Koninklijk Nederlands
Meteorologisch Instituut
Ministerie van Infrastructuur en Waterstaat

RODEO WP5 EDR API Workshop

Helsinki

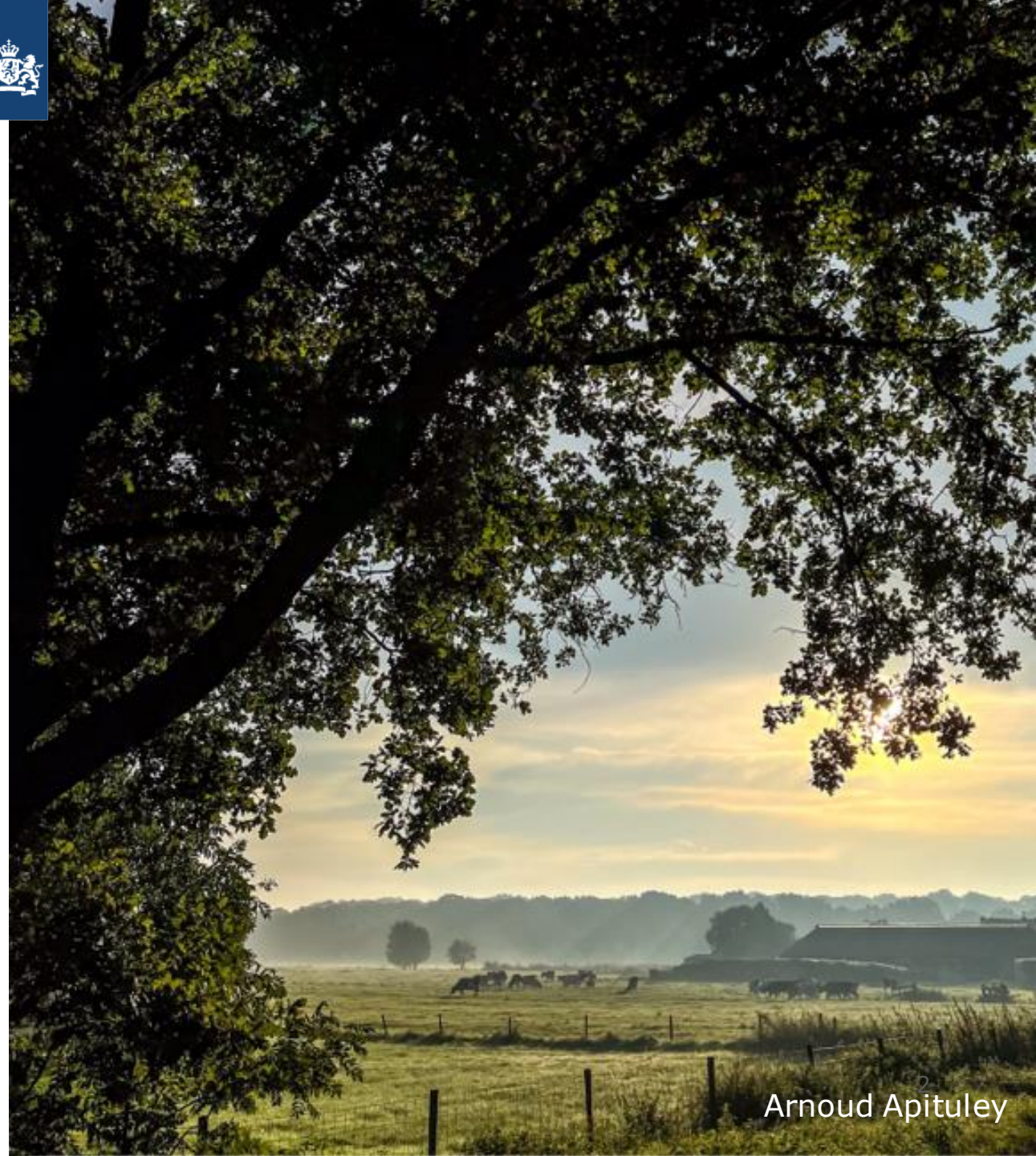
13-06-2024

Lukas Phaf

Paul van Schayck

Why OGC EDR?

- Environmental Data Retrieval
- OGC standard
  - Open Geospatial Consortium
  - Since 2016
- Discoverable
- Filtering in space and time
- Multiple datasets
  - Collections and instances
- Also used by RODEO ESOH



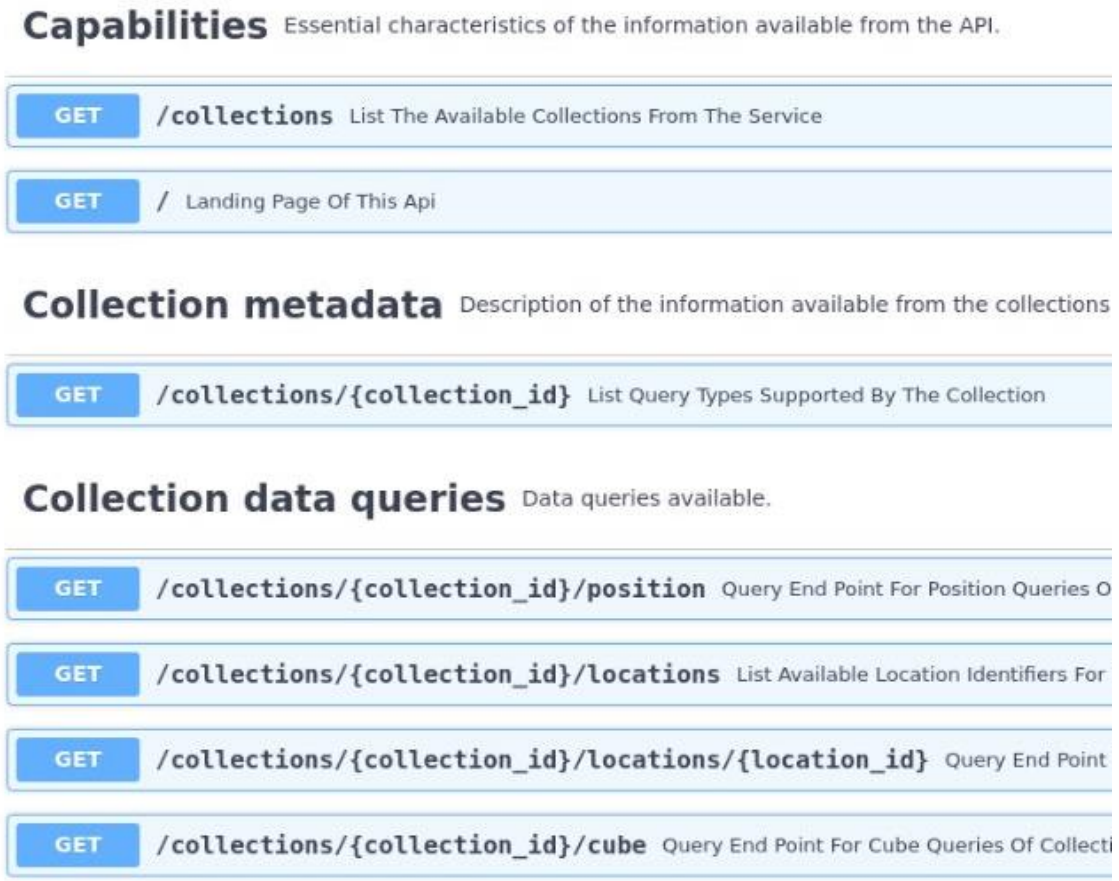
CoverageJSON

- Recommended EDR output
- A format for publishing geotemporal data (on the web)
  - Time series
  - Gridded
- Format: JSON
  - Easier than NetCDF?
  - Better than CSV
- More info:
  - https://covjson.org/


```
{
  "type": "Coverage",
  "domain": {
    "type": "Domain",
    "domainType": "PointSeries",
    "axes": {
      "x": {
        "values": [
          3.275
        ]
      },
      "y": {
        "values": [
          51.9978
        ]
      },
      "t": {
        "values": [
          "2023-01-22T11:10:00Z"
        ]
      }
    },
    "referencing": [ … 2 … ]
  },
  "parameters": {
    "t_dryb_10": { … 4 … }
  },
  "ranges": {
    "t_dryb_10": {
      "type": "NdArray",
      "dataType": "float",
      "axisNames": [ … 3 … ],
      "shape": [ … 3 … ],
      "values": [
        4.4
      ]
    }
  },
  "inspiregloss:Identifier": "06321"
}
```

How to use an EDR API?

- Any HTTPS client
  - e.g. from web browsers to Python
  - Insomnia or Postman



Demo of KNMI EDR API Synoptic Observations

- Getting started yourself
  - KNMI Developer Portal
  - KNMI EDR Documentation
  - CovJSON playground

Step 0

Setting up

Code and environment

- Clone the repository

```
git clone https://github.com/EURODEO/wp5-edr-workshop.git
cd wp5-edr-workshop
git checkout step_0
```

- Python3 virtual environment

```
python3 -m venv venv/
source venv/bin/activate
```

- Install dependencies

```
pip3 install pip-tools
pip-sync
```

Check setup

```
python3 data/data.py
```

- NetCDF datastore
  - 10 – min synoptic observations of a single day
- data.py is interface between data and EDR
- Should be replaced with your data backend (after workshop...)

```
python3 main.py
```

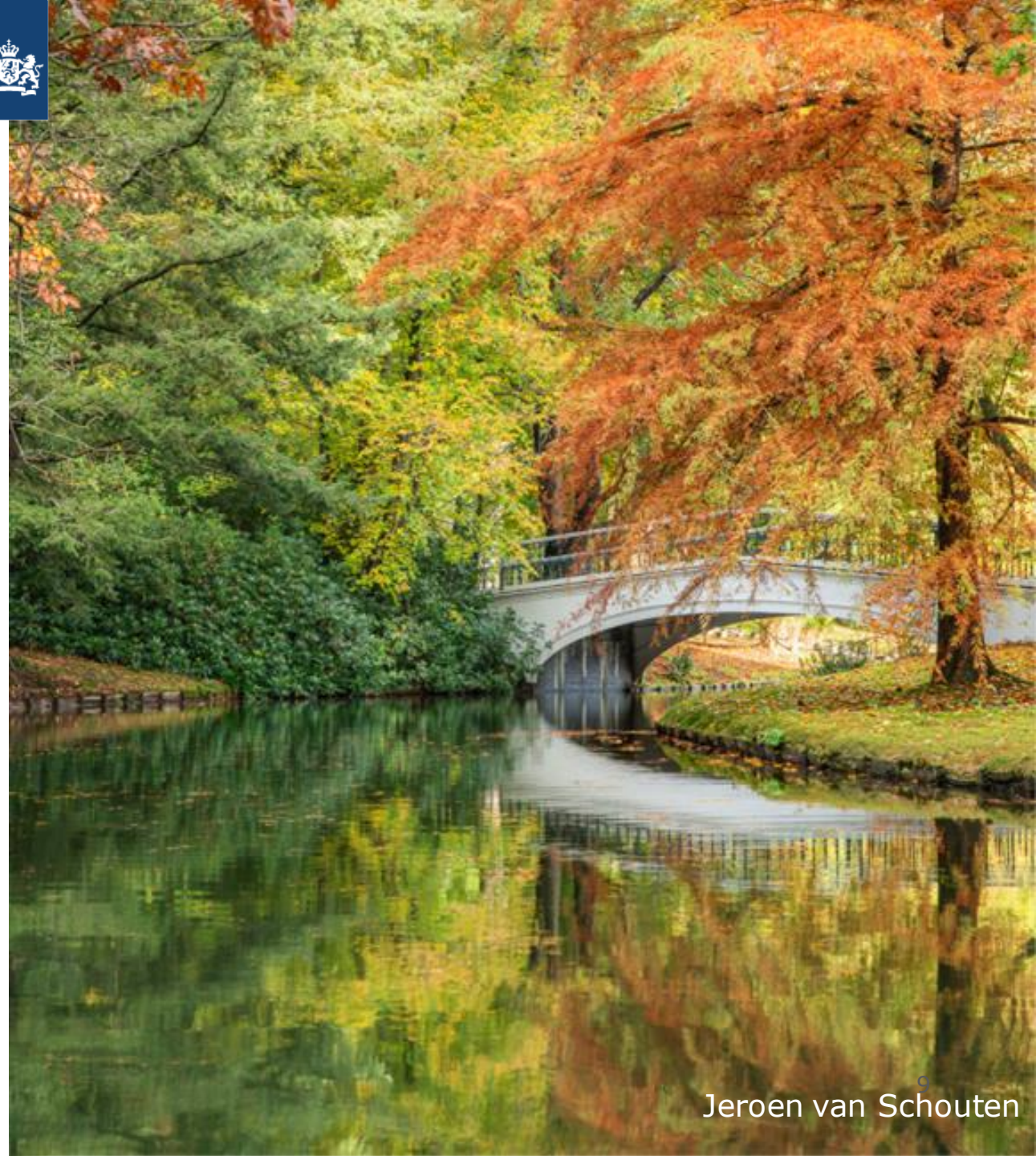
- Starts FastAPI using uvicorn as gateway
- Can be used for step debugging

```
uvicorn main:app --reload
```

- Starts uvicorn with auto reloading
- Test Swagger: http://localhost:8000/docs

How to build an EDR API?

- How ever you want!
  - OGC EDR specification
- Python packages for building EDR APIs
  - Pydantic models for CoverageJSON
  - Pydantic models for EDR
- WP5 Climate EDR:
  - OMSZ EDR API (Hugarian Meteo Service)
- Other examples:
  - EDR-isobaric (MetNorway)
  - RODEO E-SOH EDR API



Jeroen van Schouten

Step 1

Landing Page

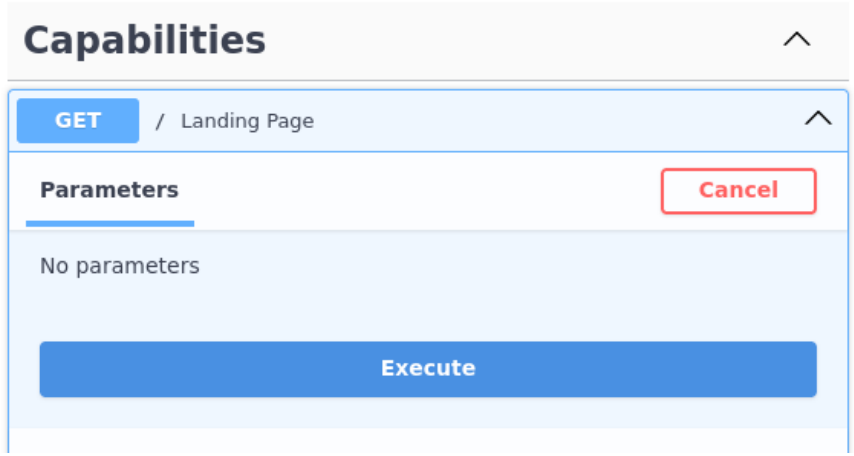
Landing page

- Specification
- Use LandingPageModel from edr_pydantic in main.py

```python
from edr_pydantic.capabilities import LandingPageModel


@app.get(
    "/",
    tags=["Capabilities"],
    response_model=LandingPageModel,
    response_model_exclude_none=True,
)
async def landing_page(request: Request) -> LandingPageModel:
    pass
```

- Test result http://localhost
- Or via Swagger



Result


```
{
    "title": "EDR tutorial",
    "description": "A simple example EDR implementation",
    "links": [
        {
            "href": "http://localhost:8000/",
            "rel": "self",
            "title": "Landing Page in JSON"
        },
        {
            "href": "http://localhost:8000/docs",
            "rel": "service-desc",
            "title": "API description in HTML"
        },
        {
            "href": "http://localhost:8000/openapi.json",
            "rel": "service-desc",
            "title": "API description in JSON"
        },
```


- Problems?

```
git checkout step_1
```

Step 2

Retrieve data for single location

CoverageJSON

- Specification
- Domain (axes)
  - Standard domains
  - We will use PointSeries
- Parameters
- Ranges


```
{
  "type": "Coverage",
  "domain": {
    "type": "Domain",
    "domainType": "PointSeries",
    "axes": {
      "x": {
        "values": [
          5.5081
        ]
      },
      "y": {
        "values": [
          52.4483
        ]
      },
      "t": {
        "values": [
          "2023-10-20T09:10:00Z"
        ]
      }
    },
    "referencing": [ ↔ 2 ↦ ]
  },
  "parameters": { ↔ 1 ↦ },
  "ranges": {
    "t_dryb_10": {
      "type": "NdArray",
      "dataType": "float",
      "axisNames": [
        "t",
        "y",
        "x"
      ],
      "shape": [
        1,
        1,
        1
      ],
      "values": [
        8.4
      ]
    }
  },
  "inspiregloss:Identifier": "06269"
}
```

Endpoint: /collections/observations/locations/{id}

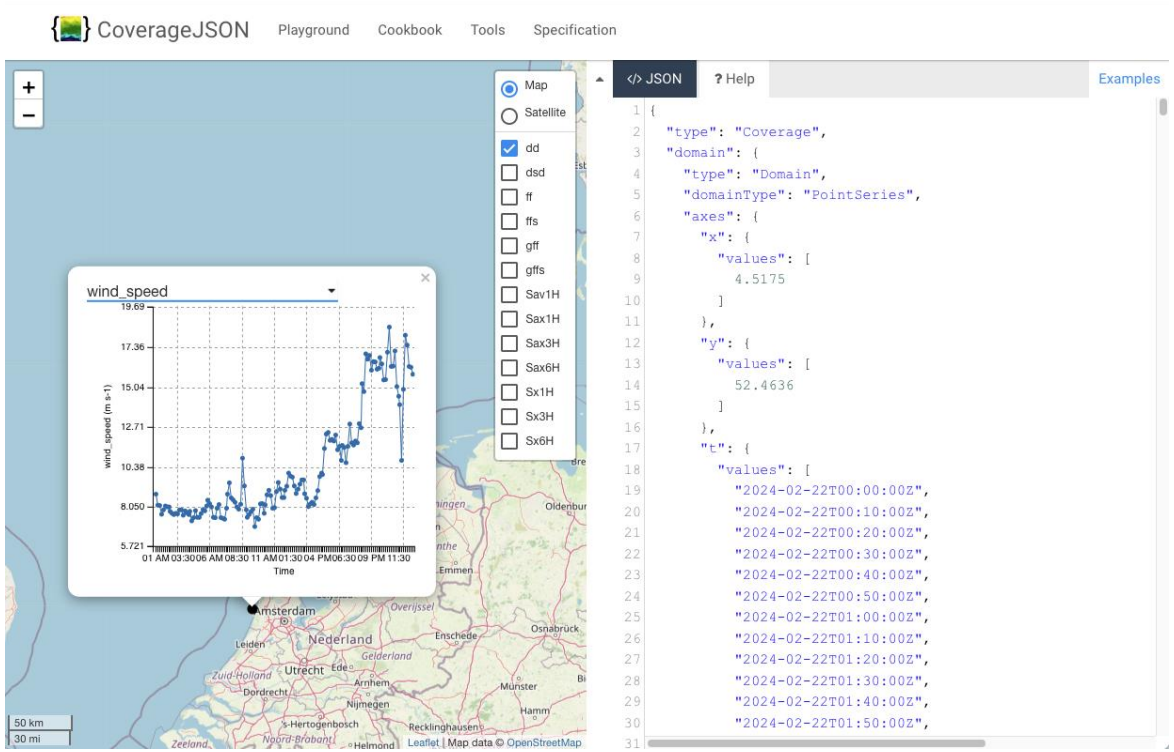
- In `api/observations.py`

```
@router.get(
    "/locations/{location_id}",
    tags=["Collection data queries"],
    response_model=Coverage,
    response_model_exclude_none=True,
    response_class=CoverageJsonResponse,
)
async def get_data_location_id(
    location_id: Annotated[str, Path(example="06260")],
    parameter_name: Annotated[
        str | None,
        Query(alias="parameter-name", description="Comma seperated list of parameter names.", example="ff, dd"),
    ] = None,
    datetime: Annotated[str | None, Query(example="2024-02-22T01:00:00Z/2024-02-22T02:00:00Z")] = None,
) -> Coverage:
    pass
```

- Use
  - data.get_stations()
  - data.get_variables()
  - data.get_data()
- Remarks:
  - No (or minimal) error handling yet
  - No filtering yet

Result

- Use CovJSON playground



- Problems?

```
git checkout step_2
```

Step 3

Filtering (time and parameters)

Filtering (parameters and time)

- Extend `get_data_location_id()`
- Filter `parameter-name`
  - Comma separated list: `dd, ff`
- Filter time
  - start/end (closed interval)
  - ISO8609 string (with Z)

- Remarks:
  - Error handling: What about parameters that don't exist
  - What about parameters that don't exist for the requested station?
- Error responses:
  - 404 for non-existent path
  - 400 for mistake in query parameters
  - 400 (?) for no data (e.g. outside time interval)

Result

- Problems?

```
git checkout step_3
```

Step 4

Collection metadata

Collection metadata

- EDR specification
  - Examples in spec
- Parameters in EDR vs Parameters in CovJSON
- Implement /collections/observations
  - Bonus: Implement /collections


```
{
  "id": "observations",
  "links": [
    {
      "href": "http://localhost:8000/collections/observations",
      "rel": "self"
    }
  ],
  "extent": {
    "spatial": {
      "bbox": [
        [ ↔ 4 ↔ ]
      ],
      "crs": "EPSG:4326"
    },
    "temporal": {
      "interval": [
        [
          "2024-02-22T00:00:00Z",
          "2024-02-22T23:50:00Z"
        ]
      ],
      "values": [
        "2024-02-22T00:00:00Z/2024-02-22T23:50:00Z"
      ],
      "trs": "datetime"
    }
  },
  "data_queries": {
    "area": { ↔ 1 ↔ },
    "locations": {
      "link": { ↔ 3 ↔ }
    }
  },
  "crs": [ ↔ 1 ↔ ],
  "output_formats": [ ↔ 1 ↔ ],
  "parameter_names": {
    "D1H": {
      "type": "Parameter",
      "id": "D1H",
      "label": "D1H",
      "description": "Rainfall Duration in last Hour",
      "unit": {
        "label": "min"
      },
      "observedProperty": {
        "id": "https://vocab.nerc.ac.uk/standard_name/rainfall_duration",
        "label": "rainfall_duration"
      }
    },
```

- Problems?

```
git checkout step_4
```

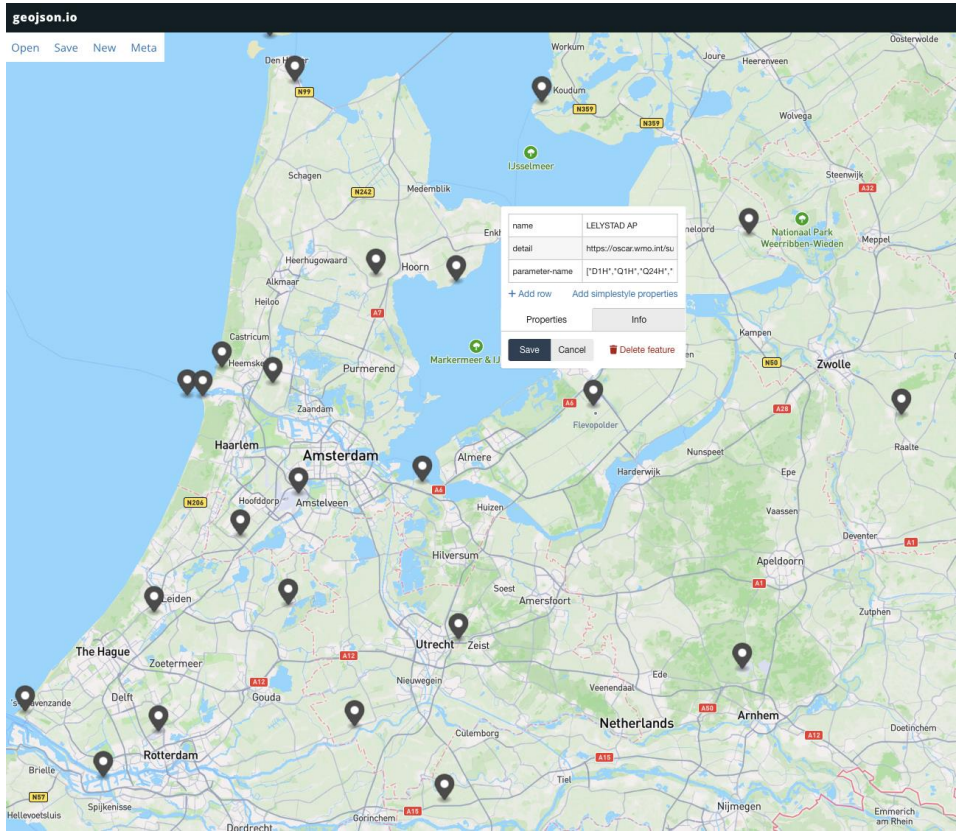
Step 5

List of locations

BONUS

List of locations

- /observations/locations
  - Return GeoJSON
  - geojson.io playground
- Query parameters
  - None in the spec
  - Suggestion: bbox, datetime and parameter-name
- EDRFeatureCollection
  - GeoJSON FeatureCollection with parameters



Step 6

/area query

BONUS

/area query

- CoverageCollection
- Addition of station identifier to Coverage: "inspiregloss:Identifier": station.wsi
- Hint: calculate stations in polygon:

```
from shapely import geometry, wkt
poly = wkt.loads(coords)
stations_in_polygon = [s for s in get_stations()
                       if geometry.Point(s.longitude,
s.latitude).within(poly)]
```