ROSE-API: OGC API for Environmental Monitoring Data

Juan Pablo Duque Ordoñez

Angelly Pugliese, Maria Brovelli

Politecnico di Milano

FOSS4G Europe 2024 – Tartu, Estonia

July 3rd 2024

Contents

1. Context
2. ROSE-API features
3. Case Study
4. Next Steps

Context

- OGC APIs are modern standards for geospatial information.
- Still gaining widespread adoption.
- Enable **FAIR** access to information (**F**indable, **A**ccessible, **I**nteroperable, **R**eusable).

Context

- Environmental monitoring stations and observations are considered high-value datasets by **INSPIRE**.

- Many organizations expose environmental data without following any standards, which makes their datasets not interoperable.

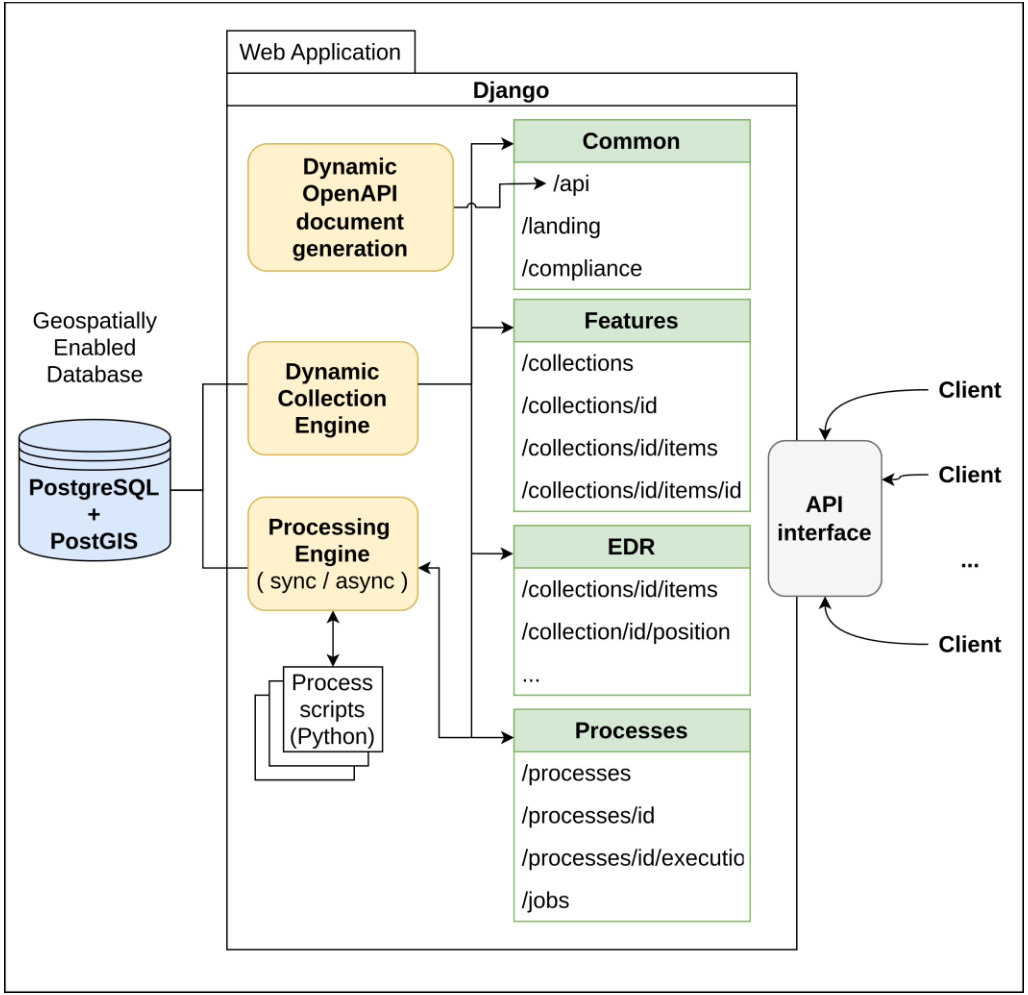
Context

- **Objective:**

Implement a OGC API compliant service for serving environmental monitoring data.

- **Use Case:**

Our implementation was validated using open air quality data from Italy's Lombardy region.



ROSE-API features

Implements the OGC APIs:

- Common
- Features
- Environmental Data Retrieval (EDR)
- Processes

ROSE-API features

- **Dynamic collections:** Users can create collections (geospatial or not) on-the-fly using JSON and the admin interface.

- **Automatic OpenAPI document:** The OpenAPI document is automatically generated from the current collections and processes.

ROSE-API features

- **Processing engine:** ROSE-API provides processing capabilities. Developers can create python scripts to enable additional functionalities to the data stored by ROSE-API.
  Scripts are listed as processes according to the OGC API – Processes standard.

ROSE-API features

- **Sync/Async processing:** ROSE-API is able to perform synchronous and asynchronous processes.

- Synchronous requests are ideal for small processes. Asynchronous requests do not block the API while processing and are useful for big processes. Uses Celery + RabbitMQ.

ROSE-API features

- **Paginated Results:** Retrieve feature data in pages to speed-up computation and manage large datasets.

- **Dynamic configuration:** Some configuration options can be set from a graphical user interface.

ROSE-API features

- **OGC API – Features endpoints:** Query collections and filter by attribute, date and time, and bounding box as specified in OGC API – Features standard.

- **OGC API – EDR endpoints:** Enable additional metadata and spatial queries for retrieveing data.

NB: ROSE-API is compliant with both standards.

ROSE-API features

- **Interconnected collections:** It is possible to relate collections by an attribute. This allows to share information (such as location) between collections.

  Example: One collection contains sensor information (location, name, etc), while other contains the sensor measurements. In this way the measurements collection is lighter as the location and sensor information is stored in a different collection

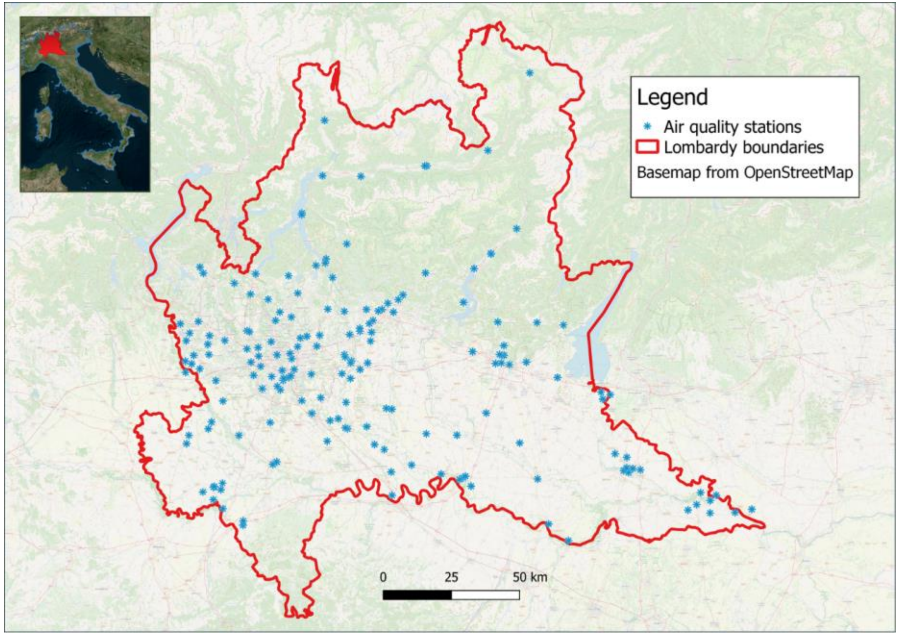
Case Study

- ARPA Lombardia data is widely used within the Polimi GeoLab.
- **Replicated** the ARPA Lombardia environmental monitoring air quality datasets within an instance of ROSE-API.

Case Study

- **Air Quality data**:

- Hourly observations from 2010 to 2024.
- 982 individual sensors.
- 55M+ observations.
- 7 pollutants.



Map with the Lombardy region Air Quality station's locations.

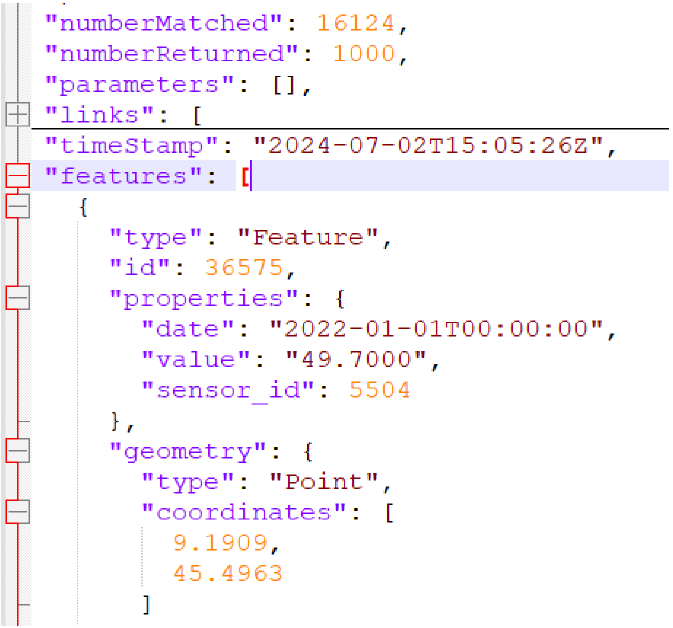
Case Study

- We tested 4 aspects of ROSE-API:
  - Querying capabilities.
  - Response Metadata.
  - Processing Capabilities.
  - Response times.

Case Study

Querying capabilities:

- Allows direct spatial queries over observations using OGC API – EDR endpoints, which is not possible using ARPA Lombardia's API.



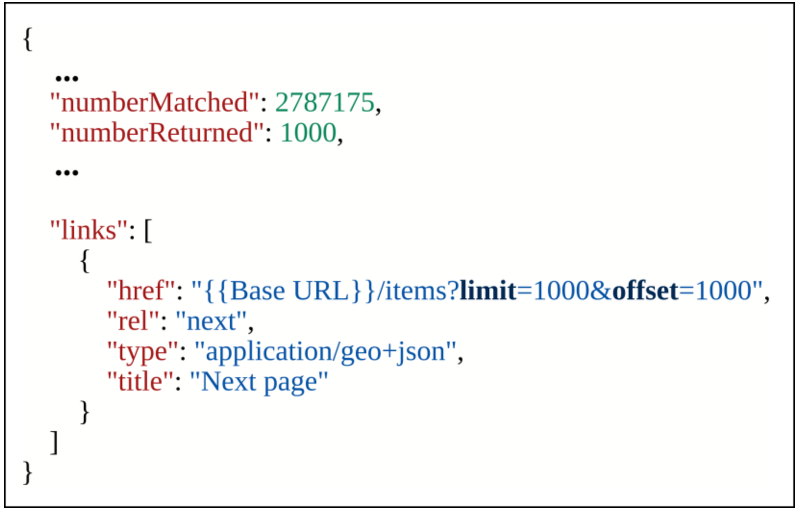
```
"numberMatched": 16124,
"numberReturned": 1000,
"parameters": [],
"links": [
"timeStamp": "2024-07-02T15:05:26Z",
"features": [
  {
    "type": "Feature",
    "id": 36575,
    "properties": {
      "date": "2022-01-01T00:00:00",
      "value": "49.7000",
      "sensor_id": 5504
    },
    "geometry": {
      "type": "Point",
      "coordinates": [
        9.1909,
        45.4963
      ]
```

Example of spatial query in ROSE-API. Query features within the polygon: POLYGON((9.09874 45.477948, 9.238129 45.536175, 9.284821 45.440863, 9.09874 45.477948))

Case Study

Response Metadata:

- A lot of metadata is available using OGC APIs.
  Including multiple links for improved machine and human navigation, pagination information, bounding boxes, etc.



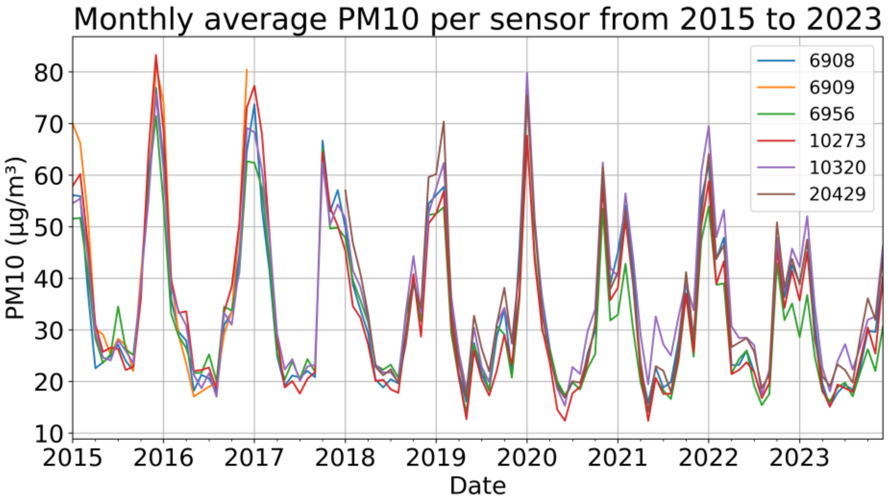
```
{
    ...
    "numberMatched": 2787175,
    "numberReturned": 1000,
    ...

    "links": [
        {
            "href": "{{Base URL}}/items?limit=1000&offset=1000",
            "rel": "next",
            "type": "application/geo+json",
            "title": "Next page"
        }
    ]
}
```

Response Metadata example

Case Study

Processing capabilities:

- Using OGC API – Processes we created a script for performing temporal aggregation (daily and monthly) to the observations.

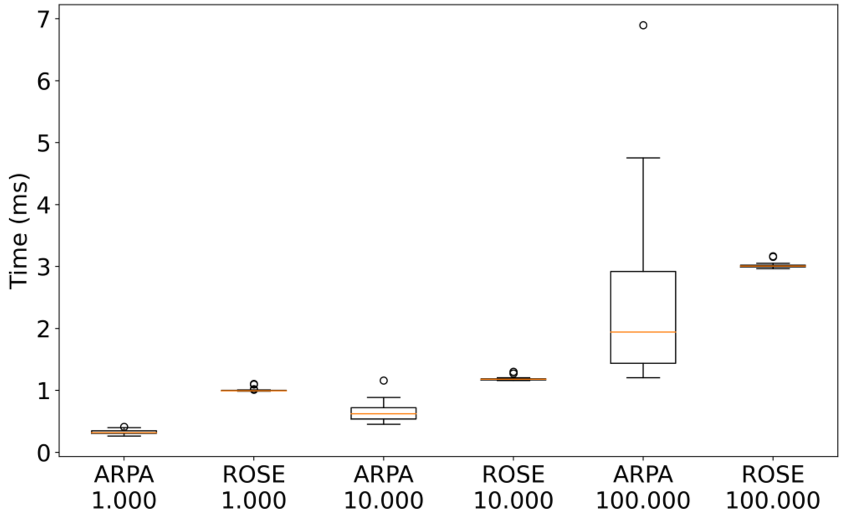


Aggregated Monthly average PM10 concentrations.

Case Study

Response times:

- With respect to the current ARPA Lombardia API, our implementation is slower. However, performance can be improved.



Box plot of response times between ARPA Lombardia API and ROSE-API

Next Steps

- Improve the graphical user interface (GUI).
- Improvement of collection creation.
- Improve data retrieval performance.
- Testing.
- Documentation and wiki pages.

How to contribute

- The Core of ROSE-API is working but still under development. **Contributions are welcome!**
- In GitHub: https://github.com/Diuke/rose-api →
- Tell us your use case.

Check what are we doing at
Polimi GeoLab!

Thank you!

Please let me know if you have any comments
or want to contribute

juanpablo.duque@polimi.it