Implementing flexible analysis capabilities in the DAM viewer

... or bringing together DASF and marine-data.de

Dr. Philipp S. Sommer

Helmholtz Coastal Data Center (HCDC)
Helmholtz-Zentrum Hereon

Robin Hess (AWI)
Björn Saß (Hereon)
Angela Schäfer (AWI)

Live demo
https://marine-data.de/preview/vef/usecases/viewer.php#dasf

Data Science Symposium Bremen,
April 5th 2024 DOI 10.5281/zenodo.10929757

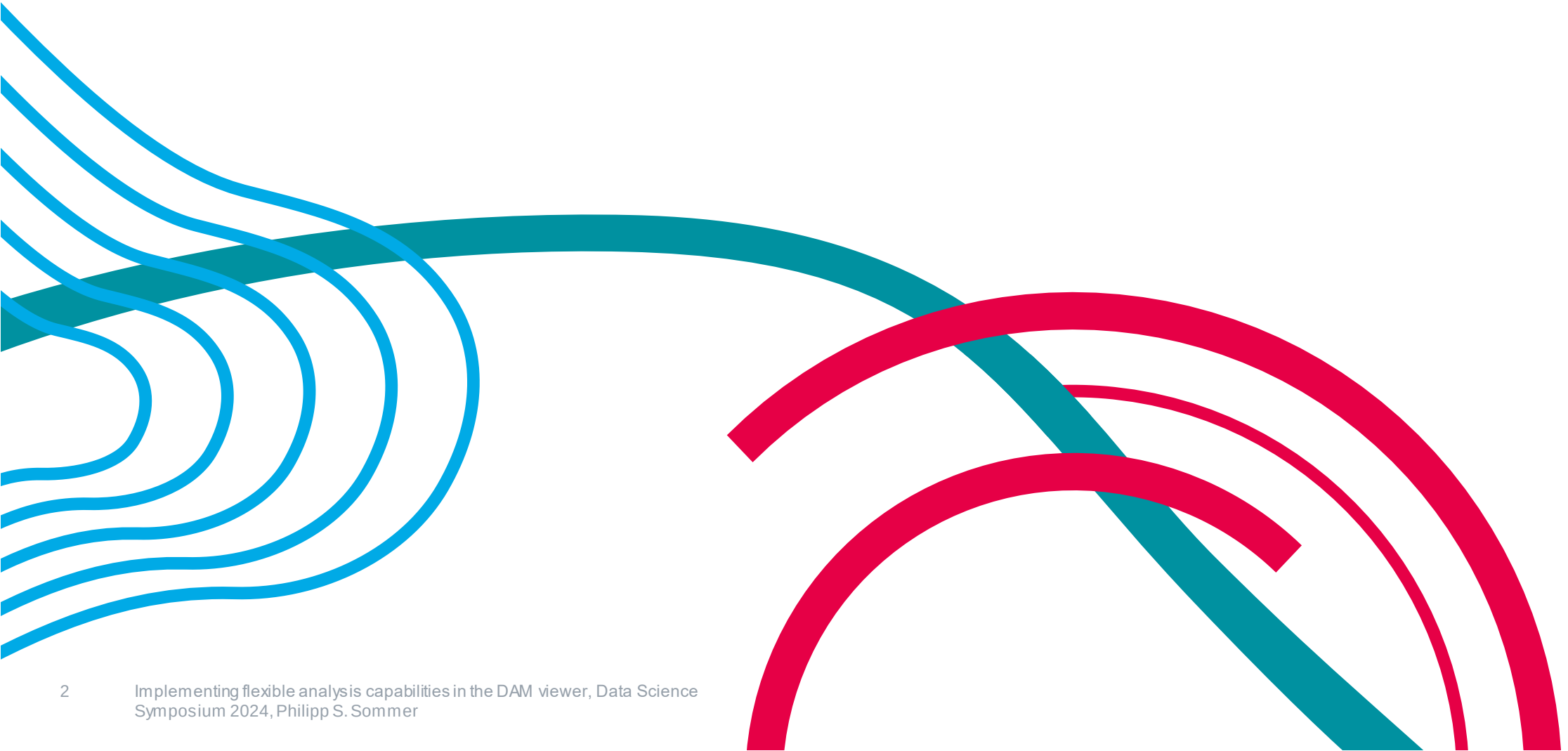
ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG

DEUTSCHE ALLIANZ MEERESFORSCHUNG

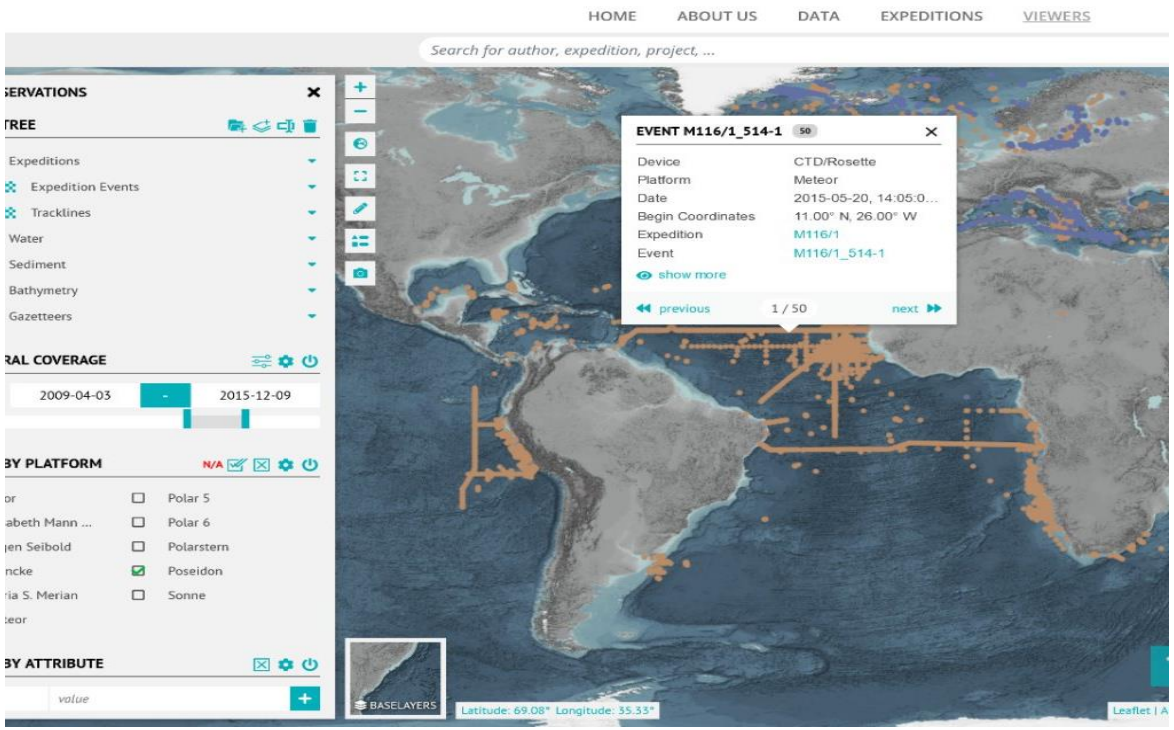
DataHub
Research Field Earth and Environment

Helmholtz-Zentrum hereon

What is the DAM Viewer?



The Visual Exploration Framework (VEF)



The visual exploration framework (VEF) is deployed as part of the Marine Data, and Earth Data Portal.

Cross-institutional viewer for finding and visualizing earth and environment research data. The portal aims to enable scientists to work effectively by giving access to re-usable data.

- **Centralized entry point**
- **Decentralized (meta)data**
- **Free-text search engine**
- **Thematic map viewers**

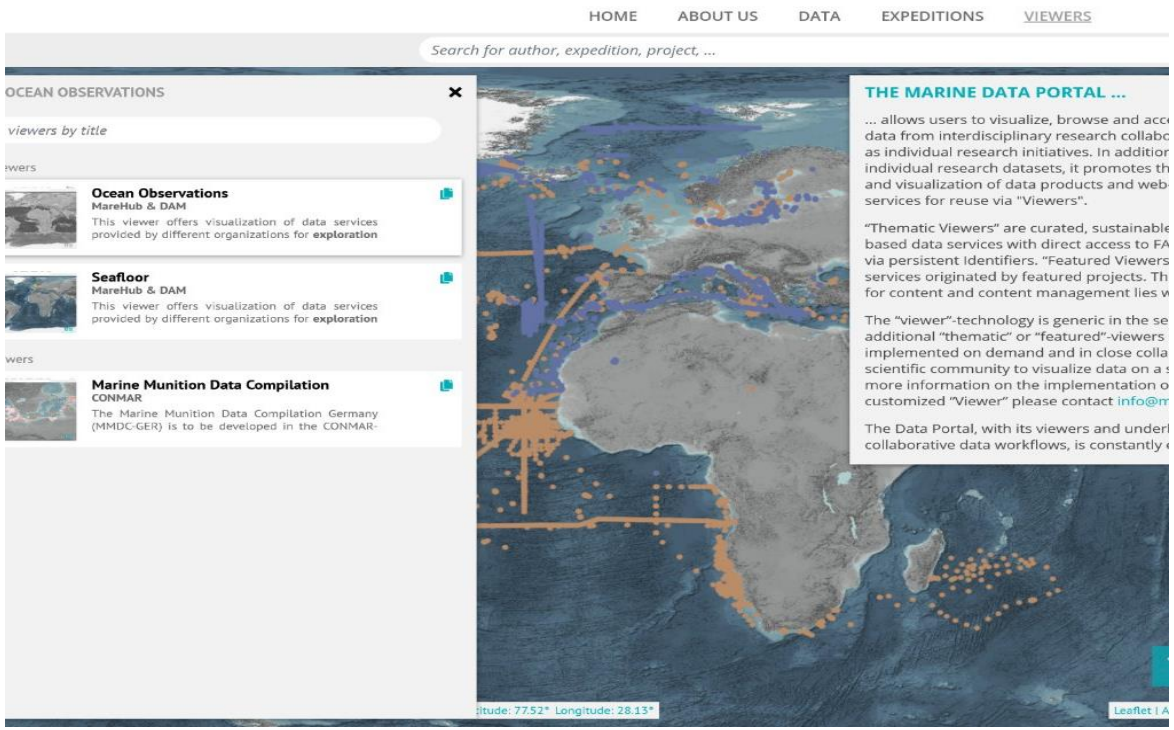
FAIR

Findable, **A**ccessible, **I**nteroperable, **R**eusable

DOI 10.5281/zenodo.10929757

Helmholtz-Zentrum hereon

Current state of thematic and featured viewers



https://marine-data.de/?site=viewer

- Thematic viewers highlight project specific content
- Highly customizable: Each user can create own viewer with the selected highlighted items

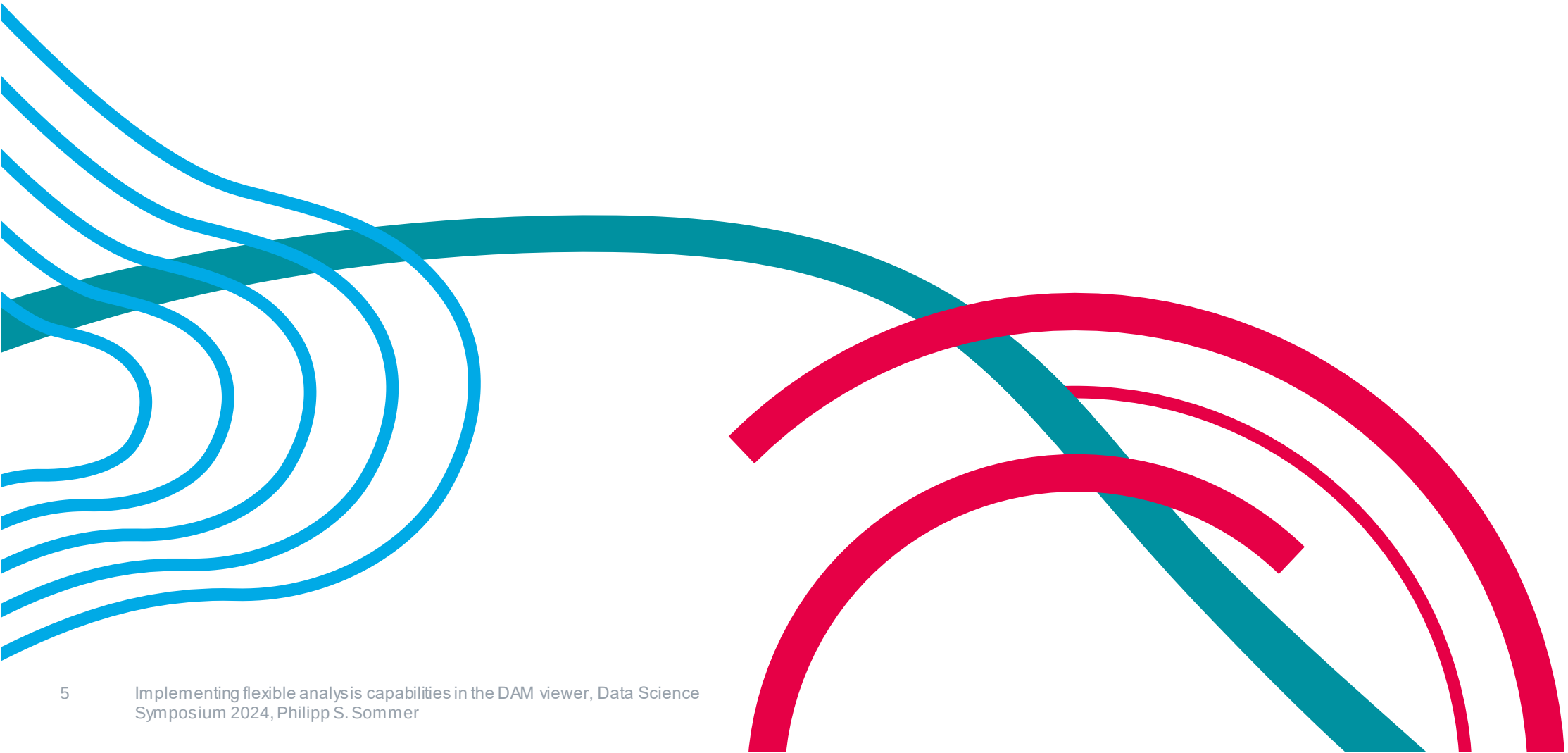
- **But:** currently there is only visualization of pre-computed data possible
  - No data analysis or customizable download

- Data Analysis and Download of selected data is a necessary requirement for when making large amounts of data available (e.g., climate model data)

DOI 10.5281/zenodo.10929757

Helmholtz-Zentrum hereon

How to implement analysis and data extraction features?



Possible use cases

Ship campaign

- Sonne (Geomar) and Ludwig Prandtl (Hereon) measure real-time-data in a campaign
- Sonne sends data to IT of Geomar, Ludwig Prandtl to Hereon

- How can people from Hereon access and analyze the data at Geomar (before it is published at Pangaea)?

Model Simulations

- Model-Intercomparison project with models from AWI, Hereon and Geomar
- Each research center runs it's own simulations with the model they know best

- How can people from Hereon access data from Geomar or AWI?
- How can we make this data accessible to the general public?

Making Analysis Features accessible is important!

- **for collaboration among scientists**
- **for knowledge transfer to stakeholders**

DOI 10.5281/zenodo.10929757

Helmholtz-Zentrum hereon

We do not have a cloud for everything

The ideal world: We all have one single big HGF cloud

generated with AI

- We run model simulations in the cloud
- Store NRT data in the cloud
- Post processing and data analysis runs in the cloud

Sharing data in the ideal world

- Someone from Hereon needs access to data from Geomar?
  - *No problem, just grant it!*

What we need:

- Access to data in another research center
- Access to computing power in another research center

And:

- It must be safe
- It must be easy
- It must be flexible

The real world: We have many different solutions

generated with AI

- Each research center (or even each individual scientist?) has different requirements
- Our IT is behind VPNs
- Each center has it's own IT for data analysis, processing, storage, etc.

Sharing data in the real world

- Someone from Hereon needs access to data from Geomar?
  - *Ok, I upload it to Dropbox.*

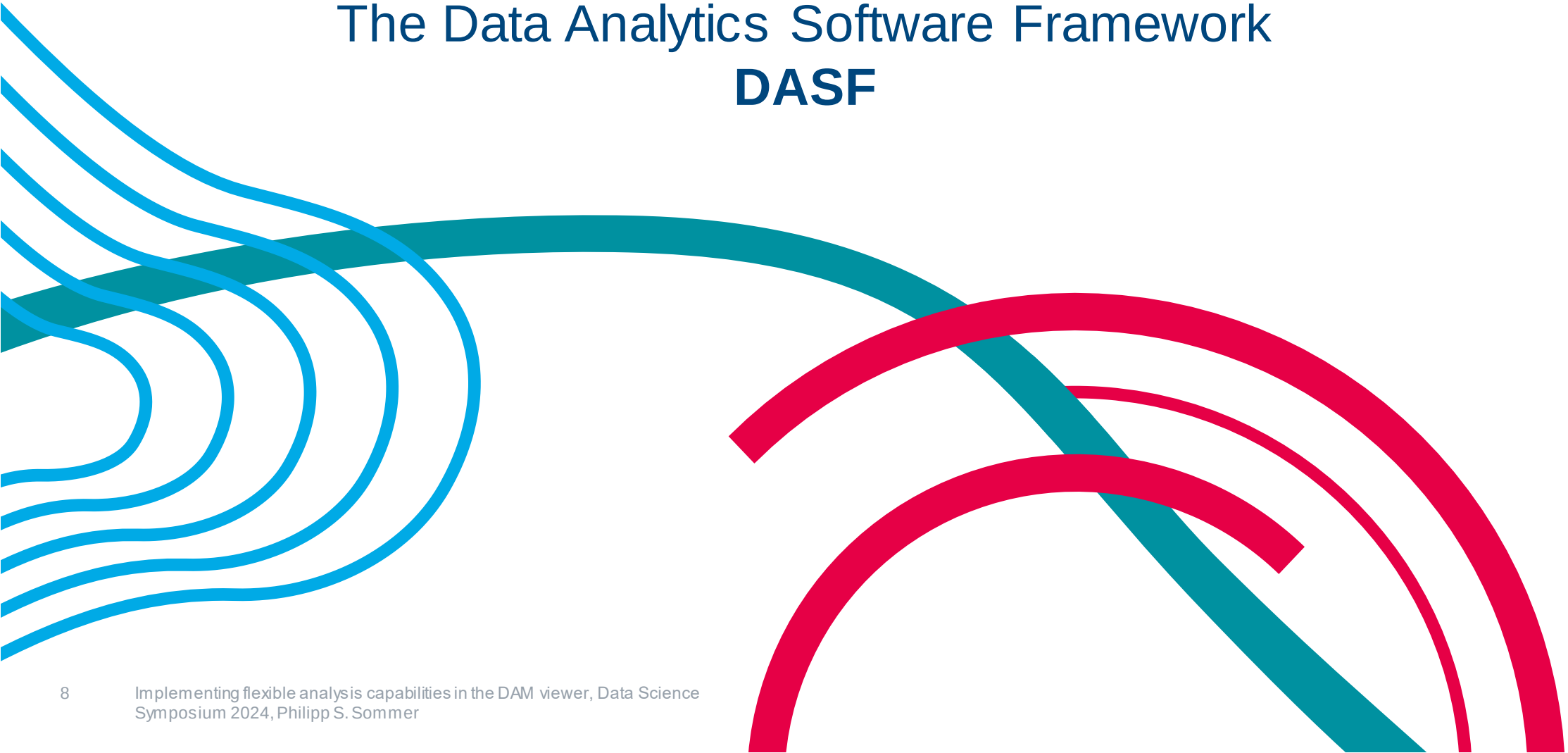
DOI 10.5281/zenodo.10929757

Helmholtz-Zentrum hereon

Our solution

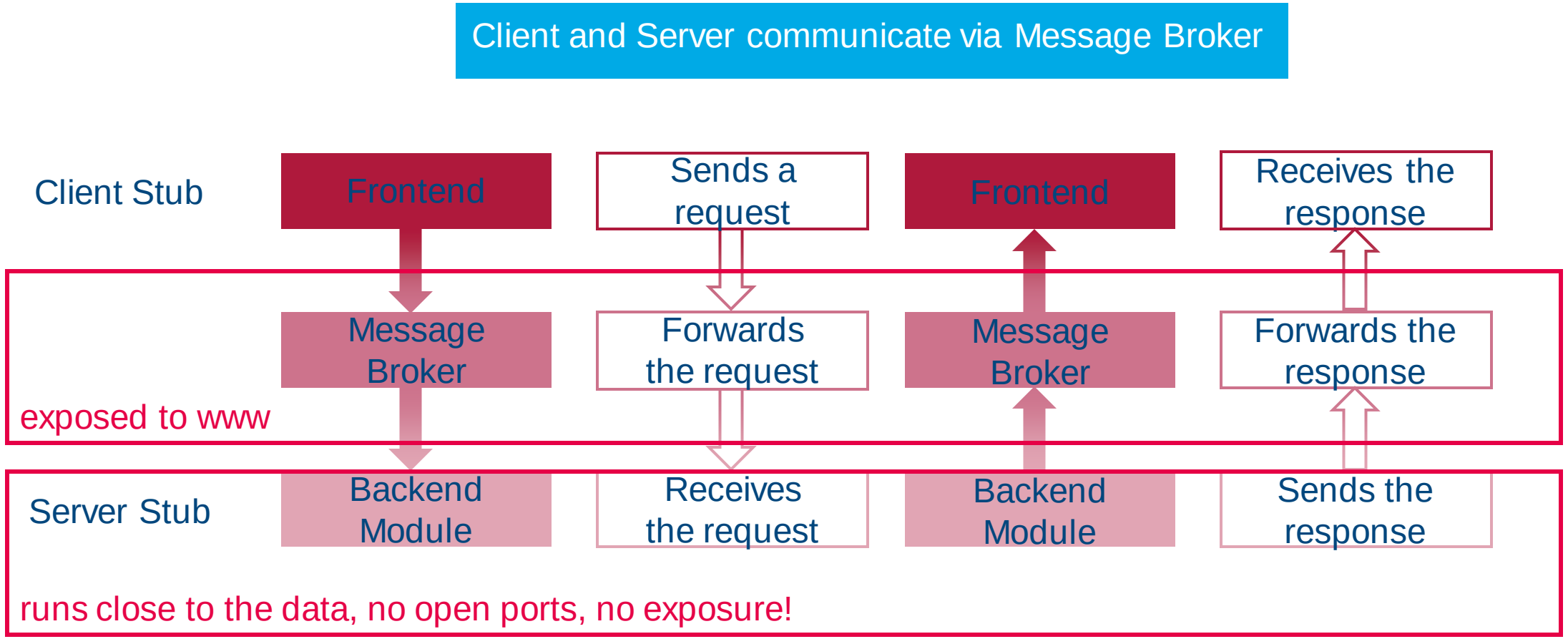
The Data Analytics Software Framework

DASF



Basic Messaging Workflow:

A remote procedure call with a man-in-the-middle



DOI 10.5281/zenodo.10929757

Helmholtz-Zentrum hereon

Usage of DASF

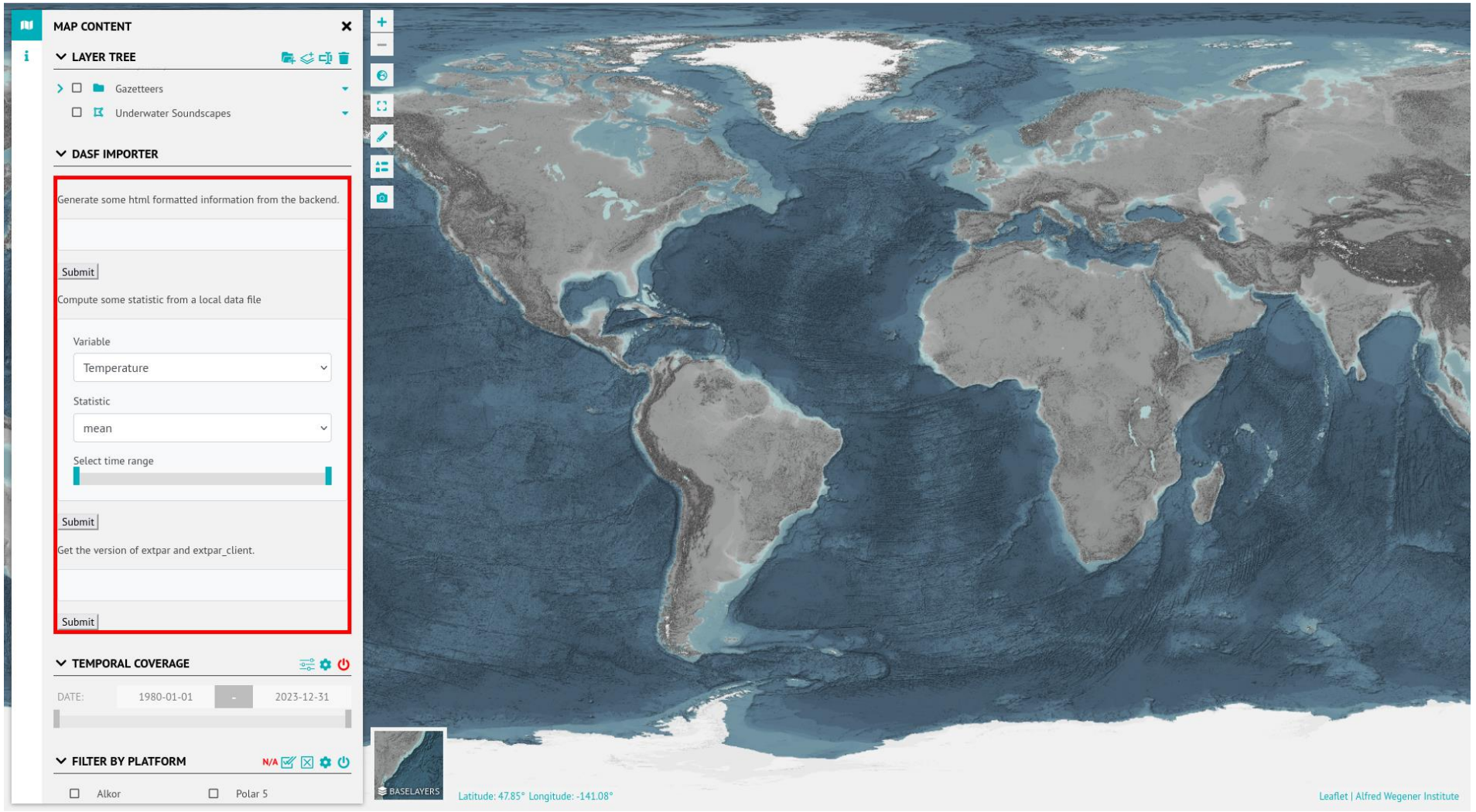
- **Core concept:** **Reduce overhead of messaging framework to an absolute minimum**
- **Strategy:** **Use pythons type annotations for validation and serialization**

- **Implementation:**
  - abstract standard python functions and classes into web requests
  - everything's basic python, (almost) no need for special stuff
  - Client stub is automatically generated
  - Requests are abstracted and standardized as JSONschema

```python
def compute_statistic(
    def compute_statistic(
        variable: Literal[
            "Temperature", "Meridional wind-velocity", "Zonal wind-velocity"
        ],
        tmin: condatetime(ge=_tmin, le=_tmax) = _tmin,  # type: ignore
        tmax: condatetime(ge=_tmin, le=_tmax) = _tmax,  # type: ignore
    )
        statistic: Literal["mean", "std"] = "mean",
    ) -> dam_dasf_demo.frontend_models.TimeSeriesModel:
        """Compute some statistic from a local data file.

        Parameters
        ----------
        variable : str
            The variable to calculate the statistic for.
        tmin: datetime.datetime
            The lower limit of the time interval to calculate the `statistic` for.
        tmax: datetime.datetime
            The upper limit of the time interval to calculate the `statistic` for.
        statistic: str
            The statistic to calculate.

        Returns
        -------
        TimeSeriesModel
            The computed time series for the specified `variable` with the given
            `statistic` in the given time interval.
        """
        request = {
            "func_name": "compute_statistic",
            "variable": variable,
            "tmin": tmin,
            "tmax": tmax,
            "statistic": statistic,
        }

        model = BackendModule.model_validate(request)
        response = model.compute()

        return response.root  # type: ignore
```


DOI 10.5281/zenodo.10929757

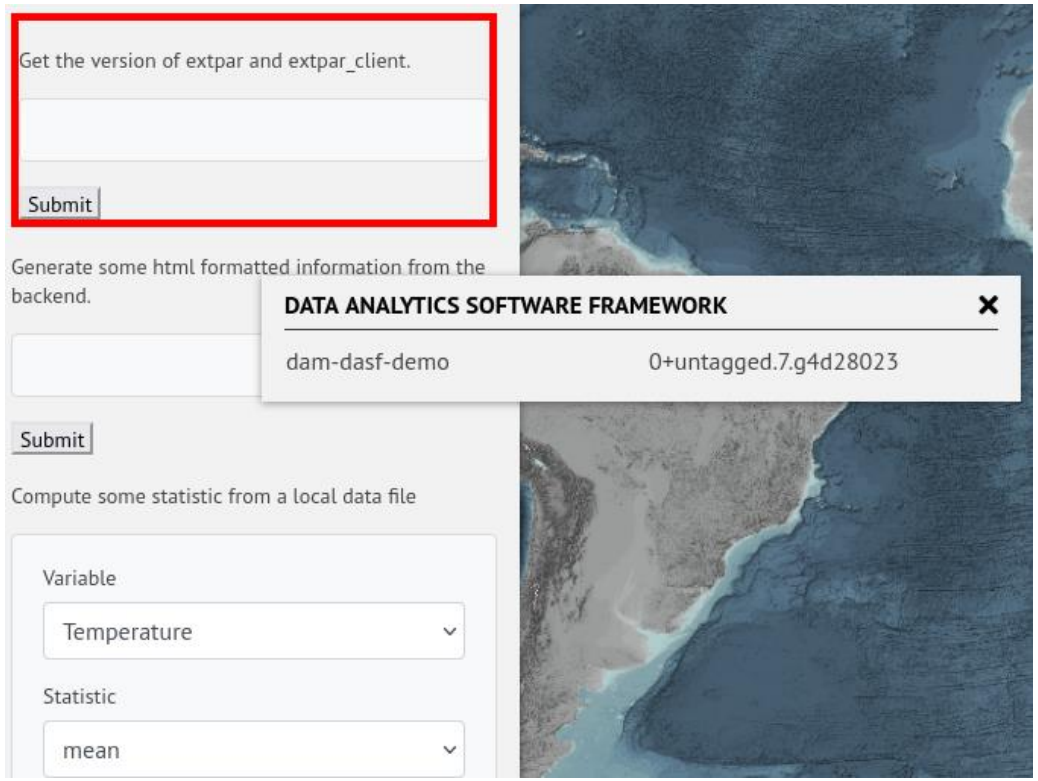
Implementation of DASF in the VEF



DOI 10.5281/zenodo.10929757

Helmholtz-Zentrum hereon

Display objects

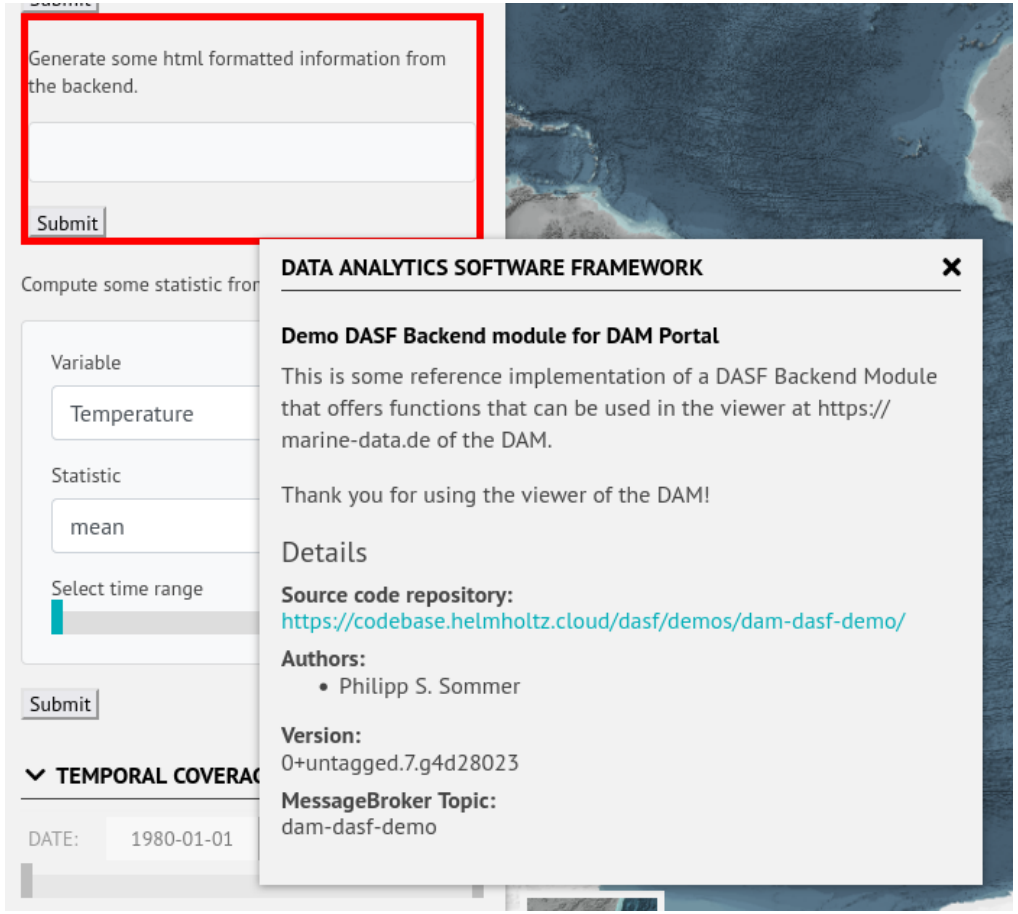



```python
__all__ = ["version_info", "html_info", "compute_statistic"]


def version_info() -> Dict[str, str]:
    """Get the version of the dam-dasf-demo."""
    import dam_dasf_demo

    info = {
        "dam-dasf-demo": dam_dasf_demo.__version__,
    }
    return info
```


DOI 10.5281/zenodo.10929757

Helmholtz-Zentrum hereon

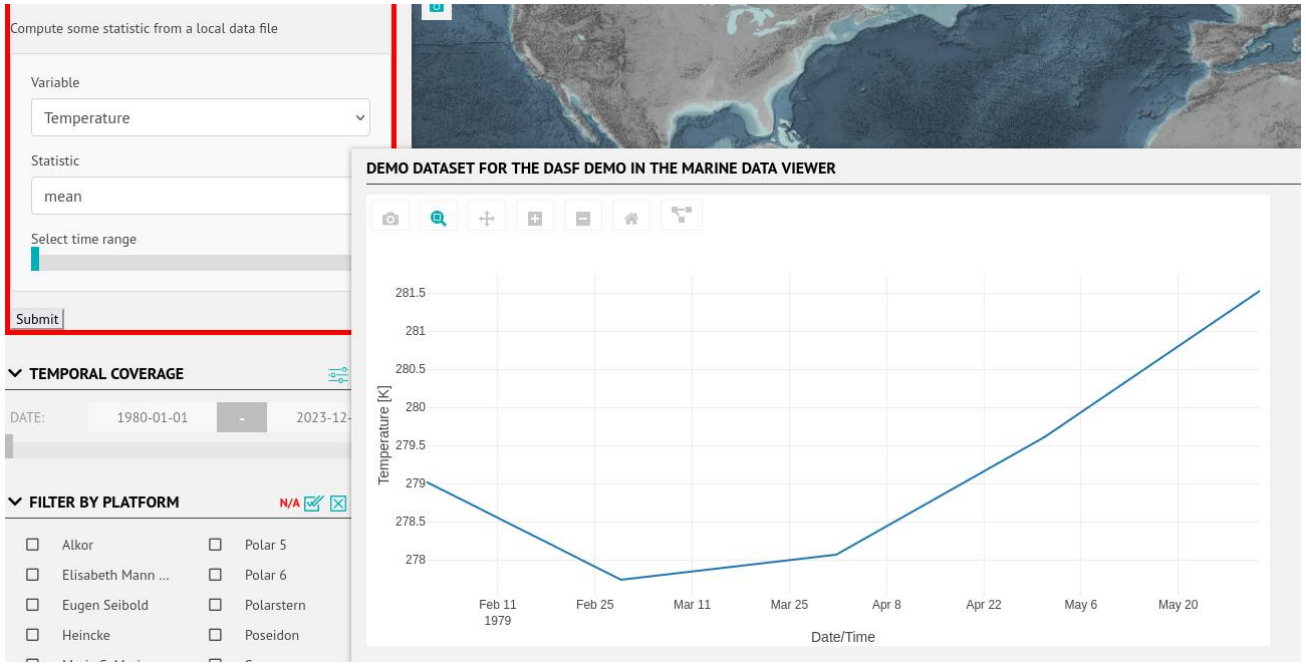
Custom HTML text




```python
def html_info() -> str:
    """Generate some html formatted information from the backend.

    Returns
    -------
    str
        The HTML info on the backend.
    """
    from demessaging.config import WebsocketURLConfig

    from dam_dasf_demo import __credits__ as credits
    from dam_dasf_demo import __version__ as version

    messaging_config = WebsocketURLConfig(
        topic=os.getenv("DE_BACKEND_TOPIC", "dam-dasf-demo")
    )
    env = Environment(
        loader=PackageLoader("dam_dasf_demo"), autoescape=select_autoescape()
    )
    template = env.get_template("html_info.html")
    return template.render(
        credits=credits,
        version=version,
        topic=messaging_config.topic,
    )
```


DOI 10.5281/zenodo.10929757

Visualize time series




```python
@add_widgets(TimeRange(tmin="tmin", tmax="tmax"))
def compute_statistic(
    variable: Literal[
        "Temperature", "Meridional wind-velocity", "Zonal wind-velocity"
    ],
    tmin: condatetime(ge=_tmin, le=_tmax) = _tmin,  # type: ignore
    tmax: condatetime(ge=_tmin, le=_tmax) = _tmax,  # type: ignore
    statistic: Literal["mean", "std"] = "mean",
) -> TimeSeriesModel:
    """Compute some statistic from a local data file

    Parameters
    ----------
    variable : str
        The variable to calculate the statistic for.
    tmin: datetime.datetime
        The lower limit of the time interval to calculate the `statistic` for.
    tmax: datetime.datetime
        The upper limit of the time interval to calculate the `statistic` for.
    statistic: str
        The statistic to calculate.

    Returns
    -------
    TimeSeriesModel
        The computed time series for the specified `variable` with the given
        `statistic` in the given time interval.
    """
    import xarray as xr

    demo_file = os.getenv("DAM_DASF_DEMO_FILE")
    ds = xr.open_dataset(demo_file)
    # some computation...

    ret = TimeSeriesModel(
        xdata=times,
        ydata={variable: stat.values.tolist()},
        title=ds.attrs["title"],
    )
    ret.layoutOptions["yaxis"] = {
        "title": "%(long_name)s [%(units)s]" % ds[var].attrs
    }
    return ret
```


DOI 10.5281/zenodo.10929757

Helmholtz-Zentrum hereon

Outlook

Where are we now?

- Implemented anonymous connection to backend module via arbitrary message broker
- implement automated rendering of forms based upon the capabilities of the backend
- Implemented basis for response type handler to be able to display various kinds of responses
- Implemented basis for custom widgets (time range slider)

What is still missing?

- Layout, layout, layout
- more custom widgets (e.g. to select a bounding box in the map)
- response handlers to add data directly to the map
- authentication against message broker via OAuth to be able to restrict access to backend module

DOI 10.5281/zenodo.10929757

Helmholtz-Zentrum hereon

Thank you!

Live demo
https://marine-data.de/preview/vef/usecases/viewer.php#dasf

Demo Backend Module
https://codebase.helmholtz.cloud/dasf/demos/dam-dasf-demo

Dr. Philipp S. Sommer

Helmholtz Coastal Data Center (HCDC)
Helmholtz-Zentrum Hereon

Max-Planck-Straße 1 | 21502 Geesthacht
T +49 4152 87-2126
philipp.sommer@hereon.de

www.hereon.de

DOI 10.5281/zenodo.10929757

ALFRED-WEGENER-INSTITUT
HELMHOLTZ-ZENTRUM FÜR POLAR- UND MEERESFORSCHUNG

DAM DEUTSCHE ALLIANZ MEERESFORSCHUNG

DataHub
Research Field Earth and Environment

Helmholtz-Zentrum hereon