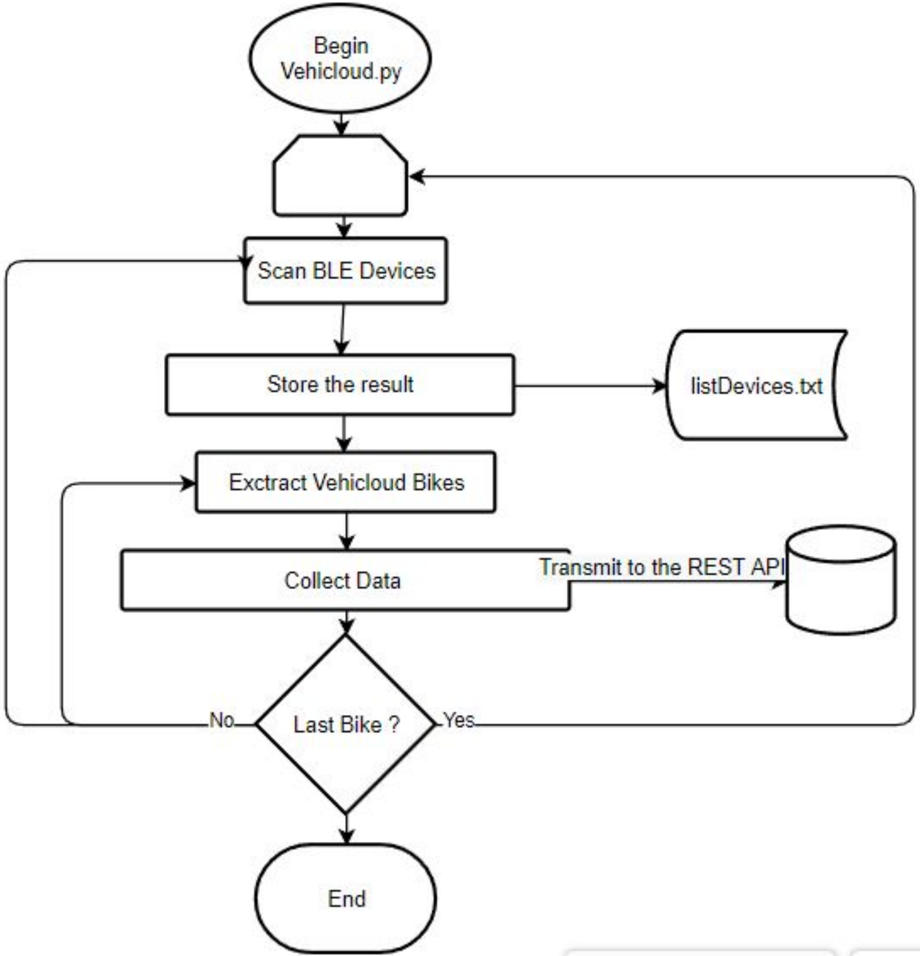
```
pi@raspberrypi:~/Vehicloud/raspy $ tree
.
├── data.json
├── data.txt
├── listBike.txt
├── list_devices.txt
├── scan_ble.sh
└── Vehicloud.py
```

Architecture

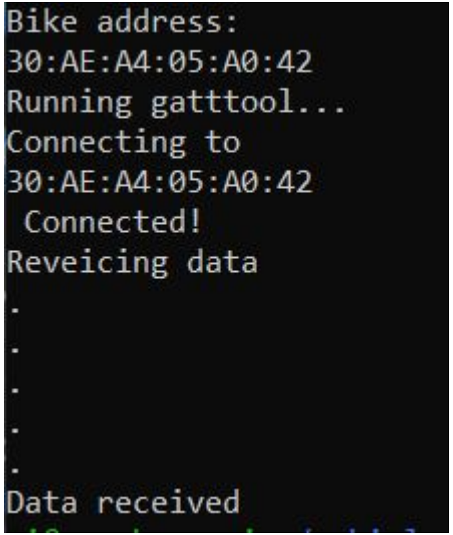
The gateway

Tools used:

- **Bluez**: native on Linux kernel
- **pexpect**: permit to spawn a child application and control it as if a human were typing commands
- **JSON**: to make JSON file in Python
- **requests**: to use the http requests in Python



The gateway

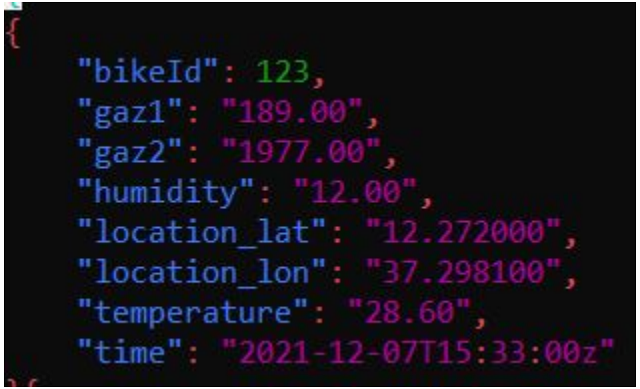


```
Bike address:
30:AE:A4:05:A0:42
Running gatttool...
Connecting to
30:AE:A4:05:A0:42
 Connected!
Reveicing data
.
.
.
.
.
Data received
```

Bluez

- hcitool lescan: scan all BLE devices
- gatttool -I <MAC address>: connect to a device
- char-write-req <handle> <characteristic>

Data format



```
{
    "bikeId": 123,
    "gaz1": "189.00",
    "gaz2": "1977.00",
    "humidity": "12.00",
    "location_lat": "12.272000",
    "location_lon": "37.298100",
    "temperature": "28.60",
    "time": "2021-12-07T15:33:00z"
}
```

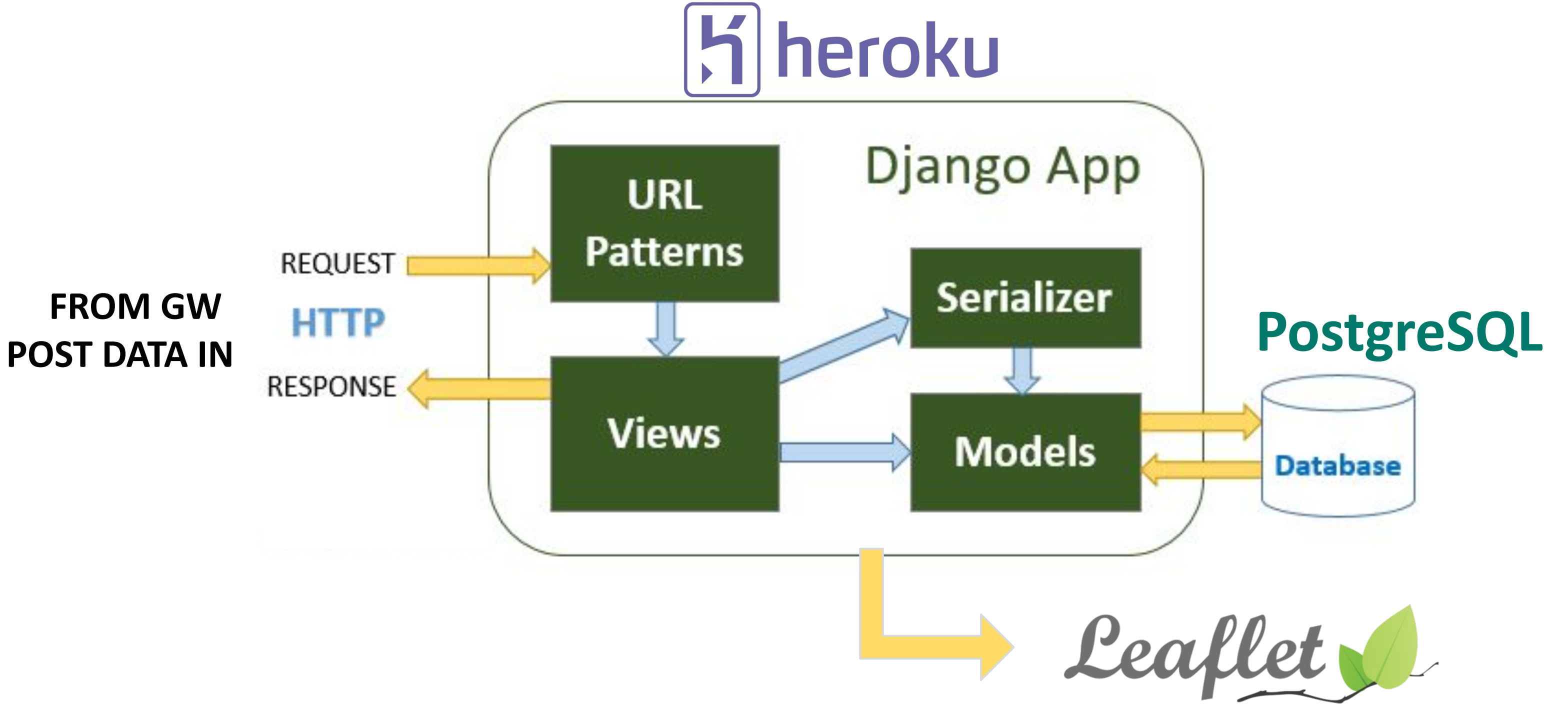
JSON and **requests** Python library:

- Convert data in Hexadecimal to JSON file
- Post the JSON file to the API

IV-DATA PRESENTATION

IV

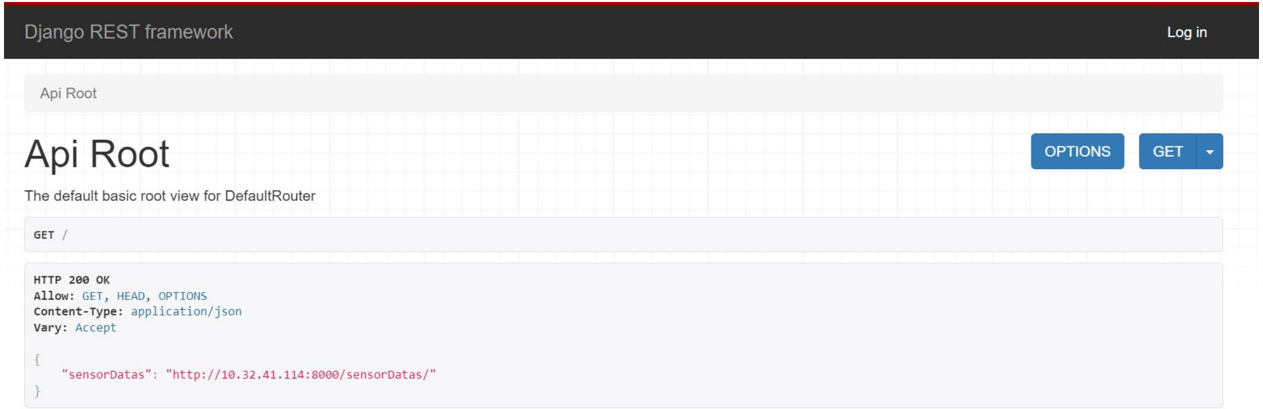
Our backend architecture



Front end

IV

Our API is built with Django



Api Root

The default basic root view for DefaultRouter

```
GET /
```

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "sensorDatas": "http://10.32.41.114:8000/sensorDatas/"
}
```

http://<server-IP>:8000/ <- main menu with hyperlink

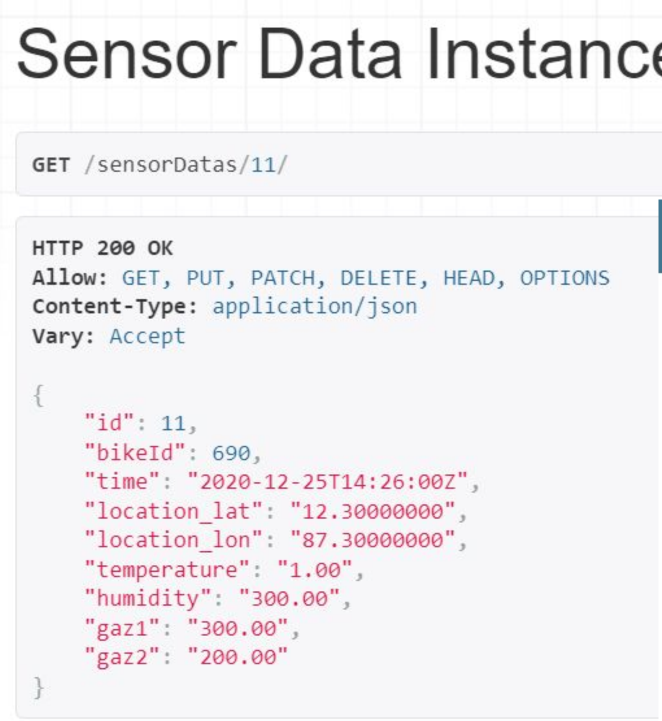
http://<server-IP>:8000/sensorDatas/ <- all data received

http://<server-IP>:8000/sensorDatas/<message-ID> <- individual messages with a unique ID

http://<server-IP>:8000/admin/ <- administration and log in

IV

Work done on the API

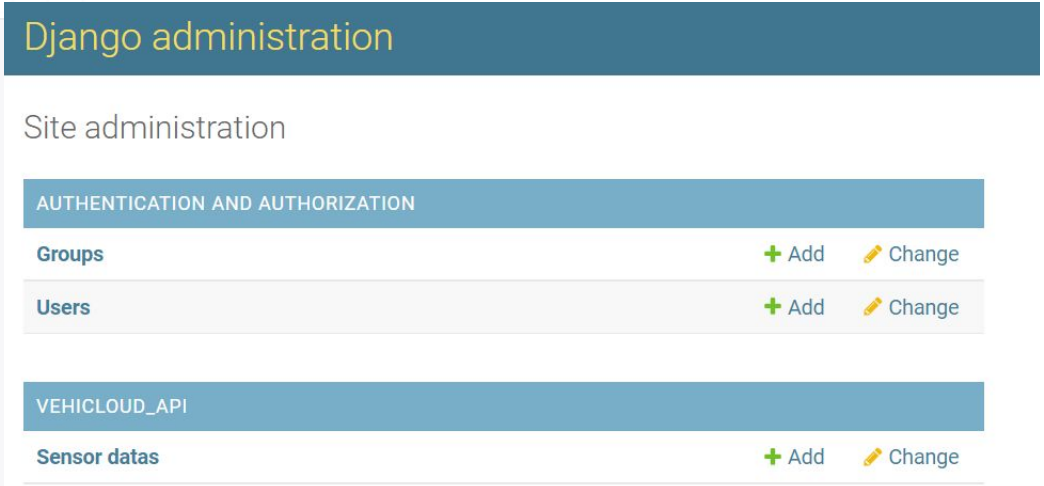


Sensor Data Instance

```
GET /sensorDatas/11/
```

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 11,
    "bikeId": 690,
    "time": "2020-12-25T14:26:00Z",
    "location_lat": "12.30000000",
    "location_lon": "87.30000000",
    "temperature": "1.00",
    "humidity": "300.00",
    "gaz1": "300.00",
    "gaz2": "200.00"
}
```



Django administration

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups + Add ✎ Change

Users + Add ✎ Change

VEHICLOUD_API

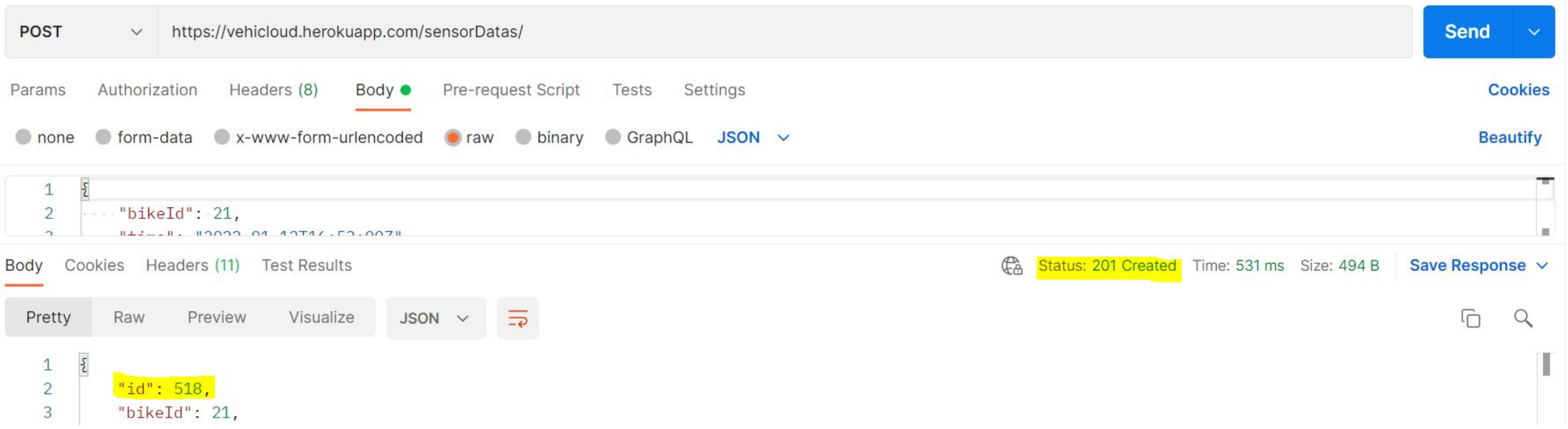
Sensor datas + Add ✎ Change

IV Deploying our RESTful API to the web

heroku

Free hosting of our Web API!

Our new domain: vehicloud.herokuapp.com



POST https://vehicloud.herokuapp.com/sensorDatas/

```
{
    "bikeId": 21,
```

Status: 201 Created Time: 531 ms Size: 494 B

```
{
    "id": 518,
    "bikeId": 21,
```

IV

How do we store our data? With a database!

PostgreSQL

- Persistent memory
- 10,000 messages stored for free
- Easy integration to our Django app

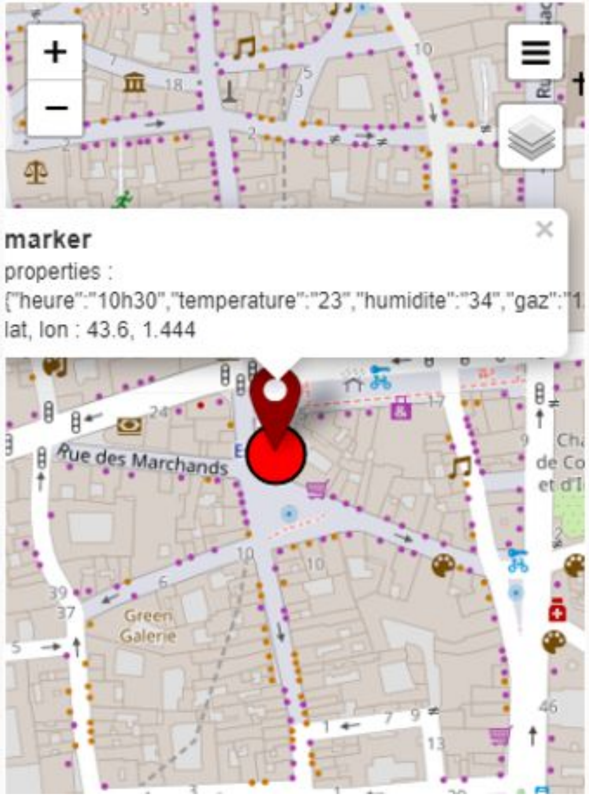
Implementation of SQL requests to:

- Isolate the data of interest (temperature, humidity, variations, etc.) to be displayed
- Exploit and stock the previous data for ulterior analysis

IV

First attempt at mapping data:

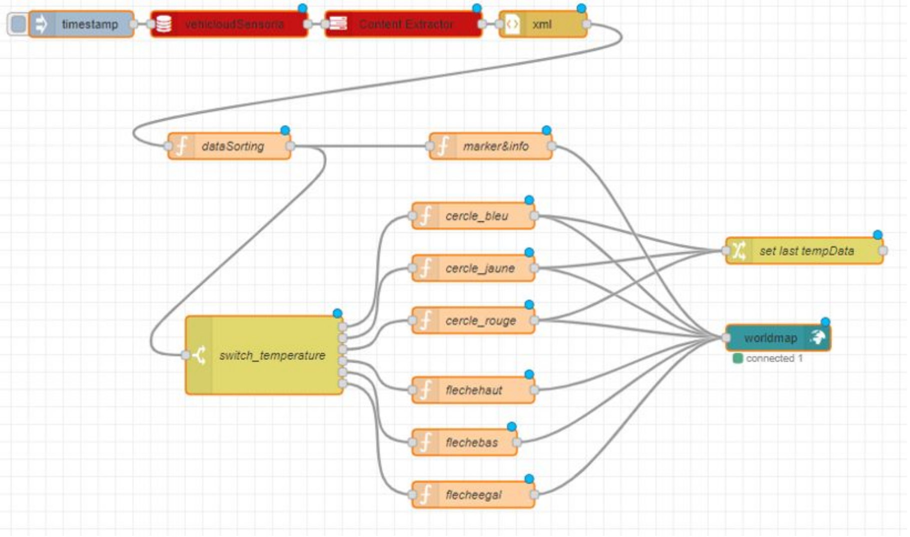
Node-RED



marker

properties : {"heure":"10h30","temperature":"23","humidite":"34","gaz":"1

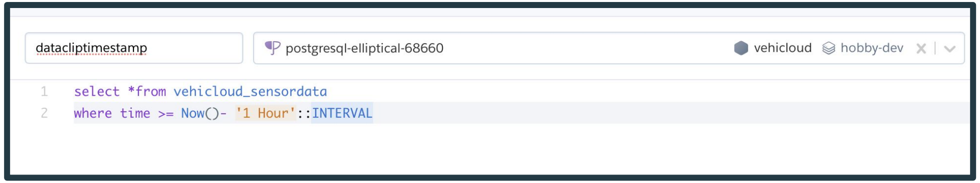
lat, lon : 43.6, 1.444



Our solution to retrieve data from the database

Use of Heroku DataClips

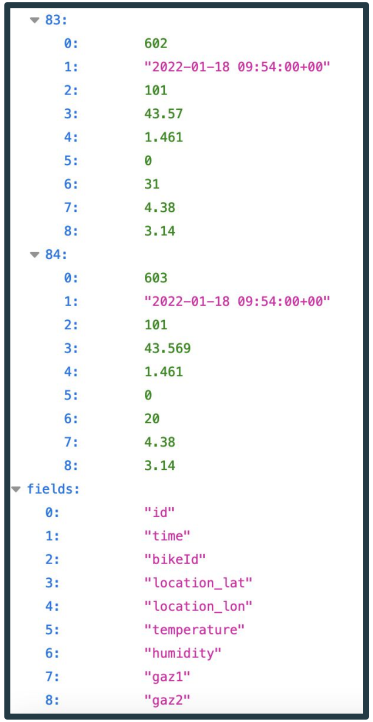
→ Retrieval of the most recent data



datacliptimestamp | postgresql-elliptical-68660 | vehicloud | hobby-dev

```
select *from vehicloud_sensordata
where time >= Now()- '1 Hour'::INTERVAL
```

→ Download of the result as a JSON file



```
▼ 83:
    0:      602
    1:      "2022-01-18 09:54:00+00"
    2:      101
    3:      43.57
    4:      1.461
    5:      0
    6:      31
    7:      4.38
    8:      3.14
▼ 84:
    0:      603
    1:      "2022-01-18 09:54:00+00"
    2:      101
    3:      43.569
    4:      1.461
    5:      0
    6:      20
    7:      4.38
    8:      3.14
▼ fields:
  0:        "id"
  1:        "time"
  2:        "bikeId"
  3:        "location_lat"
  4:        "location_lon"
  5:        "temperature"
  6:        "humidity"
  7:        "gaz1"
  8:        "gaz2"
```

IV

Data display: Use of a Leaflet heatmap

Leaflet

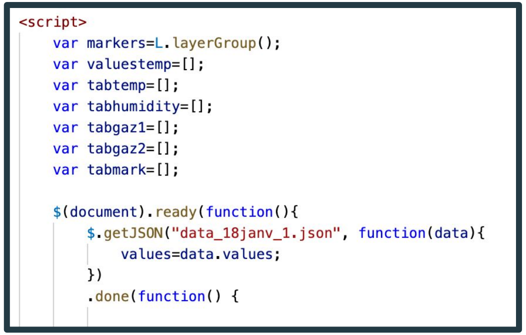
(JavaScript Library)

JSON object:

- ❑ Values
    - ❑ Id
    - ❑ Timestamp
    - ❑ Bikeld
    - ❑ Latitude
    - ❑ Longitude
    - ❑ Temperature
    - ❑ Humidity
    - ❑ Gaz 1
    - ❑ Gaz 2

➢ **Display of the data: heatmap**

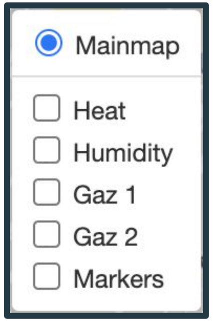
➢ **Each layer: dedicated heatmap**



```
<script>
    var markers=L.layerGroup();
    var valuestemp=[];
    var tabtemp=[];
    var tabhumidity=[];
    var tabgaz1=[];
    var tabgaz2=[];
    var tabmark=[];

    $(document).ready(function(){
        $.getJSON("data_18janv_1.json", function(data){
            values=data.values;
        })
        .done(function() {
```

Upload of the JSON file to an html script



● Mainmap

□ Heat

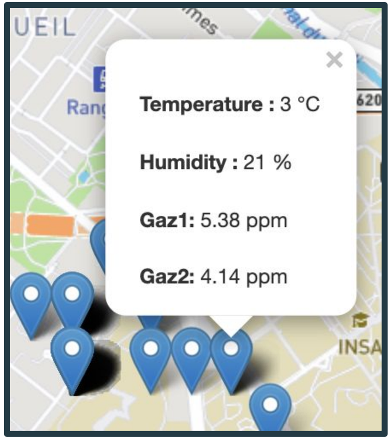
□ Humidity

□ Gaz 1

□ Gaz 2

□ Markers

<u>Different layers</u>



Temperature : 3 °C

Humidity : 21 %

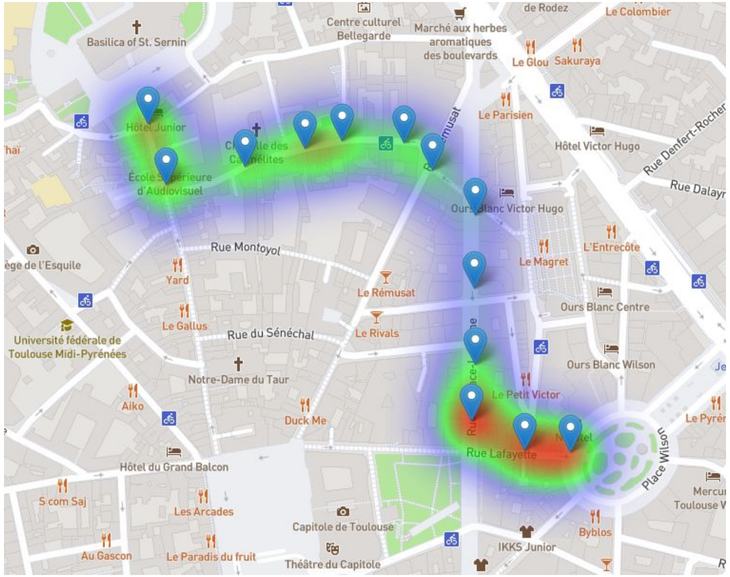
Gaz1: 5.38 ppm

Gaz2: 4.14 ppm

<u>The Marker layer</u>

IV

Demonstration of the Leaflet interface at the end of the presentation !

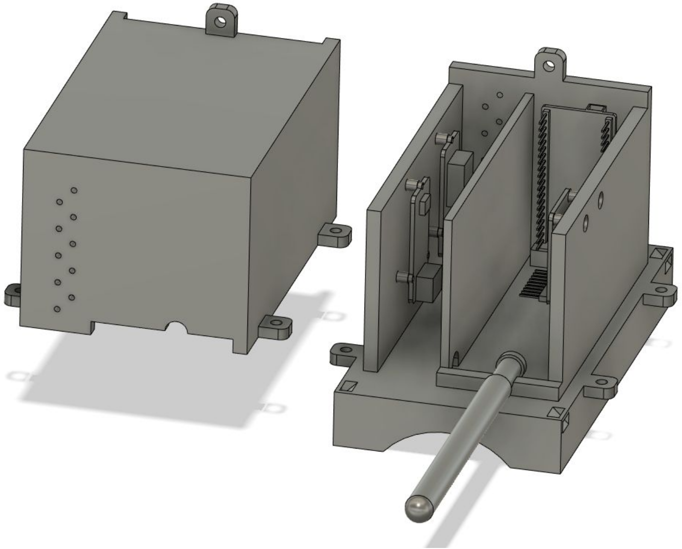


V-CASING

Prototyping: CAD design

AUTODESK® FUSION 360

- ➢ Modern and Intuitive
- ➢ Free license for hobbyists
- ➢ Ideal for 3D printing



Environmental constraints:

- Air humidity
- Vibrations and shocks

Thank you for your attention

We are now ready for a demonstration.