

# **M2TIW-INT SIG tp noté**

## **Antennes au Colorado et leurs couvertures**

Étudiant1 -> Nom : MOLINARES VALENCIA Prénom : Diogenes Numéro :  
p2019196

Étudiant2 -> Nom : BUNEL Prénom : Maxime Numéro : p1914012

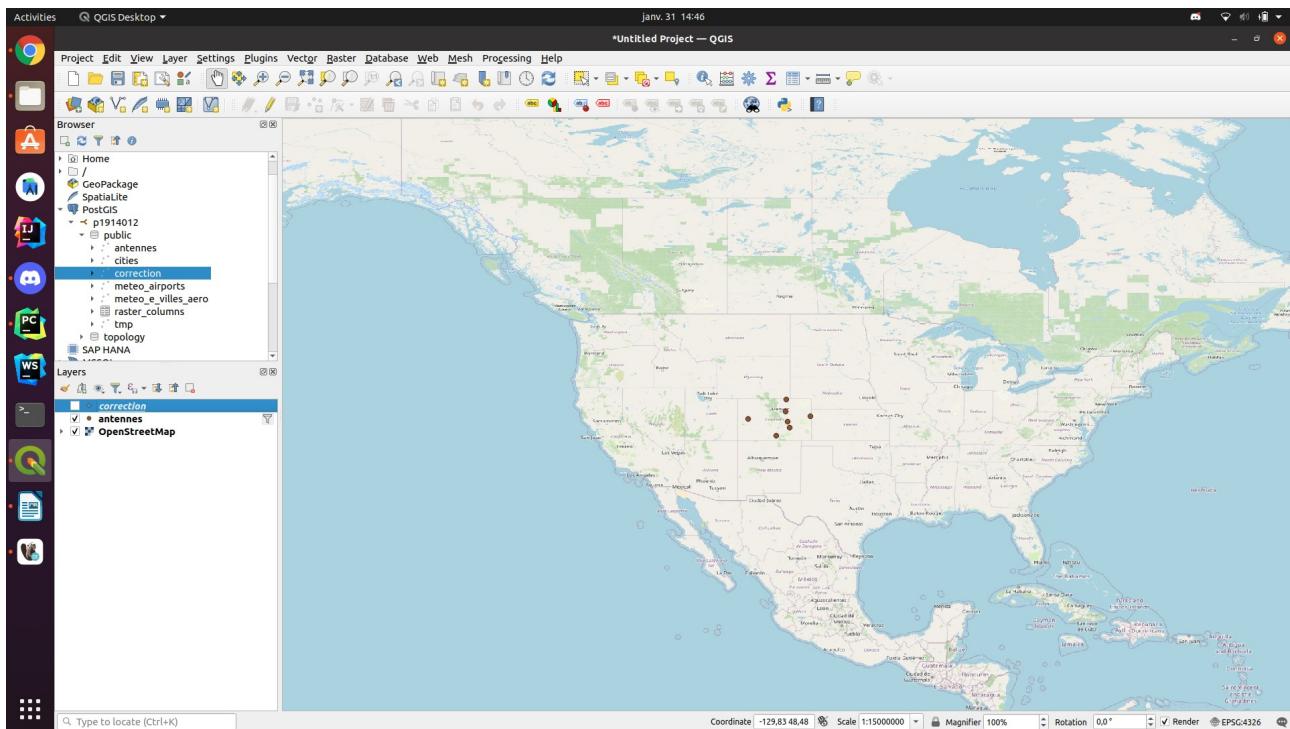
Étudiant3 -> Nom : GIRAUD Prénom : Julien Numéro : p1704709

## a) Filtrage sous QGis

a.1) Copie d'écran IMAGE "a\_filtrage\_commande" : 1 copie d'écran de QGis montrant la commande QGis qui a permis de filtrer les antennes

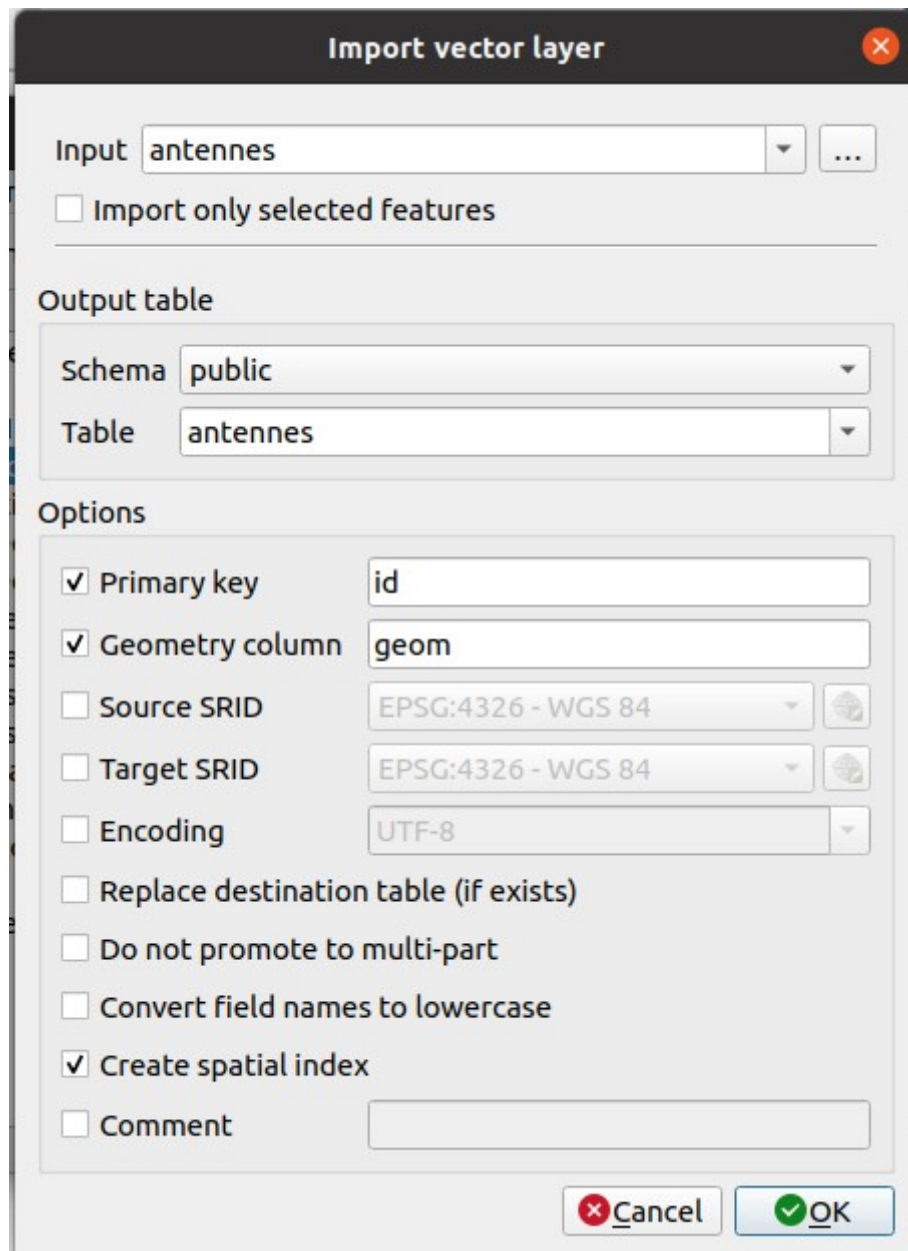
The screenshot shows the 'Query Builder' dialog box in QGIS. The title bar reads 'Query Builder'. The main area is titled 'Set provider filter on antennes'. It is divided into two main sections: 'Fields' and 'Values'. The 'Fields' section on the left lists the following fields: id, STFIPS, STATE, STPOSTAL, VERSION, and REVISION. The 'Values' section on the right has a search bar with the text 'Search...' and a large empty text area below it. Below the search bar are two buttons: 'Sample' and 'All'. There is also a checkbox labeled 'Use unfiltered layer' which is currently unchecked. Below these sections is a section titled 'Operators' with a dropdown arrow and a grid of operators: '=', '<', '>', 'LIKE', '%', 'IN', 'NOT IN', '<=', '>=', '!=', 'ILIKE', 'AND', 'OR', and 'NOT'. At the bottom is a section titled 'Provider Specific Filter Expression' with a large text area containing the expression: `"STATE" = 'COLORADO'`. The bottom of the dialog has a row of buttons: 'Help' (with a question mark icon), 'Test', 'Clear', 'Save...', 'Load...', 'Cancel' (with a red X icon), and 'OK' (with a green checkmark icon).

**a.2) Copie d'écran IMAGE "a\_filtrage\_resultat" : 1 copie d'écran de QGIS montrant la couche filtrée avec en fond de carte la mappemonde OpenStreetMap centrée sur les USA à l'échelle 1/15 000 000**

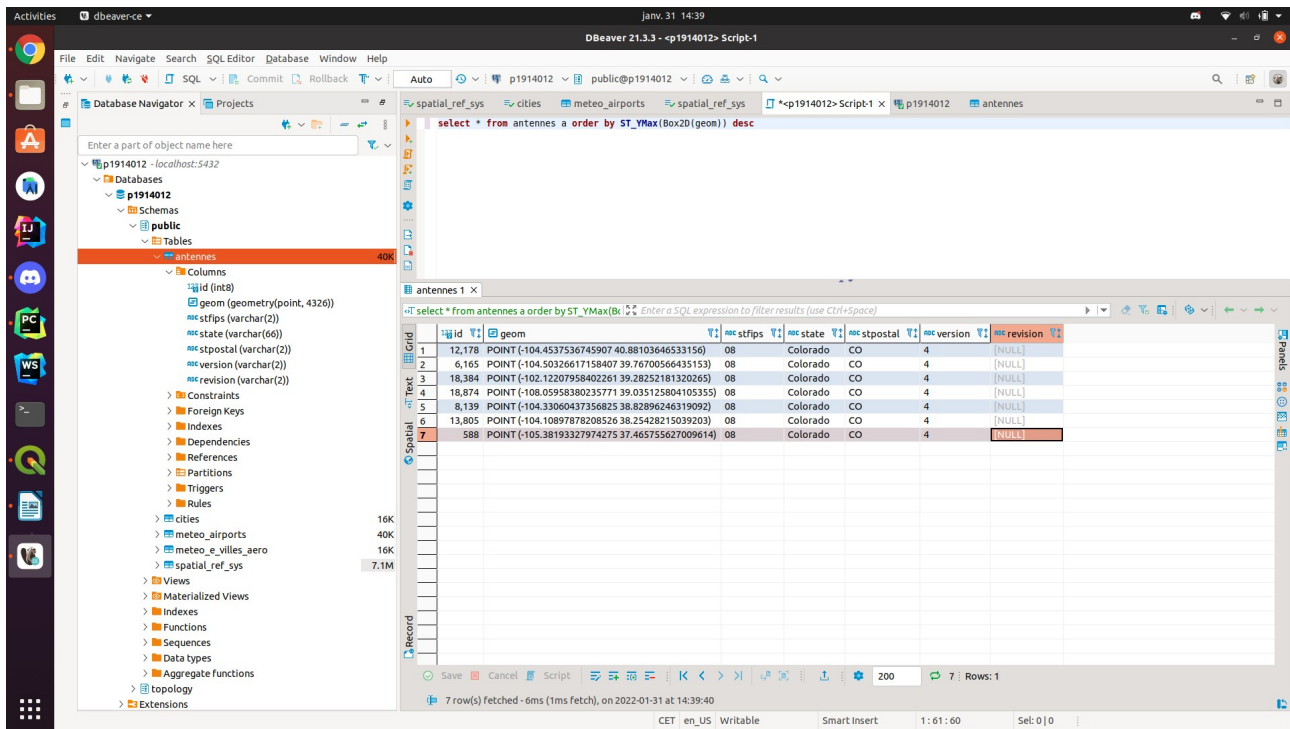


## b) Export QGis vers PostGis

b.1) Copie d'écran IMAGE "b\_export\_qgis\_commande" : 1  
copie d'écran de QGis montrant la commande QGis qui a  
permis d'exporter la table



**b.2) Copie d'écran IMAGE "b\_export\_postgis\_table" : 1 copie d'écran de PostGis (par exemple pgAdmin ou DBeaver) de la requête qui affiche tous les tuples de la table "antennes" avec son résultat**



The screenshot shows the DBeaver 21.3.3 interface. The left sidebar displays the database structure for 'p1914012', including tables like 'antennes'. The main window shows a SQL query: `select * from antennes order by ST_YMax(BOX2D(geom)) desc`. The results are displayed in a table with 7 rows and 6 columns: `id`, `geom`, `stflips`, `state`, `stpostal`, and `revision`. The data is sorted by `ST_YMax` in descending order.

id	geom	stflips	state	stpostal	revision
12,178	POINT (-104.4537536745907 40.88103646533156)	08	Colorado	CO	4
6,165	POINT (-104.50326617158407 39.76700566435153)	08	Colorado	CO	4
18,384	POINT (-102.12207958402261 39.28252181320265)	08	Colorado	CO	4
18,874	POINT (-108.05958380235771 39.035125804105355)	08	Colorado	CO	4
8,139	POINT (-104.33060437356825 38.82896246319092)	08	Colorado	CO	4
13,805	POINT (-104.10897878208526 38.25428215039203)	08	Colorado	CO	4
588	POINT (-105.38193327974275 37.465755627009614)	08	Colorado	CO	4

## c) Correction sous PostGis

**c.1) TEXTE SQL "c\_correction.sql" : 1 fichier qui contient :  
la requête SQL qui crée la table "correction" + la requête qui  
affiche tous les tuples de la table "correction"**

```
-- REQUETES SQL ...
```

```
create table correction as (select id, st_translate(geom,-0.5,0)  
from antennes a);
```

```
select * from correction c order by ST_YMax(Box2D(st_translate))  
desc
```

## c.2) Copie d'écran IMAGE

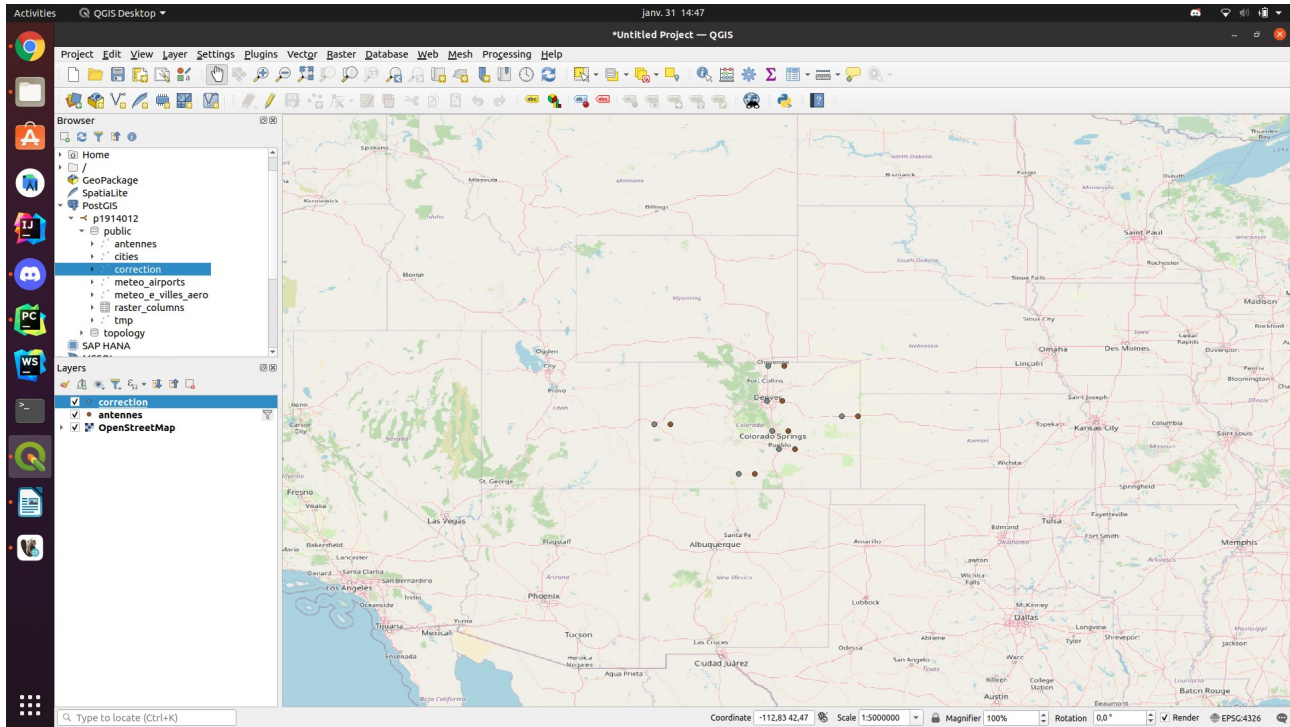
"c\_correction\_postgis\_resultat" : 1 copie d'écran de PostGis de la requête qui affiche tous les tuples de la table "correction" avec son résultat

The screenshot shows the DBeaver 21.3.3 interface. The left sidebar displays the database structure for 'p1914012' on 'localhost:5432', with the 'correction' table selected. The main editor shows a SQL query: `select * from correction c order by ST_YMax(BOX2D(st_translate)) desc`. The results pane displays 7 rows of data, each containing an 'id' and a 'st\_translate' point.

id	st_translate
1	POINT (-104.9537536745907 40.88103646533156)
2	POINT (-105.00326617158407 39.76700566435153)
3	POINT (-102.62207958402261 39.28252181320265)
4	POINT (-108.55958380235771 39.035125804105355)
5	POINT (-104.83060437356825 38.82896246319092)
6	POINT (-104.60897878208526 38.25428215039203)
7	POINT (-105.88193327974275 37.465755627009614)

7 row(s) fetched - 6ms (2ms fetch), on 2022-01-31 at 14:48:29

**c.3) Copie d'écran IMAGE "c\_correction\_qgis\_resultat" : 1 copie d'écran de QGis qui affiche les couches superposées : antennes, antennes corrigées, fond OSM**





## d) Couvertures des antennes en Python

**d.1) TEXTE "d\_recuperation\_calcul.txt" : 1 fichier dans lequel est écrite la formule et le résultat du calcul de 50km sur l'équateur en degrés**

La formule est ...

$$L = \pi * R * a / 180$$

- "L" la distance de l'arc de cercle entre deux points (en km),
- "R" le rayon du cercle (en km),
- "a" l'angle (en degrés).

Avec  $R = 6378.1$  km

$$L * 180 / \pi * R = a$$
$$6378.1 * 180 / (\pi * 50) = 0.449^\circ$$

## **d.2) TEXTE CODE "d\_recuperation.py" : 1 fichier nommé "recuperation.py" qui contient le code Python**

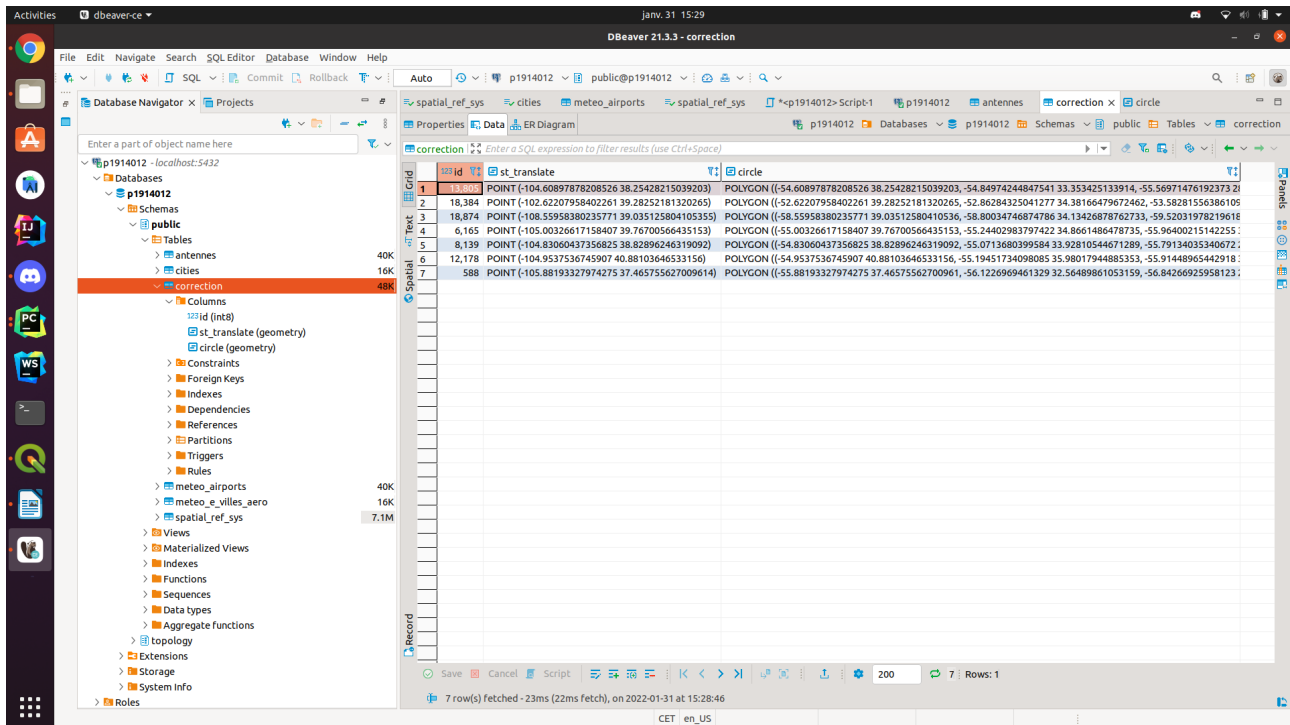
# Code Python ...

```
import pg8000.native
from shapely import wkb
```

```
con = pg8000.native.Connection("p1914012",
password="nBdh3CzPeZyp", database="p1914012")
table = con.run("SELECT * FROM correction")
con.run("ALTER TABLE correction ADD COLUMN IF NOT EXISTS circle
geometry")
for line in table:
line[1] = wkb.loads(bytes.fromhex(line[1]))
circle = line[1].buffer(50.0)
con.run("UPDATE correction SET circle = :circle WHERE id=:id",
circle=circle, id=line[0])
```

**d.3) Copie d'écran IMAGE**  
**"d\_recuperation\_console\_resultat" : 1 copie d'écran de la console où le script liste les infos des antennes**

**"d\_recuperation\_console\_resultat" : 1 copie d'écran de la console où le script liste les infos des antennes**



## e) Serveur WFS en Python

### e.1) TEXTE CODE "e\_wfs.py" : 1 fichier qui contient le code Python

```
# Code Python ...

# Ecrivez votre code Python ici ...
# Partie récupération des données
import geojson
import requests
import pg8000.native
from shapely import wkb
import simplekml
import json

antennes = []
con = pg8000.native.Connection("p1914012",
password="nBdh3CzPeZYp", database="p1914012")
table = con.run("SELECT * FROM correction")
for line in table:
    # line[5] = wkb.loads(bytes.fromhex(line[5]))
    antennes.append([line[0],
[wkb.loads(bytes.fromhex(line[1])).y,wkb.loads(bytes.fromhex(line[
1])).x], line[2]])
print(antennes)

# Partie serveur Flask

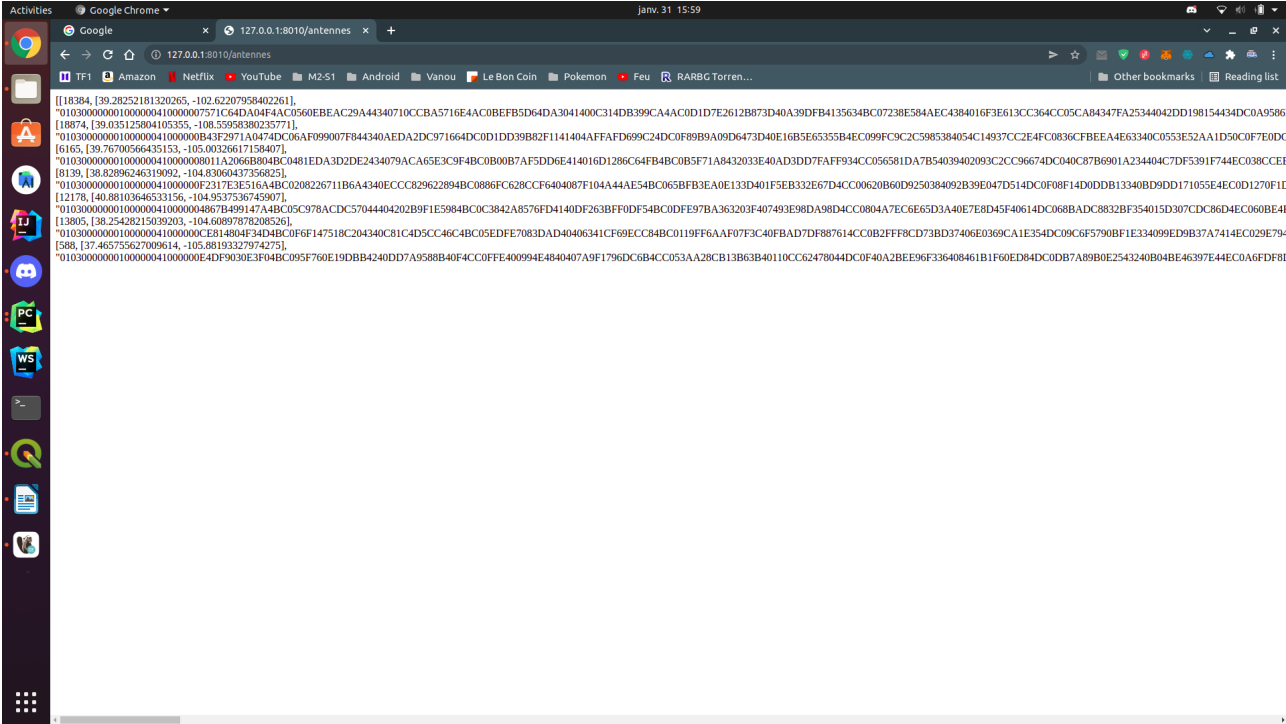
from flask import Flask
from flask_cors import CORS

app = Flask(name)
CORS(app)

# Sur terminal
# export FLASK_APP=main_2_2
# flask run --port 8010
@app.route("/antennes")
def get_antennes():
    return json.dumps(antennes)
```

**e.2) Copie d'écran IMAGE "e\_wfs\_geojson\_proprietes" : 1 copie d'écran du navigateur affichant les antennes en Geojson filtrées -> développez toutes les propriétés et réduisez toutes les géométries pour qu'on puisse voir toutes les antennes retournées, et qu'on voie l'url**

## les coordonnées de quelques points



## f) Carte en ligne en JS

### f.1) TEXTE CODE "f\_carteweb.html" : 1 fichier qui contient le code html et javascript

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Document</title>
  <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
integrity="sha512-
xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAsh0MAS6/
keqq/sMZMZ19scR4PsZChSR7A=="
crossorigin=""/>
  <!-- Make sure you put this AFTER Leaflet's CSS -->
  <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
integrity="sha512-
XQoYMQMTK8LvdXxyG3nZ448h0EQiglfqkJs1N0QV44cWnUrBc8Pka0cXy20w0vlaXa
VUearIOBhiXZ5V3ynxwA=="
crossorigin=""></script>
</head>
<body>
  <div id="map" style="height: 100vh"></div>
  <script>
    (async () => {
      var map = L.map('map').fitBounds([[37.0, -105.6],
[41.0, -104.7]]);
      var tiles =
L.tileLayer('https://api.mapbox.com/styles/v1/{id}/tiles/{z}/{x}/{
y}?
access_token=pk.eyJ1IjoibWFwYm94IiwiYSI6ImNpejY4NXVycTA2emYycXBndH
RqcmZ3N3gifQ.rJcFIG214AriISLbB6B5aw', {
        maxZoom: 7,
        attribution: 'Map data &copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors, ' +
        'Imagery © <a
href="https://www.mapbox.com/">Mapbox</a>',
        id: 'mapbox/streets-v11',
        tileSize: 512,
        zoomOffset: -1
      }).addTo(map);

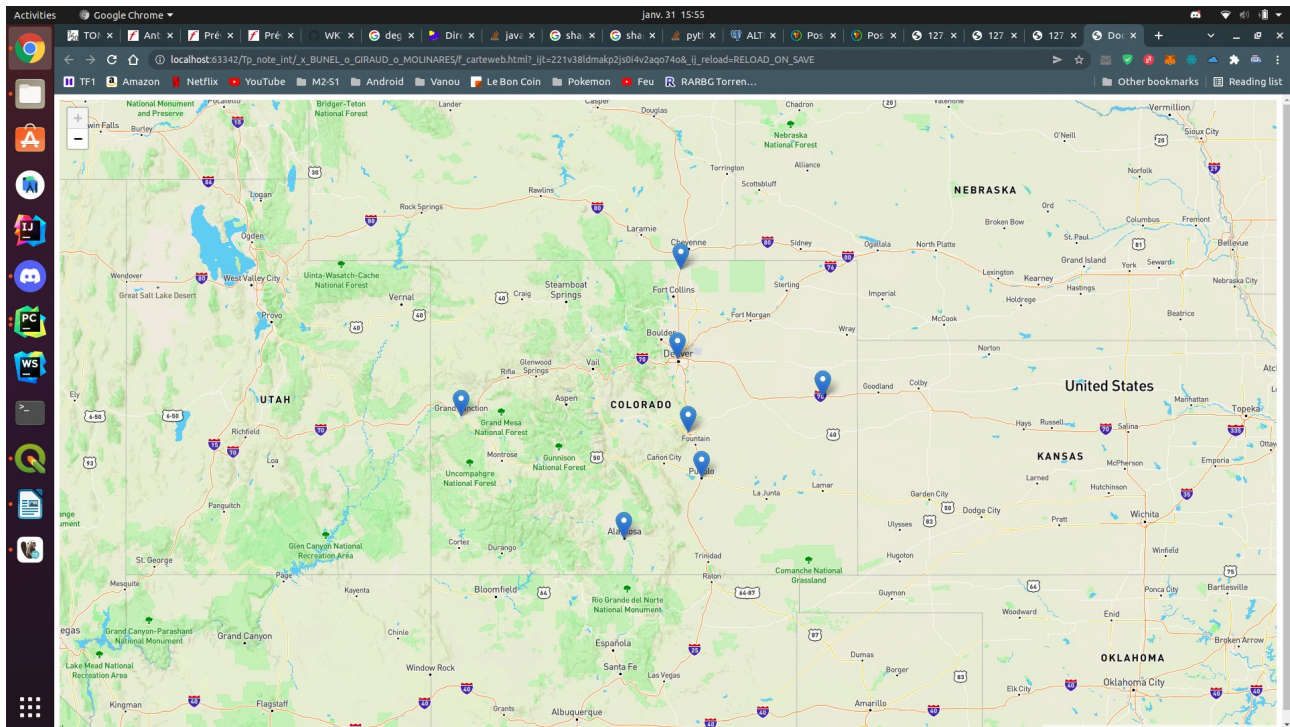
      var response = await
fetch("http://localhost:8010/antennes");
```

```
var contenu = await response.json();

contenu.forEach(c => {
    // add polygon
    L.marker(c[1]).addTo(map);
    // add point
    // L.polygon(c[2], {color: 'red'}).addTo(map);
})
}
})();
</script>
</body>
</html>
```



## f.2) Copie d'écran IMAGE "f\_carteweb" : 1 copie d'écran du navigateur affichant les antennes dans le rectangle englobant sur le fond de carte OpenStreetMap



**f.3) TEXTE "f\_carteweb.txt" : 1 fichier texte qui contient les réponses aux questions**

Réponses : ...