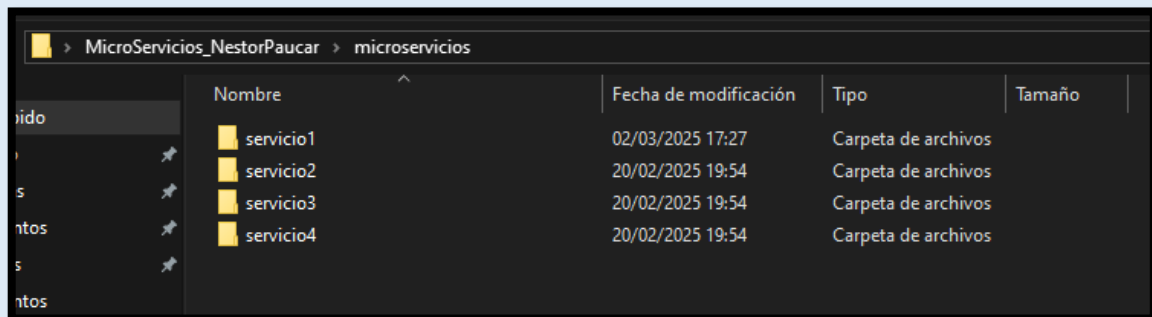


## MICROSERVICIOS

### 1. Comprobar si tenemos instalado Python:

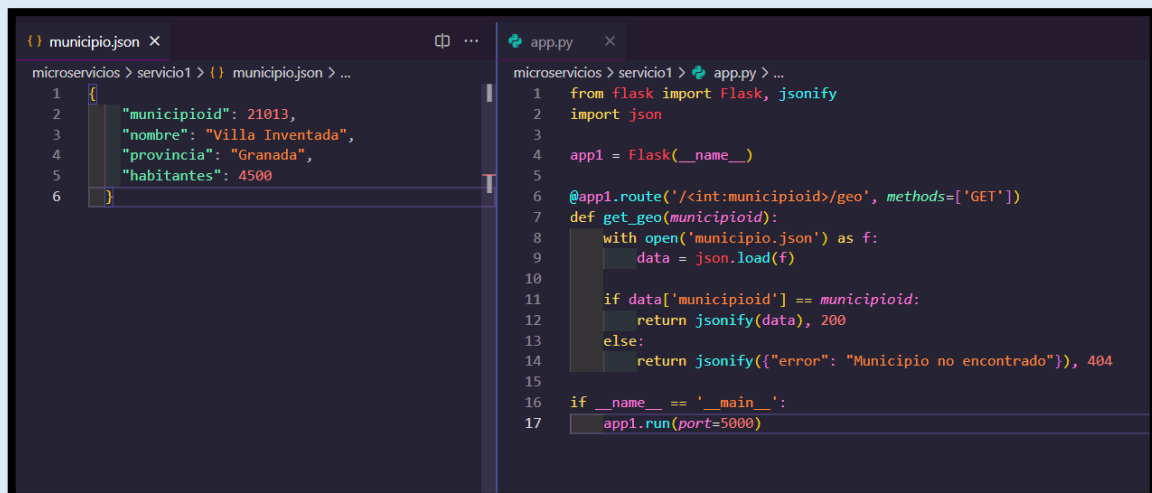
```
C:\Users\nesto>python --version
Python 3.12.0
C:\Users\nesto>
```

### 2. Creación de la estructura del proyecto:



Nombre	Fecha de modificación	Tipo	Tamaño
servicio1	02/03/2025 17:27	Carpeta de archivos	
servicio2	20/02/2025 19:54	Carpeta de archivos	
servicio3	20/02/2025 19:54	Carpeta de archivos	
servicio4	20/02/2025 19:54	Carpeta de archivos	

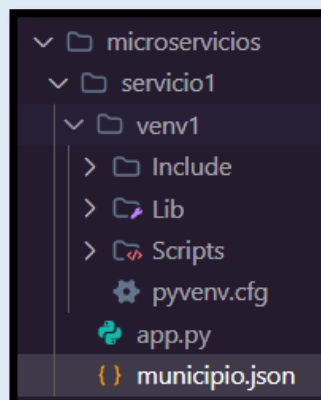
### 3. Creación de servicio1:



```
municipio.json
1 {
2   "municipioid": 21013,
3   "nombre": "Villa Inventada",
4   "provincia": "Granada",
5   "habitantes": 4500
6 }
```

```
app.py
1 from flask import Flask, jsonify
2 import json
3
4 app1 = Flask(__name__)
5
6 @app1.route('/<int:municipioid>/geo', methods=['GET'])
7 def get_geo(municipioid):
8     with open('municipio.json') as f:
9         data = json.load(f)
10
11     if data['municipioid'] == municipioid:
12         return jsonify(data), 200
13     else:
14         return jsonify({"error": "Municipio no encontrado"}), 404
15
16 if __name__ == '__main__':
17     app1.run(port=5000)
```

Creación de un entorno virtual “venv”:



#### 4. Creación de servicio2:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\nesto\Desktop\MicroServicios_NestorPaucar\microservicios\servicio2> python3 -m venv venv2
```

```
microservicios > servicio2 > app.py > ...
1  from flask import Flask, jsonify
2  import json
3
4  app2 = Flask(__name__)
5
6  @app2.route('/<int:municipioid>/meteo', methods=['GET'])
7  def get_meteo(municipioid):
8      try:
9          with open('meteo.json') as f:
10             datos = json.load(f)
11
12             if str(municipioid) in datos:
13                 return jsonify(datos[str(municipioid)]), 200
14             else:
15                 return jsonify({"error": "Municipio no encontrado"}), 404
16
17     except Exception as e:
18         return jsonify({"error": "Error al obtener datos", "detalle": str(e)}), 500
19
20 if __name__ == '__main__':
21     app2.run(port=5001)
22
```

#### 5. Creación de servicio3:

```
PS C:\Users\nesto\Desktop\MicroServicios_NestorPaucar\microservicios\servicio3> python -m venv venv3
```

```
EXPLORER  ...  demo.json X  app.py
microservicios > servicio3 > demo.json > ...
1  {
2      "21013": {
3          "alcalde": "Rubén Rodríguez Camacho",
4          "partidopolitico": "PSOE",
5          "numhabitantes": 14263
6      }
7  }
8

microservicios > servicio3 > app.py > ...
1  from flask import Flask, jsonify
2  import json
3
4  app3 = Flask(__name__)
5
6  @app3.route('/<int:municipioid>/demo', methods=['GET'])
7  def get_demo(municipioid):
8      try:
9          with open('demo.json', encoding='utf-8') as f:
10             datos = json.load(f)
11
12             if str(municipioid) in datos:
13                 return jsonify(datos[str(municipioid)]), 200
14             else:
15                 return jsonify({"error": "Municipio no encontrado"}), 404
16
17     except Exception as e:
18         return jsonify({"error": "Error al obtener datos", "detalle": str(e)}), 500
19
20 if __name__ == '__main__':
21     app3.run(port=5002)
```

## 6. Creación de servicio 4:

```

6
7 from flask import Flask, jsonify
8 import requests
9
10 app4 = Flask(__name__)
11
12 @app4.route('/<int:municipioid>/<parametro1>/<parametro2>', methods=['GET'])
13 def get_combined(municipioid, parametro1, parametro2):
14     base_urls = {
15         "geo": f"http://localhost:5000/{municipioid}/geo",
16         "meteo": f"http://localhost:5001/{municipioid}/meteo",
17         "demo": f"http://localhost:5002/{municipioid}/demo"
18     }
19
20     datos = {}
21
22     for param in [parametro1, parametro2]:
23         if param not in base_urls:
24             return jsonify({"error": f"Parámetro desconocido: {param}"}), 400
25
26         try:
27             res = requests.get(base_urls[param])
28             if res.status_code == 200:
29                 datos[param] = res.json()
30             else:
31                 datos[param] = {"error": f"Servicio {param} devolvió {res.status_code}"}
32         except Exception as e:
33             datos[param] = {"error": f"No se pudo conectar al servicio {param}", "detalle": str(e)}
34
35     return jsonify(datos), 200
36
37 if __name__ == '__main__':
38     app4.run(port=5003)
39
40
41 if __name__ == '__main__':
42     app4.run(port=5003)

```

```

PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio4> python -m venv venv4
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio4> pip install Flask

```

## 7. Creación de los requirements de cada servicio:

Un requirements en cada servicio:

```

PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio1> pip > requirements.txt
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio1> cd..
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios> cd servicio2
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio2> pip > requirements.txt

```

```

PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio2> cd..
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios> cd servicio3
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio3> pip > requirements.txt
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio3> cd..
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios> cd servicio4
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio4> pip > requirements.txt
PS C:\Users\nesto\Desktop\MicroServicios_NestorPauca\microservicios\servicio4>

```

## 8. Uso de contenedores:

### Servicio1:

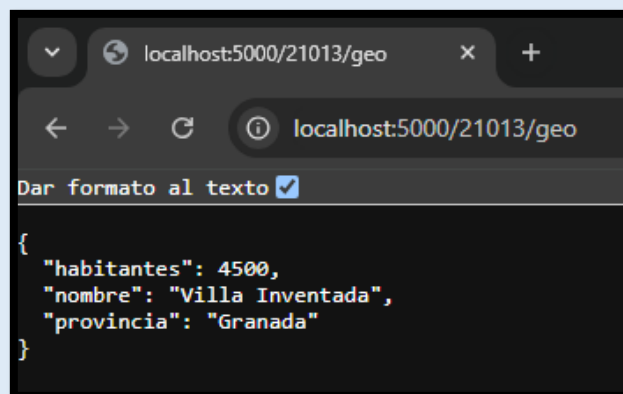
```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\nesto\Desktop\MicroServicios_NestorPaucar\microservicios\servicio1> docker build -t servicio1 .
>>
[+] Building 14.5s (9/9) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 477B
=> [internal] load metadata for docker.io/library/python:3.12-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/4] FROM docker.io/library/python:3.12-alpine@sha256:c08bfdbffc9184cdfd225497bac12b2c0dac1d24bbe13287cfb7d99f1116cf43
=> => resolve docker.io/library/python:3.12-alpine@sha256:c08bfdbffc9184cdfd225497bac12b2c0dac1d24bbe13287cfb7d99f1116cf43
=> => sha256:9a82b5a7bd4e070a570f3429798fcb6fda1d3b5999244e24358d5e82a4919bad 5.27kB / 5.27kB
=> => sha256:f18232174bc91741fdf3da96d85011092101a032a93a388b79e99e69c2d5c870 3.64MB / 3.64MB
=> => sha256:022d2bfc10efcf304ab2472f132398163ac52ca7de9a9991193c518dec3e3896 460.17kB / 460.17kB
=> => sha256:cb5f2229c4048898438f135d4edd5a66f8c0644f3a243aa4043760a3e5c515a8 13.67MB / 13.67MB
=> => sha256:c08bfdbffc9184cdfd225497bac12b2c0dac1d24bbe13287cfb7d99f1116cf43 9.03kB / 9.03kB
=> => sha256:4cad1c099a56dafcfee656a1bdd88c97a1372db02e14cffa9b1869515956deb 1.74kB / 1.74kB
=> => extracting sha256:f18232174bc91741fdf3da96d85011092101a032a93a388b79e99e69c2d5c870
=> => sha256:65bf70710079b13e40e5bff350e3047d007801d1ade028c3904a5f0a25662a7e 247B / 247B
=> => extracting sha256:022d2bfc10efcf304ab2472f132398163ac52ca7de9a9991193c518dec3e3896
=> => extracting sha256:cb5f2229c4048898438f135d4edd5a66f8c0644f3a243aa4043760a3e5c515a8
=> => extracting sha256:65bf70710079b13e40e5bff350e3047d007801d1ade028c3904a5f0a25662a7e
=> [internal] load build context
=> => transferring context: 19.65MB
=> [2/4] WORKDIR /app
=> [3/4] COPY . .
=> [4/4] RUN pip install --no-cache-dir -r requirements.txt
=> exporting to image
=> => exporting layers
=> => writing image sha256:c83c0505d86f3a7d5fa26c8361d779802832a202377522c73bfaf94b367bbe58
=> => naming to docker.io/library/servicio1

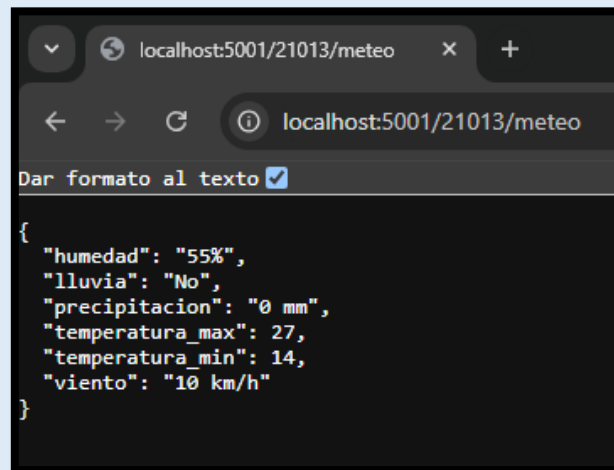
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/267erihspamcaio351jfwSot1

What's next:
  View a summary of image vulnerabilities and recommendations → docker scout quickview
PS C:\Users\nesto\Desktop\MicroServicios_NestorPaucar\microservicios\servicio1>
```

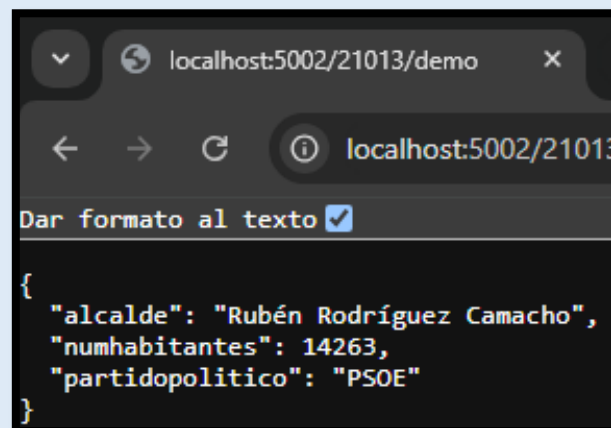
Usando la siguiente URL <http://localhost:5000/21013/geo> para el *servicio1* nos da este resultado.



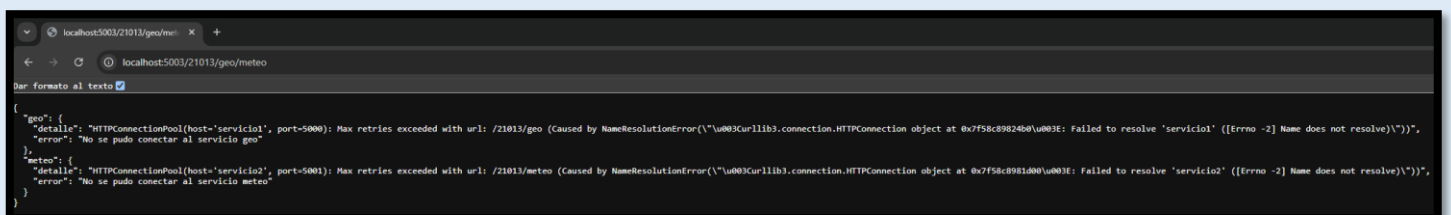
Usando la siguiente URL <http://localhost:5001/21013/meteo> para el *servicio 2* nos dará este resultado:



Usando la siguiente URL <http://localhost:5002/21013/demo> para el *servicio 3* nos dará este resultado:

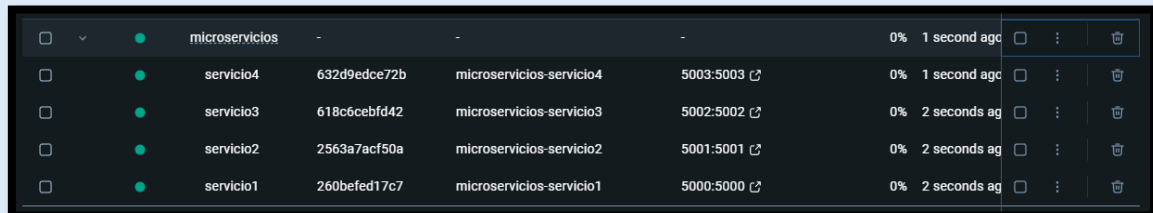


Usando la siguiente URL <http://localhost:5003/21013/demo> para el *servicio 4* nos dará este resultado:



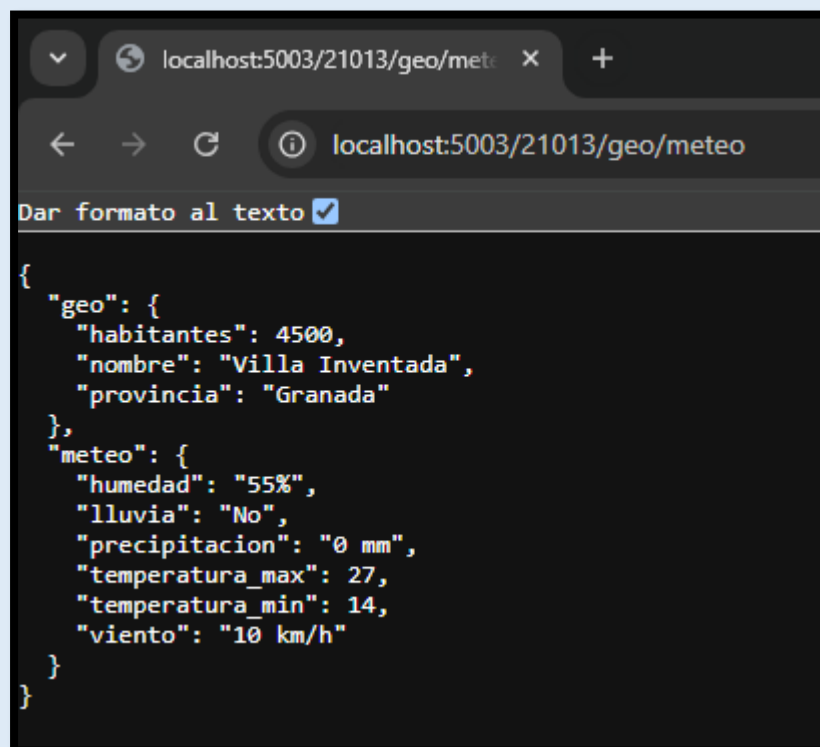
En la anterior imagen no nos devuelve lo que queremos ya que en el código me da problemas cuando uso localhost en microservicios en Docker.

Así que he optado por usar Docker compose y agrupar los 4 servicios en uno y así poder ejecutarlos todos y que me devuelva lo que quiero:



<input type="checkbox"/>	microservicios	-	-	-	0%	1 second ago	<input type="checkbox"/>	:	🗑
<input type="checkbox"/>	servicio4	632d9edce72b	microservicios-servicio4	5003:5003 ↗	0%	1 second ago	<input type="checkbox"/>	:	🗑
<input type="checkbox"/>	servicio3	618c6cebfd42	microservicios-servicio3	5002:5002 ↗	0%	2 seconds ago	<input type="checkbox"/>	:	🗑
<input type="checkbox"/>	servicio2	2563a7acf50a	microservicios-servicio2	5001:5001 ↗	0%	2 seconds ago	<input type="checkbox"/>	:	🗑
<input type="checkbox"/>	servicio1	260befed17c7	microservicios-servicio1	5000:5000 ↗	0%	2 seconds ago	<input type="checkbox"/>	:	🗑

Usando la misma URL y ahora Docker compose, para llamar a los dos: geo y meteo me da este resultado:



```
{
  "geo": {
    "habitantes": 4500,
    "nombre": "Villa Inventada",
    "provincia": "Granada"
  },
  "meteo": {
    "humedad": "55%",
    "lluvia": "No",
    "precipitacion": "0 mm",
    "temperatura_max": 27,
    "temperatura_min": 14,
    "viento": "10 km/h"
  }
}
```