Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Bundesamt für Landestopografie swisstopo

wissen wohin
savoir où
sapere dove
knowing where

swisstopo

Publikation der Meteo- und Klimadaten von MeteoSchweiz über die Publikationsplattform BGDI

Publication des données météorologiques et climatiques de MétéoSuisse via la plateforme de publication IFDG

Publikationsplattform BGDI

Plateforme de publication IFDG

*.geo.admin.ch

map.geo.admin.ch

api.geo.admin.ch

OGC®
Making location count.

data.geo.admin.ch

STAC
SpatioTemporal
Asset Catalog

BGDI: Bundes Geodaten-Infrastruktur

IFDG: Infrastructure fédérale de données géographiques

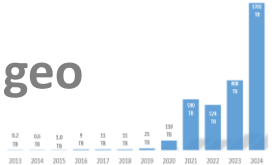
Nutzung *.geo.admin.ch

Nutzung seit 2010
Utilisation depuis 2010

map.geo

wm(t)s.geo

data.geo



Aktuelle 24h Nutzung data.geo.admin.ch

Data Volume

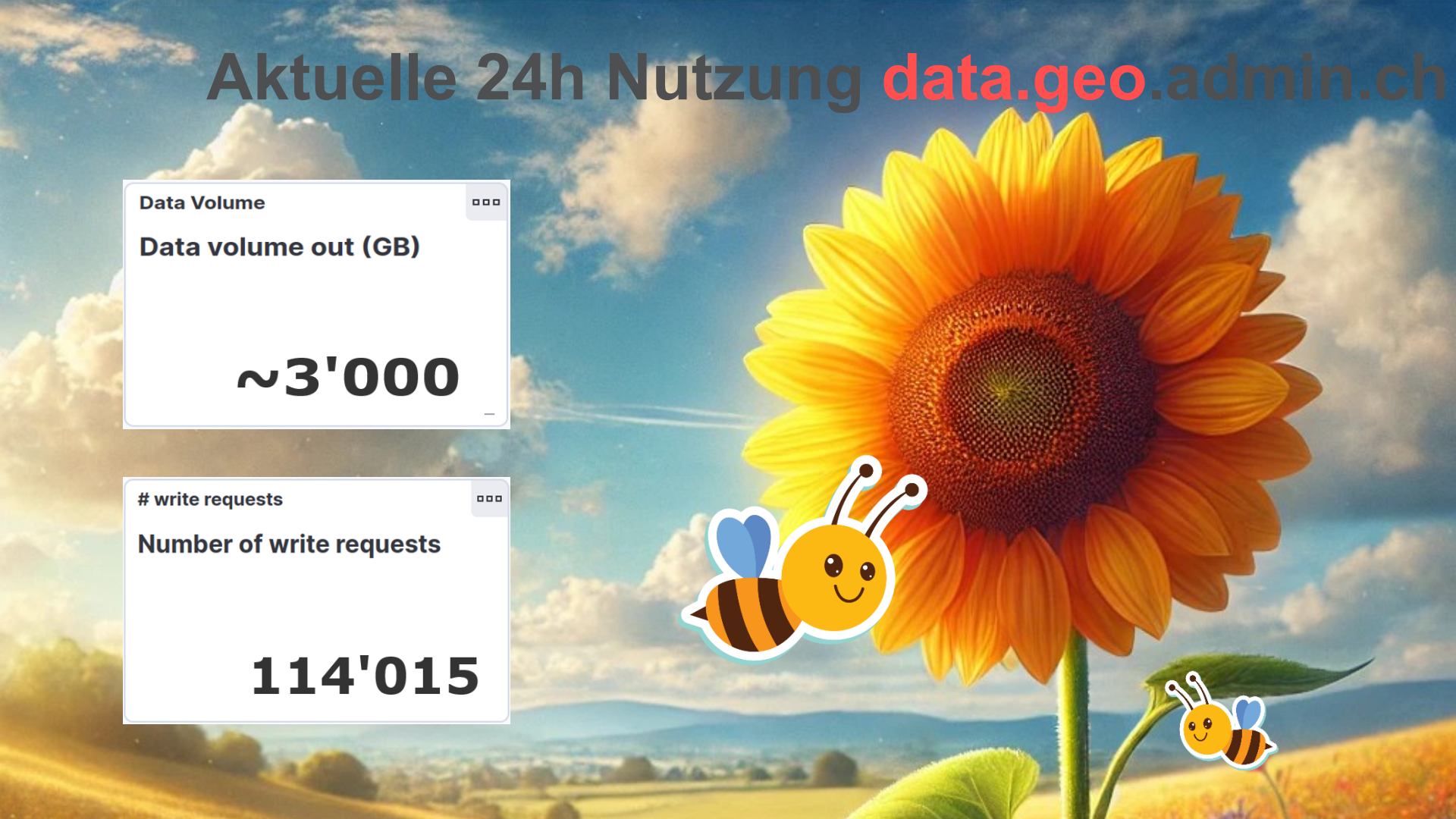
Data volume out (GB)

~3'000

 # write requests

Number of write requests

114'015




Mehr als 900 Geodatensätze des Bundes und weiterer Partner sind auf geo.admin.ch publiziert.
Plus de 900 géodonnées de la Confédération et d'autres partenaires sont publiées sur geo.admin.ch.

OGD swisstopo 2021

- Grosse Datenmengen strukturiert bereitstellen
- Generische Lösung
- Existierender Standard mit etablierter Nutzercommunity und Tooling

Referenten / Intervenants

Christian Lukasczyk MeteoSchweiz
Leiter Verkauf und Support, Co-Projektleiter OGD24

Andreas Amsler MeteoSchweiz
OGD Steward, Leiter Teilprojekte Daten & Technologie OGD24

Jürgen Hansmann swisstopo
Product Owner PP BGDI Plattform

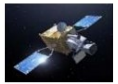
Christoph Böcklin swisstopo
Technischer Leiter PP BGDI

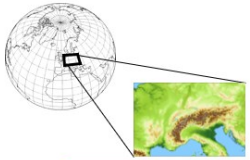
Stefan Biegger swisstopo
Leiter Publikationsplattform BGDI (PP BGDI)

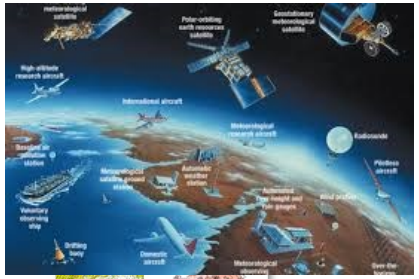

Einführung Vorhaben
Neue Datenprodukte
Life of a Meteo Dataset
Technische Aspekte
Datenprodukte nutzen
Ausblick
API


Einführung Vorhaben
Neue Datenprodukte
Life of a Meteo Dataset
Technische Aspekte
Datenprodukte nutzen
Ausblick
API

Einführung





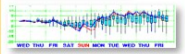


Anforderungen

Messungen & Datenerhebung

Modelle & Vorhersagen

Verarbeitung & Produktion



Daten & Informationen

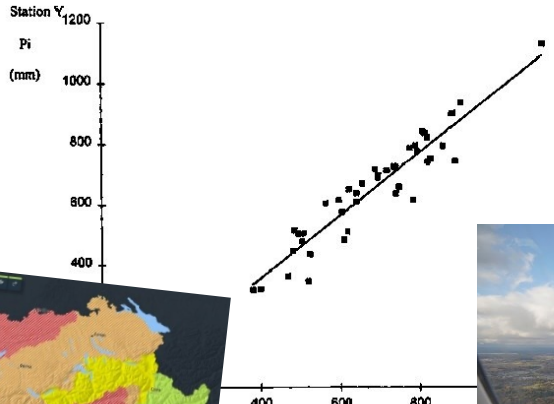
Interpretation & Beratung

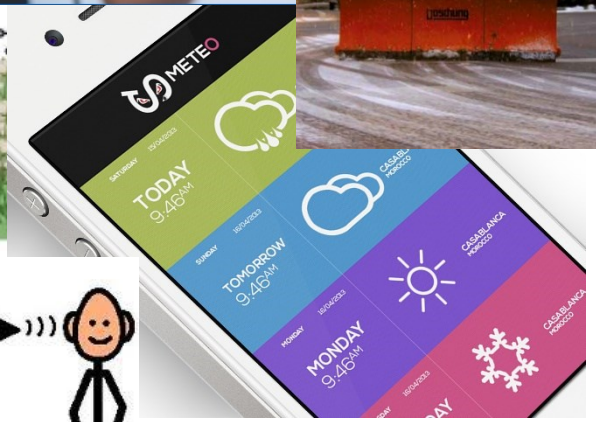
Distribution & Datenzugang

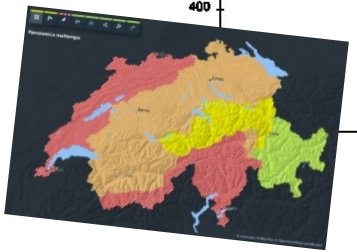
Anwendungen & Handeln

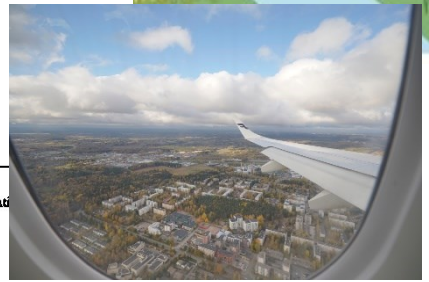
Rückmeldung & Verifikation

Anwendungen









Kundensegmente

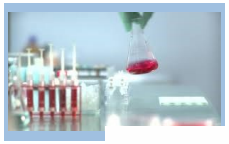
Schweizer Bevölkerung

Freizeit-Aviatik

Behörden mit Bevölkerungs-schutz

Öffentliche Hand

Lehre und Forschung



Wirtschaft

Internationale Organisationen

WMO

WORLD METEOROLOGICAL ORGANIZATION

Private Wetterfirmen

meteoradar dr. willi schmid

MeteoGroup

METEODAT

meteor

oblue weather prediction

METEOTEST

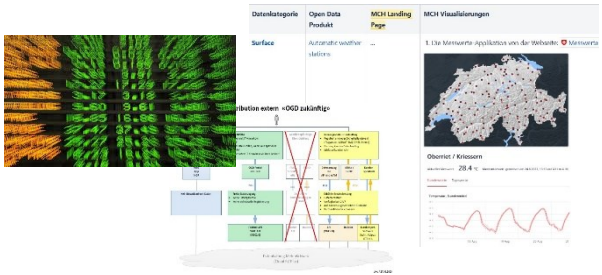
SRF

Ziele Projekt OGD24

Revision Verordnung MetV

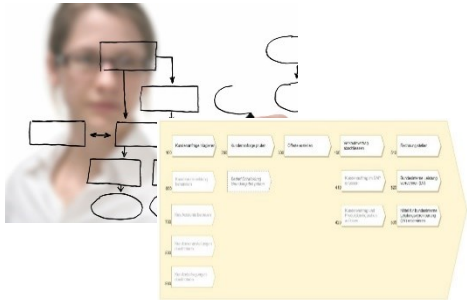
Dokumentation / Metadaten

Personalentwicklung - Training



Datendistribution neu API - Pull

Kompatibilität EU
High Value Datasets



Prozesse neu für Self Service

Ein realer Anwendungsfall

Foto von Bermix Studio auf Unsplash


Einführung Vorhaben
Neue Datenprodukte
Life of a Meteo Dataset
Technische Aspekte
Datenprodukte nutzen
Ausblick
API

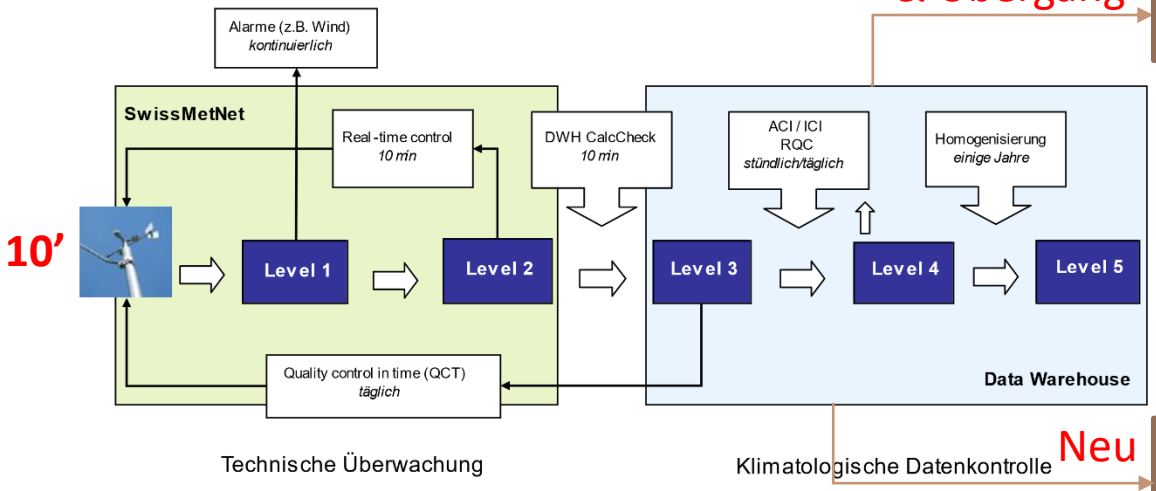
Meteo- und Klimadaten

Neue Datenprodukte – *Planung 'subject to change'*

	Messdaten		Klimadaten	Radardaten	Prognosedaten
	Boden	Atmosphäre			
Punkt	8 Datensätze	Radiosondierungen	Homogene Messdaten		Lokal-prognose
			Normwerte		
Raster			Bodengestützte	Combiprecip	Numerisches Wettermodell
			Satellitengestützte		
			Radargestützte	weitere	Kurzfrist-prognose
			Normwerte		
			Klimaszenarien		


Einführung Vorhaben
Neue Datenprodukte
Life of a Meteo Dataset
Technische Aspekte
Datenprodukte nutzen
Ausblick
API

Datenverarbeitungskette MeteoSchweiz



Bisher & Übergang

Push

DWHretrieve | Dadeco | MHS | Mail ftp

Ziel: Zeitnahe Bereitstellung:
< Messintervall (5-9 Min)

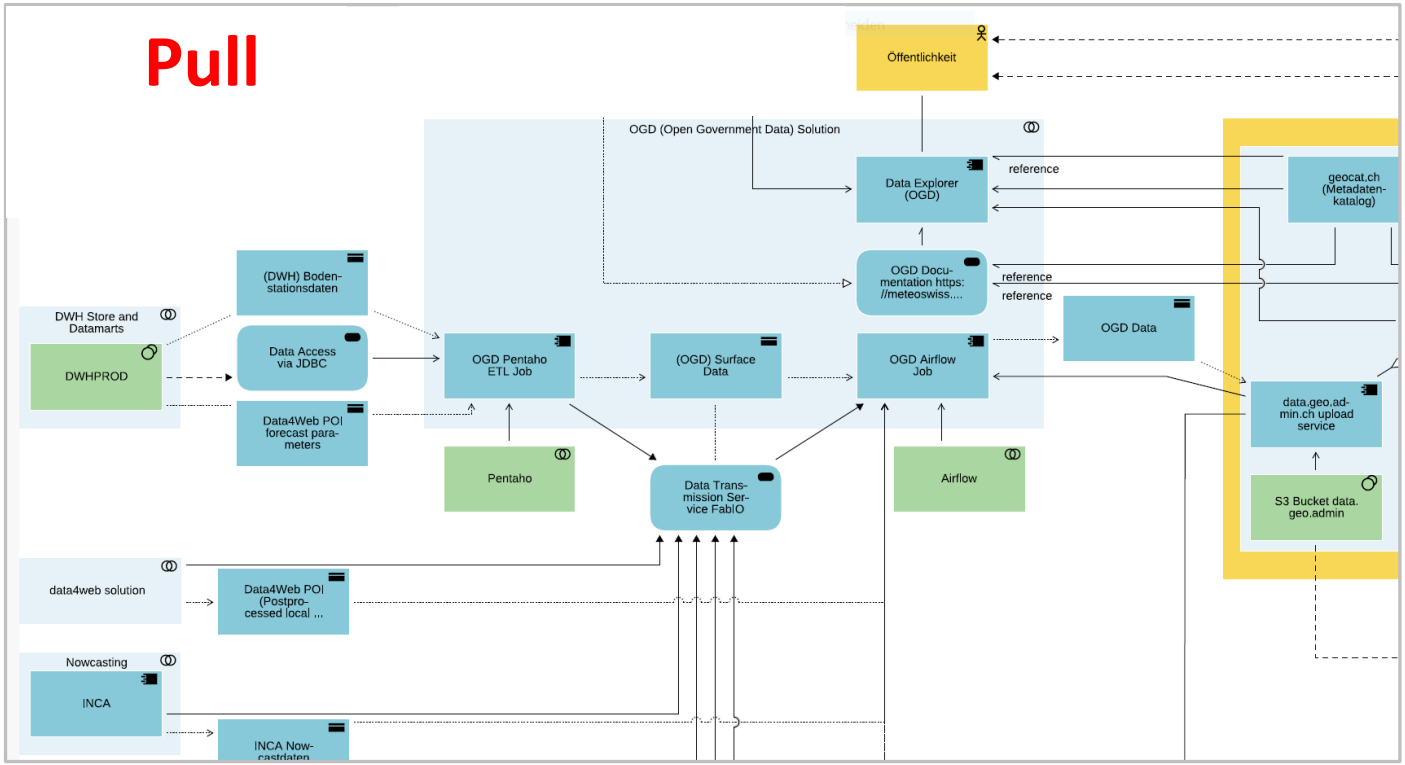
Note: Nachbearbeitung!

Neu

Pentaho | FabIO MinIO | Airflow | BGDI CSCS S3 | API

Pull

OGD-Generierung und -Integration



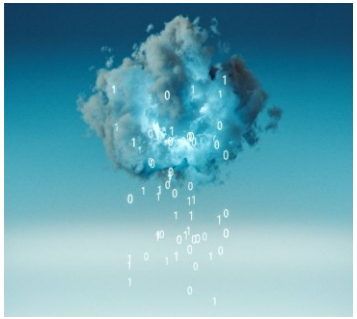
© Quelle MeteoSchweiz


Einführung Vorhaben
Neue Datenprodukte
Life of a Meteo Dataset
Technische Aspekte
Datenprodukte nutzen
Ausblick
API

SpatioTemporal Asset Catalog

STAC / STAC API

data.geo.admin.ch



OGD Meteoschweiz

STAC / STAC API

data.geo.admin.ch

OGD Meteoschweiz

Breite Nutzerbasis STAC / STAC API

MAXAR

sentinel-2

Landsat 8

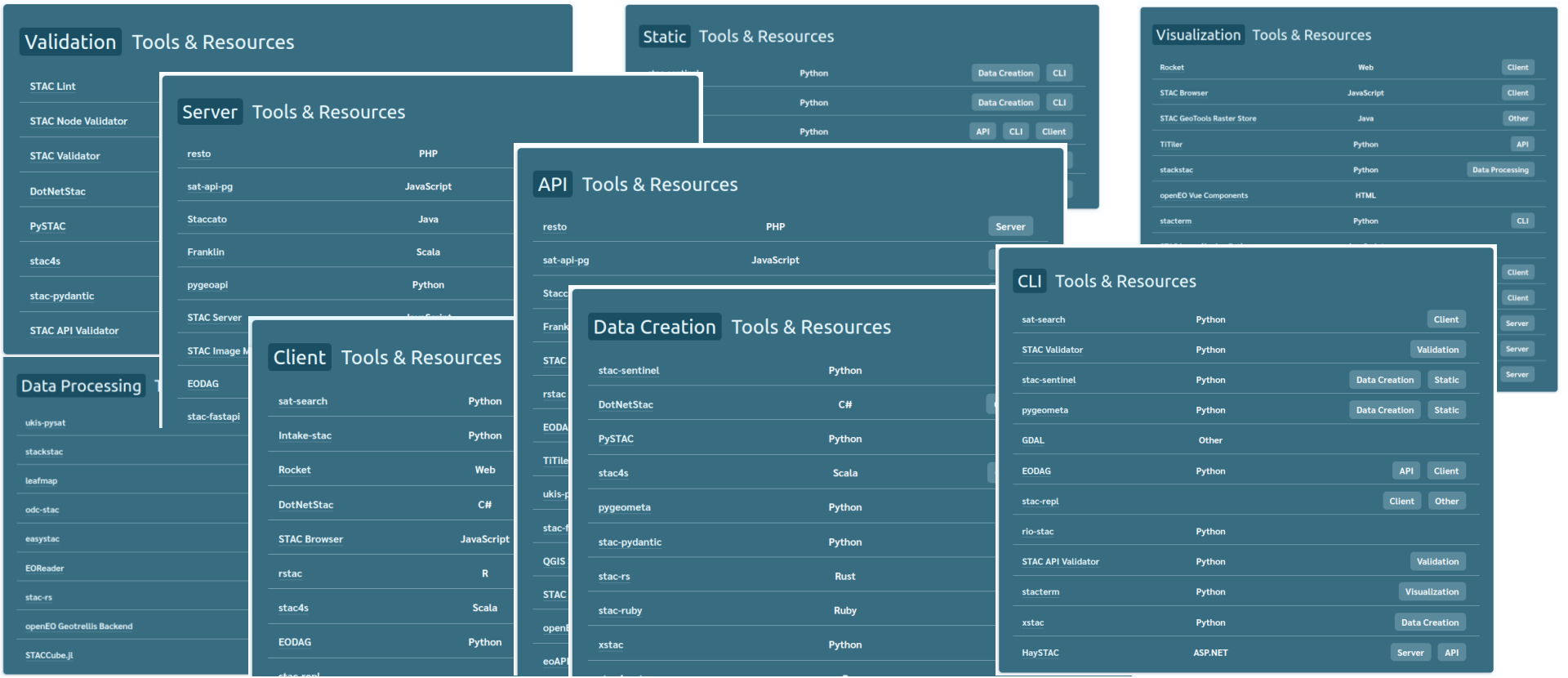
NASA

Google Earth Engine

data.geo.admin.ch

Microsoft Planetary Computer

Stac Tools & resources

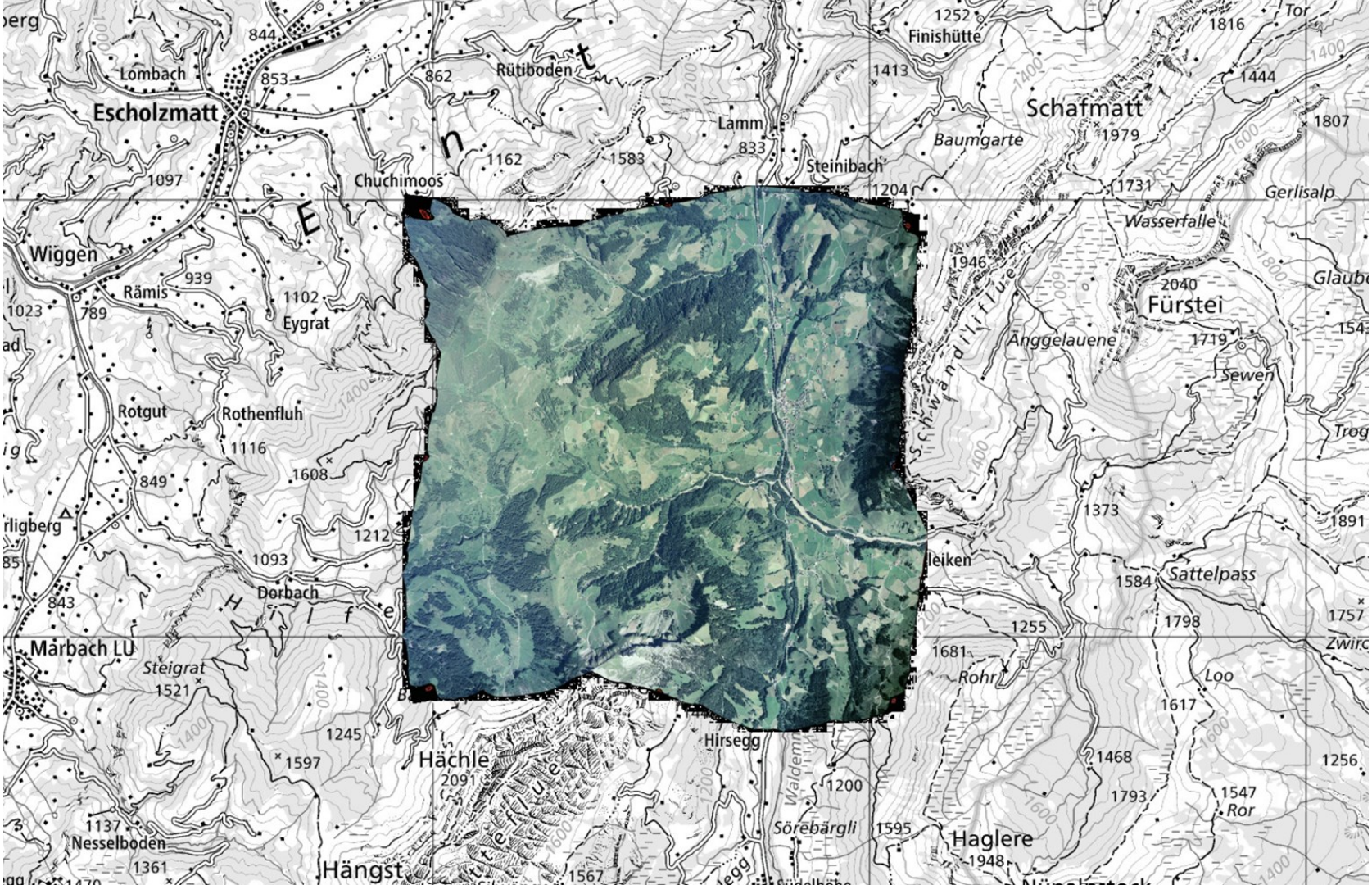


https://stacspec.org/en/about/tools-resources/

https://stacspec.org/ https://stacspec.org/en/about/datasets/

STAC / STAC API

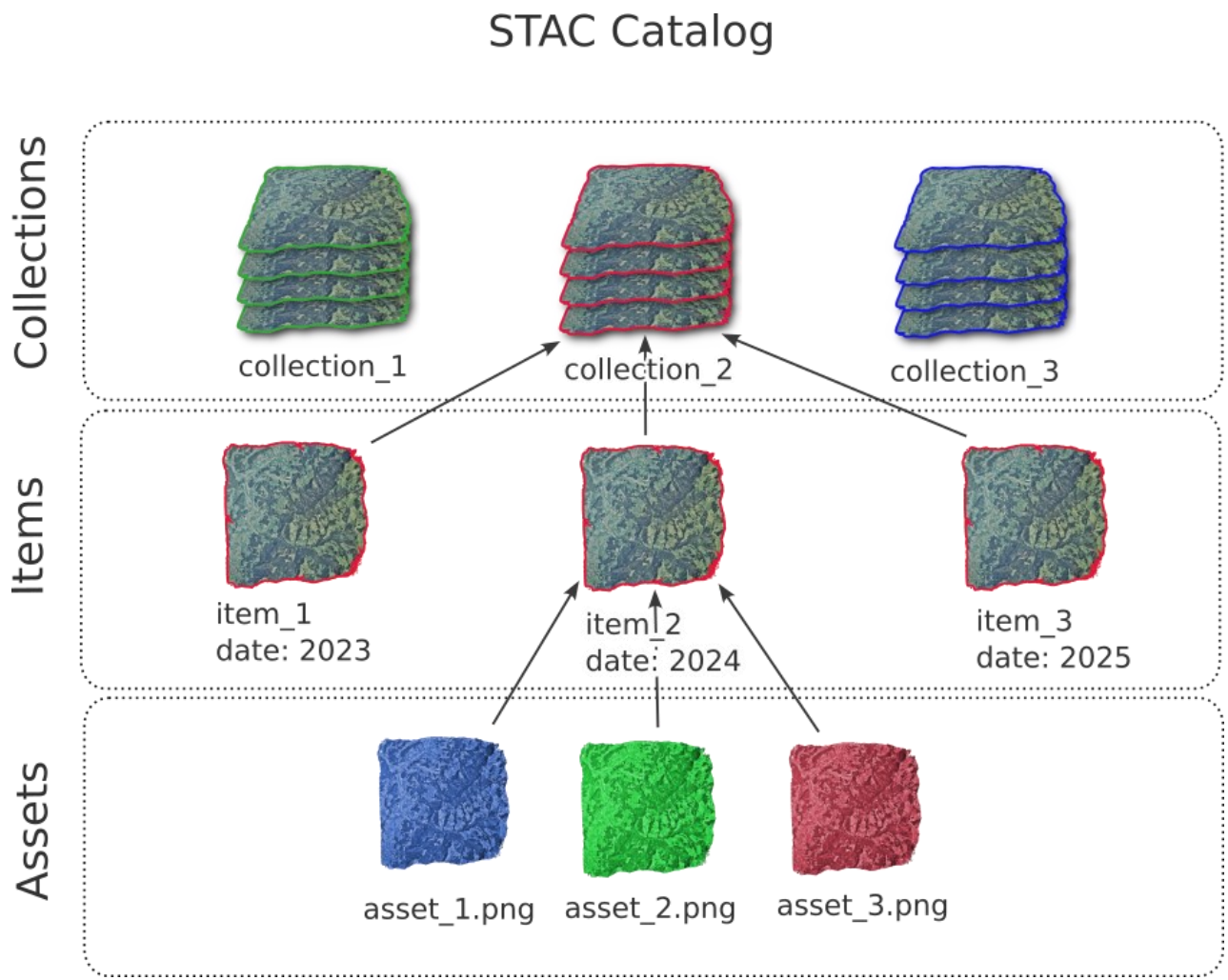
- STAC = SpatioTemporal Asset Catalogs
- STAC Spezifikation bietet eine gemeinsame Struktur für die **Beschreibung und Katalogisierung von “raum-zeitlichen” Assets**.
- Ein raumbezogenes Asset ist eine Datei, die Informationen über die Erde in einem **bestimmten Raum und einer bestimmten Zeit enthält**.


Escholzmatt
Lombach
Rütiboden
Schafmatt
Baumgarte
Steinibach
Chuchimoos
Wiggen
Rämis
Eygrat
Rothenfluh
Rotgut
Fürstei
Anggelauene
Sewen
Wasserfalle
Gerlisalp
Schwändiflue
Dorbach
Hilfe
Marbach LU
Steigrat
Hächle
Hirsegg
Sattelpass
Rohr
Loo
Haglere
Sörebärgli
Hängst
Nesselboden
Waldemmeflue

Stac Spezifikation

Die wichtigsten Teile von STAC:

- **STAC Collection:** Sammlung von einem oder mehreren Items, welche logisch zusammengehören**.**
- **STAC Item**: zentrale atomare Einheit, die ein einzelnes raum-zeitliches Datenobjekt als GeoJSON-Feature plus Datetime und Links darstellt.
- **STAC Asset**: Repräsentation der Daten eines Item in einem bestimmten Dateiformat
- **STAC-API**: RESTful-Endpunkt, der die Suche nach STAC-Objekten ermöglicht


STAC Catalog
Collections
Items
Assets
collection_1
collection_2
collection_3
item_1
date: 2023
item_2
date: 2024
item_3
date: 2025
asset_1.png
asset_2.png
asset_3.png

STAC API

- Technische Schnittstelle um Daten und Metadaten zu lesen und zu schreiben
- JSON/ReST
- Erweitert statische STAC Kataloge um transaktionale Operationen und Suchmöglichkeiten
- Spezifikation folgt den OGC API Standards

Open Geospatial Consortium

SpatioTemporal Asset Catalog

STAC / STAC API

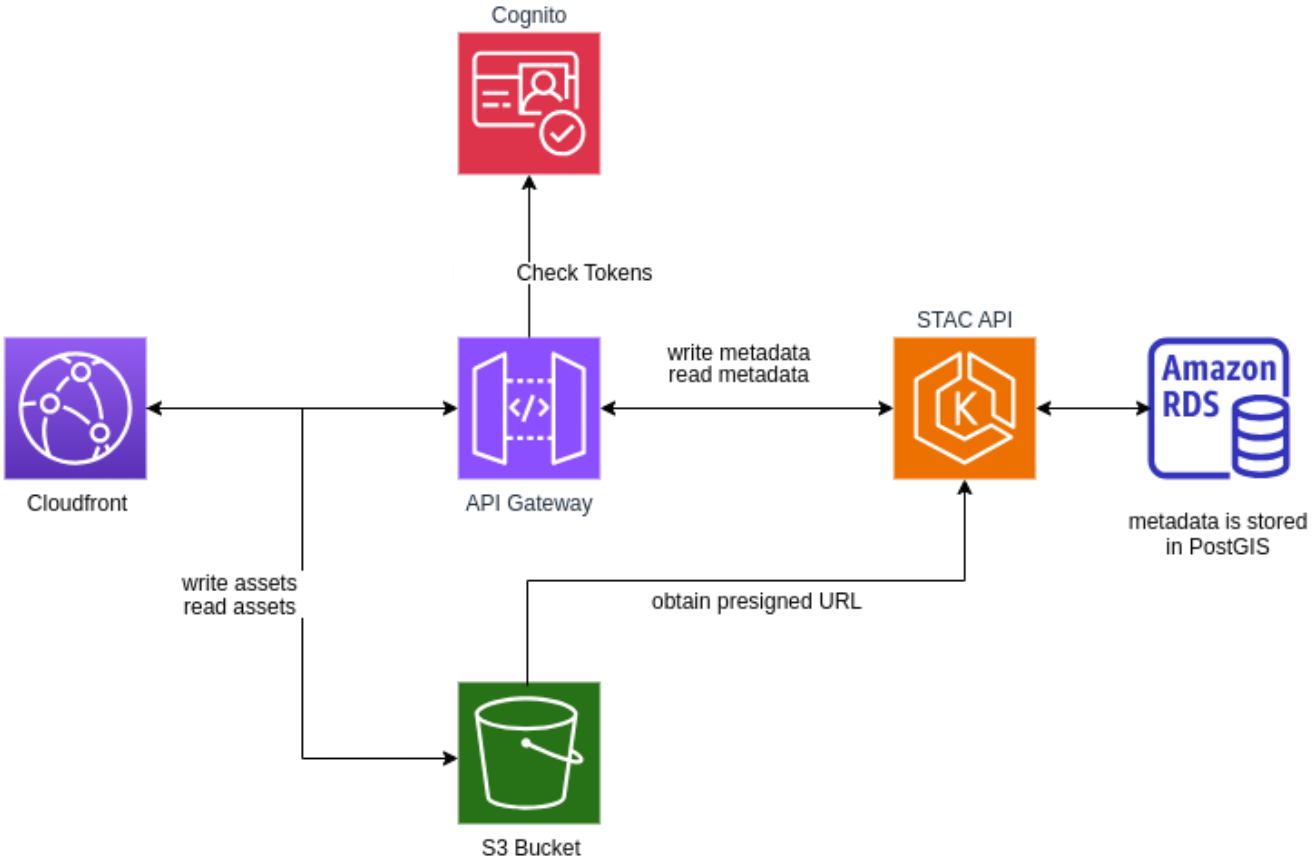
data.geo.admin.ch

OGD Meteoschweiz

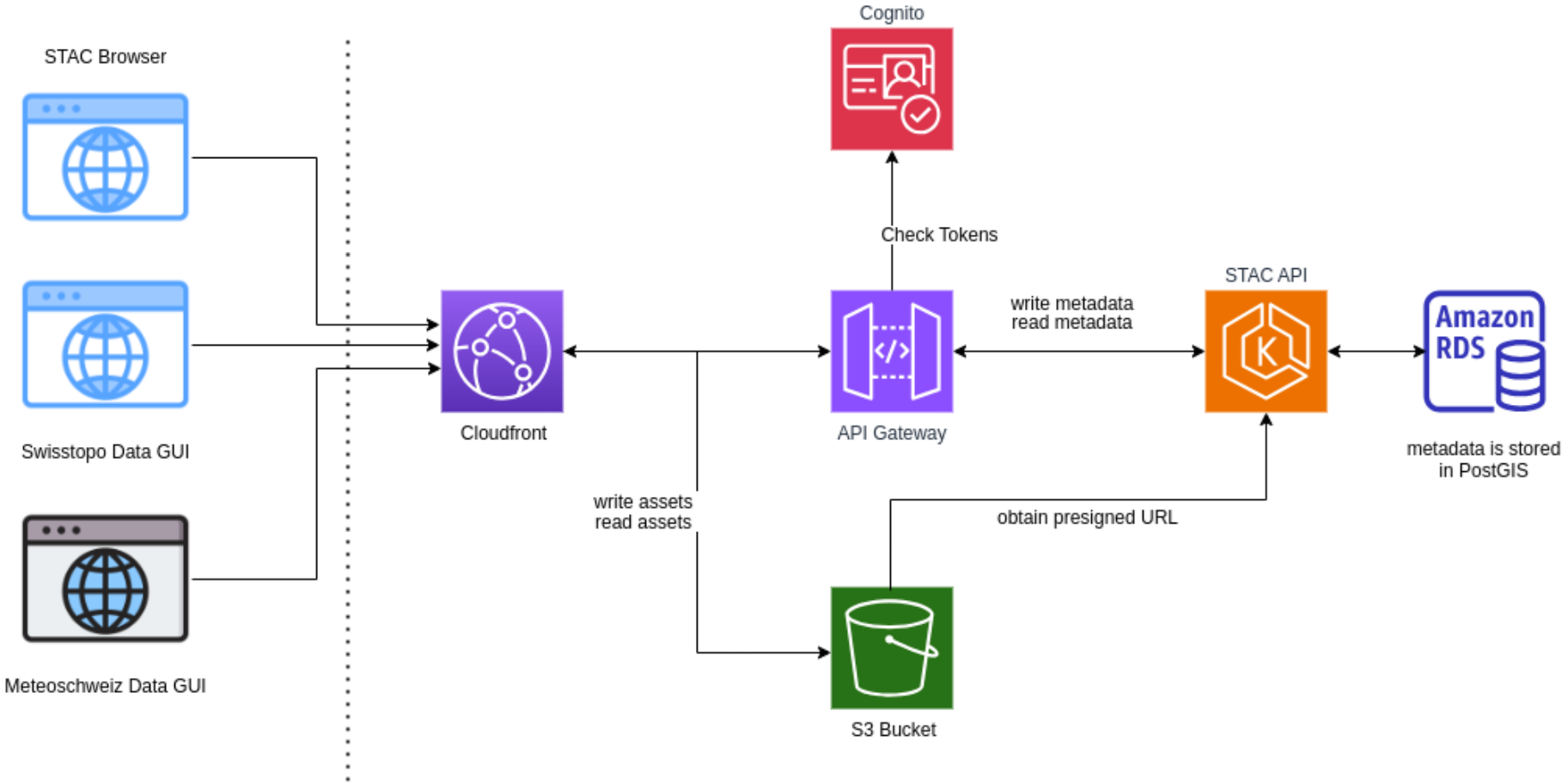
STAC API für data.geo.admin.ch

- Datenprovider sollten in der Lage sein, STAC-Metadaten und -Assets selbständig zu schreiben
- Implementation (Python/Django) der STAC-API-Spezifikation mit einer benutzerdefinierten transaktionalen Erweiterung github.com/geoadmin/service-stac
- Start Entwicklung 2020, go-live März 2021

Architektur



Generische Schnittstelle



Search...

Capabilities

Data

STAC

Data Management

Supported Media Type

PUT Update or create a collection

PATCH Partial update of a collection

POST Bulk create features

PUT Update or create a feature

PATCH Update an existing feature by Id with a partial item definition

DEL Delete an existing feature by Id

PUT Update or create a collection asset

PATCH Update an existing collection asset by Id with a partial asset definition

DEL Delete an existing collection asset by Id

API docs by Redocly

Update or create a feature

PUT /collections/{collectionId}/items/{featureId}

Update or create a feature with Id `featureId` with a complete feature definition. If the feature doesn't exist it is then created.

NOTE: Optional fields that are not part of the PUT payload, will be erased in the resource. For example if the resource has a properties.title and the PUT payload doesn't, then the resource's properties.title will be removed.

PATH PARAMETERS

collectionId (required)
string
Local identifier of a collection

featureId (required)
string
Local identifier of a feature

HEADER PARAMETERS

If-Match
string
Example: `d01af8b8ebbf899e30095be8754b377ddb0f0ed0f7fddbc33ac23b0d1969736b`

The RFC7232 `If-Match` header field makes the PUT/PATCH/DEL request method conditional. It is composed of a comma separated list of ETags or value "*".

The server compares the client's ETags (sent with `If-Match`) with the ETag for its current version of the resource, and if both values don't match (that is, the resource has changed), the server sends back a `412 Precondition Failed` status, without a body, which tells the client that he would overwrite another changes of the resource.

REQUEST BODY SCHEMA: application/json

Request samples

Payload

Content type
application/json

Copy Expand all Collapse all


```
{
   "id": "cs3-20160503_132131_05",
 - "geometry": {
      "type": "Polygon",
    + "coordinates": [ … ]
   },
 - "properties": {
      "datetime": "2016-05-03T13:22:30
      "title": "A CS3 item",
      "forecast:reference_datetime": "
      "forecast:horizon": "P3DT2H",
      "forecast:duration": "PT4H",
      "forecast:variable": "air_temper
      "forecast:perturbed": true
   },
 - "links": [
    + { … },
    + { … }
   ]
}
```

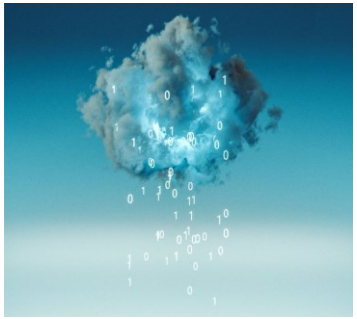

Response samples

200 201 400 403 404 500

SpatioTemporal Asset Catalog

STAC / STAC API

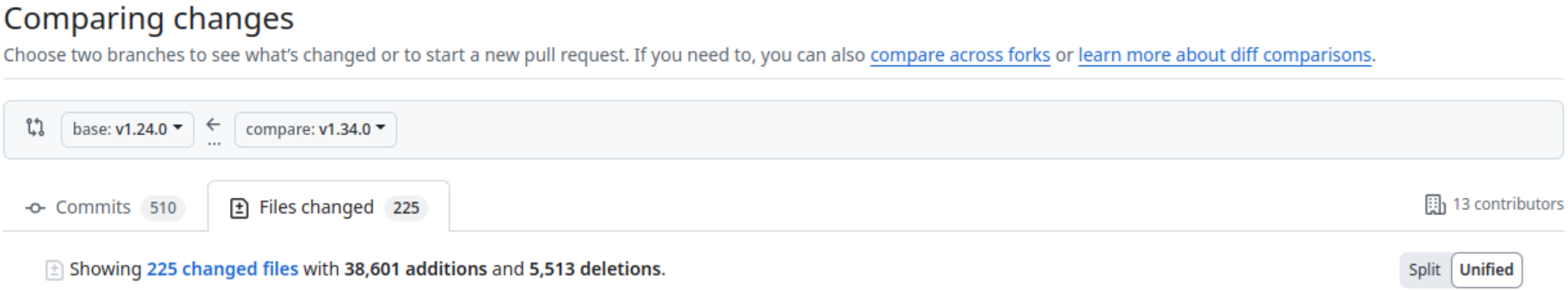
data.geo.admin.ch



OGD Meteoschweiz

Erweiterungen für MeteoSchweiz

- Zentrale Authentifizierung (AWS Cognito)
- Performance Verbesserungen (Bulk Uploads)
- Forecast Extension
- Collection Assets
- Weitere kleinere Features



Forecast Extension

- Herausforderung: wie können die Wettermodell Daten sinnvoll in STAC abgebildet werden?
  ~ 25Mio Datenpunkte / ~ 25 Tb pro Tag
- Datenpunkte decken immer die ganze Schweiz ab → räumliche Differenzierung fällt weg
- STAC erlaubt fachzpezifische Erweiterungen → Forecast Extension https://github.com/stac-extensions/forecast/

Forecast Extension

- Zusätzliche Felder in den Item Properties

```
{
  ....
  "properties": {
    ....
    "forecast:reference_datetime": "2018-02-12T23:20:50Z",
    "forecast:horizon": "P3DT6H",
    "forecast:duration": "P3DT6H",
    "forecast:variable": "air_temperature",
    "forecast:perturbed": true
}
```

- Gruppierung von Parametern führt zur Reduktion auf ca. 1Mio Items
- Items sind suchbar


Einführung Vorhaben
Neue Datenprodukte
Life of a Meteo Dataset
Technische Aspekte
Datenprodukte nutzen
Ausblick
API

Ein realer Anwendungsfall

Foto von Bermix Studio auf Unsplash

Wie und wo finden Interessierte die Daten ?

Die **Web Landing Page** von MeteoSchweiz listet alle zugänglichen / geplanten Open Data auf.

Die **Metadatenkataloge** opendata.swiss und European Data Portal verweisen auf alle zugänglichen Open Data.

- MeteoSchweiz verwaltet ihre ‘Discovery Metadata’ im **Geocat.ch**

geocat.ch
THE SWISS **GEO**GRAPHIC
CATALOGUE

Datenprodukte nutzen

1

Die **STAC API** der BGDI erlaubt es, alle Datenprodukte automatisch zu beziehen.

2

Der **‘Data Explorer’** von MeteoSchweiz erlaubt es, Boden- und homogene Messdaten zu filtern und manuell herunterzuladen.

STAC Browser

swissEO S2-SR: Optical satellite data (Sentinel-2)

Source | Share | Language: English

in data.geo.admin.ch | Up | Browse | Search

Description

Optical satellite data (Sentinel-2) which, among other things, show the reflectances of the land surface for the four channels Red, Green, Blue and near Infrared in a spatial resolution of 10 metres. Switzerland is fully imaged approximately every three days, but the usefulness of the data is heavily dependent on meteorological conditions, as the imaging sensor cannot see through clouds. In addition to the already applied localisation of the data, a co-registration of the data optimised for Switzerland is applied for a sub-pixel positional accuracy and the data is delivered in a uniform projection. Also included are optimised quality layers with masks for clouds (and cloud shadows) and topographic shadows. The data processing is still in the commission phase (set-up and validation) until the first quarter of 2025, so the data may be subject to change during this period. Contains modified Copernicus Sentinel data.

License	Terms of use
Temporal Extent	2/23/2022, 10:30:31 AM UTC - 11/2/2024, 10:42:11 AM UTC

Items

« First | ‹ Previous | Next › | Filter

swisseo_s2-sr_v100_2022-02-23t103031

2/23/2022, 10:30:31 AM UTC

Open Data automatisch beziehen

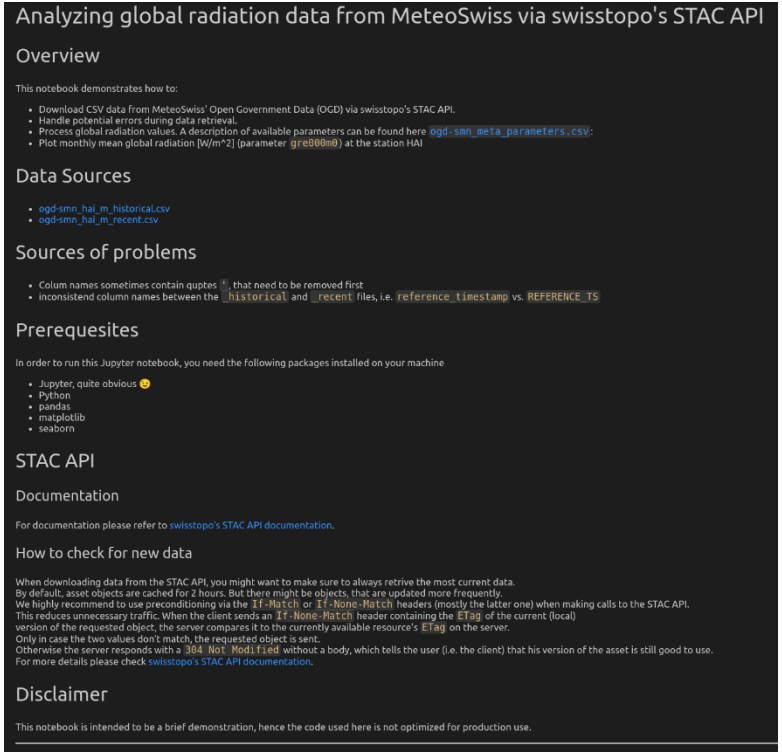
- STAC Browser
- Jupyter Notebook für automatisiertes Herunterladen und Visualisieren via STAC API

Open Data automatisch beziehen

Die beiden wichtigsten Punkte:

- Bitte «**If-None-Match**» Header mit **ETag** verwenden, um unnötigen Traffic zu vermeiden

- Jupyter-Notebook-Code für Demo, nicht für produktiven Gebrauch (dort bräuchte es z.B. ausgefeilteres error handling, etc.)



Analyzing global radiation data from MeteoSwiss via swisstopo's STAC API

Overview

This notebook demonstrates how to:

- Download CSV data from MeteoSwiss' Open Government Data (OGD) via swisstopo's STAC API.
- Handle potential errors during data retrieval.
- Process global radiation values. A description of available parameters can be found here `ogd-smn_meta_parameters.csv`:
- Plot monthly mean global radiation [W/m^2] (parameter `gre000m0`) at the station HAI

Data Sources

- ogd-smn_hai_m_historical.csv
- ogd-smn_hai_m_recent.csv

Sources of problems

- Colum names sometimes contain quotes `"` that need to be removed first
- inconsistend column names between the `_historical` and `_recent` files, i.e. `reference_timestamp` vs. `REFERENCE_TS`

Prerequesites

In order to run this Jupyter notebook, you need the following packages installed on your machine

- Jupyter, quite obvious 😉
- Python
- pandas
- matplotlib
- seaborn

STAC API

Documentation

For documentation please refer to swisstopo's STAC API documentation.

How to check for new data

When downloading data from the STAC API, you might want to make sure to always retrive the most current data.
By default, asset objects are cached for 2 hours. But there might be objects, that are updated more frequently.
We highly recommend to use preconditioning via the `If-Match` or `If-None-Match` headers (mostly the latter one) when making calls to the STAC API.
This reduces unnecessary traffic. When the client sends an `If-None-Match` header containing the `ETag` of the current (local)
version of the requested object, the server compares it to the currently available resource's `ETag` on the server.
Only in case the two values don't match, the requested object is sent.
Otherwise the server responds with a `304 Not Modified` without a body, which tells the user (i.e. the client) that his version of the asset is still good to use.
For more details please check swisstopo's STAC API documentation.

Disclaimer

This notebook is intended to be a brief demonstration, hence the code used here is not optimized for production use.

Open Data automatisch beziehen

```python
# Import necessary libraries
import pandas as pd  # Data processing: https://pandas.pydata.org/docs/
import matplotlib.pyplot as plt  # Visualization: https://matplotlib.org/stable/contents.html
import seaborn as sns  # Also used for visualization: https://seaborn.pydata.org/


# Define file URLs
historical_url = "https://sys-data.int.bgdi.ch/ch.meteoschweiz.ogd-smn/hai/ogd-smn_hai_m_historical.csv"
recent_url = "https://sys-data.int.bgdi.ch/ch.meteoschweiz.ogd-smn/hai/ogd-smn_hai_m_recent.csv"

# define the parameter we want to use
globrad_col = "gre000m0"

# define time range we want to analyse
# for our example we want to compare the months september to december of the years 2023 and 2024
start_year = 2023
end_year = 2024

start_month = 9
end_month = 12


def download_csv(url):
    """
    Downloads a CSV file from a given URL and returns a Pandas DataFrame.
    """
    try:
        # in production, please check if a local copy of the files already exists. If so, please
        # send the ETag of the local resource (that you got in the response when initially
        # requesting the resource) in an If-None-Match header. The server will only send the file,
        # if the remote version is newer than your local file. This avoids unnecessary traffic.
        # (also check here: https://data.geo.admin.ch/api/stac/static/spec/v1/apitransactional.html#tag/Data/operation/getAssetObject)
        # For our short example, we don't need to do 4all this.
        df = pd.read_csv(url, delimiter=';')
        return df
    # too broad exception, I know. Please use better error handling in production ;-)
    except Exception as e:
        print(f"Error downloading {url}: {e}")
        return None
```

Open Data automatisch beziehen

Once we have defined some initial parameters and the function for downloading the csv files from the STAC API, we need to define a function for sanitizing the column names of our pandas dataframes.
This is necessary, since some of the column names of the raw data contain `'`. Additionally, we make sure, that we use the same name for the column which contains the timestamp, since in the raw data files the name varies.

```python
def sanitize_column_names(df):
    """
    Removes single quotes from column names in a given Pandas DataFrame.
    Also rename "REFERENCE_TS" to "reference_timestamp".

    Args:
        df (pd.DataFrame): The input DataFrame.

    Returns:
        pd.DataFrame: A new DataFrame with cleaned column names.
    """
    new_column_names = {col: col.replace("'", "") for col in df.columns}
    df_cleaned = df.rename(columns=new_column_names)

    if "REFERENCE_TS" in df_cleaned.columns:
        df_cleaned = df_cleaned.rename(columns={"REFERENCE_TS": "reference_timestamp"})

    if not globrad_col in df_cleaned.columns:
        print("Error: No valid global radiation column found!")
        return None


    return df_cleaned
```

Now we call the functions we have defined above

```python
df_recent = download_csv(recent_url)
df_historical = download_csv(historical_url)

# sanitize column names
df_recent = sanitize_column_names(df_recent)
df_historical = sanitize_column_names(df_historical)
```

Open Data automatisch beziehen

Now that we downloaded the files, we can process the data. We need to concatenate the two dataframes from the two files we use, the `_historical` and the `_recent` ones.
We also drop potential duplicates and filter the data according to our desired years and months we want to study (defined above).

```python
def process_data(df_recent, df_historical):
    """
    Process the data, i,e, concatenate the two data frames from the _historical and the _recent files.
    Also drop potential duplicates and only return data in the defined time range we want to analyse.

    Args:
        df_recent, df_historical (pd.DataFrame): The input DataFrames.

    Returns:
        pd.DataFrame: A new DataFrame with processed data.
    """


    # concatenate the two data frames of the historical and the recent data into one single data frame
    df = pd.concat([df_historical[['reference_timestamp', globrad_col]], df_recent[['reference_timestamp', globrad_col]]], ignore_index=True)
    # remove duplicates (there should be no duplicates, but just in case...)
    df = df.drop_duplicates(ignore_index=True)

    df["timestamp"] = pd.to_datetime(df["reference_timestamp"], format="%d.%m.%Y %H:%M", errors="raise")
    df["year"] = df["timestamp"].dt.year
    df["month"] = df["timestamp"].dt.month
    df["month_str"] = df["timestamp"].dt.strftime("%B")


    df = df[df["month"].between(start_month, end_month) & df["year"].between(start_year, end_year)]


    return df


df = process_data(df_recent, df_historical)
```

[62]

+ Code + Markdown

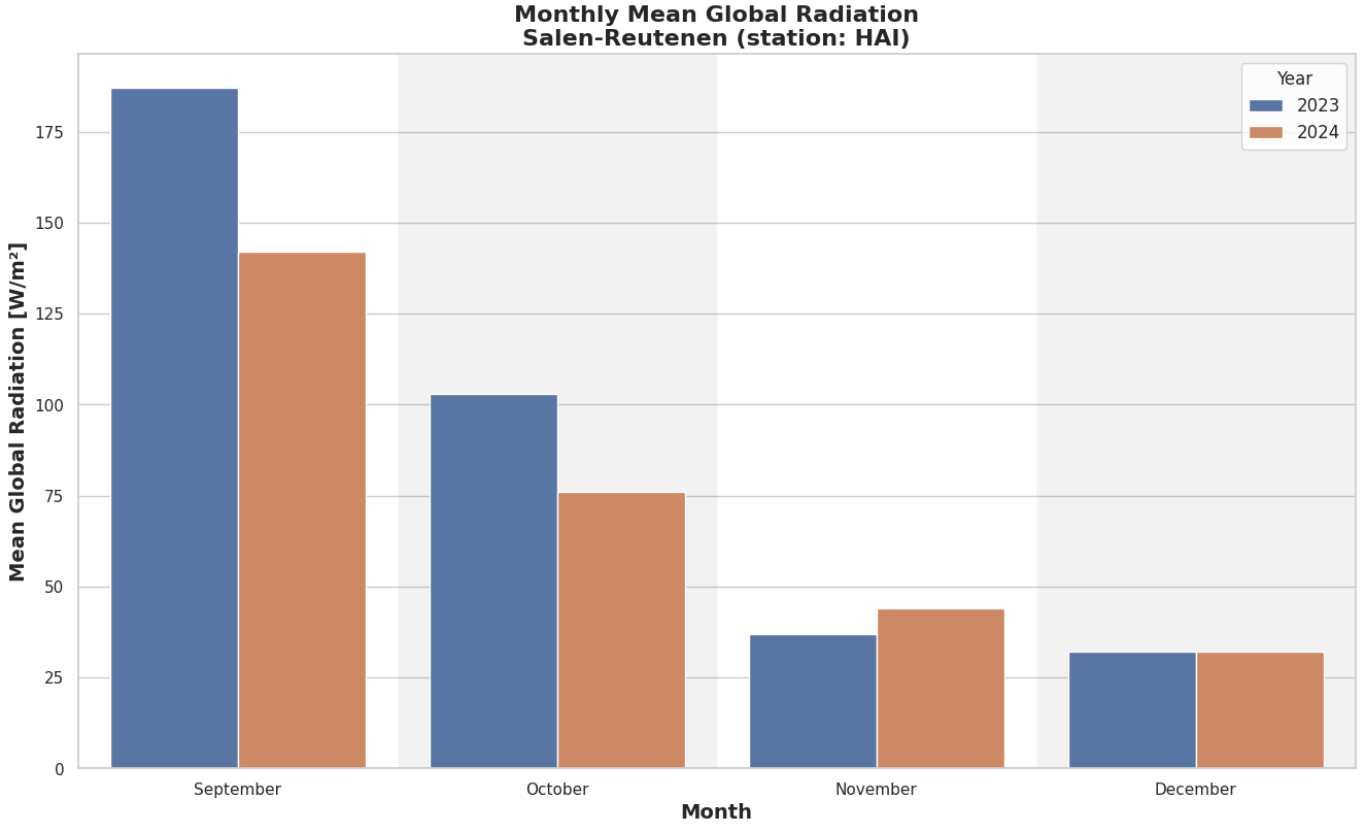
Open Data automatisch beziehen

Visualization

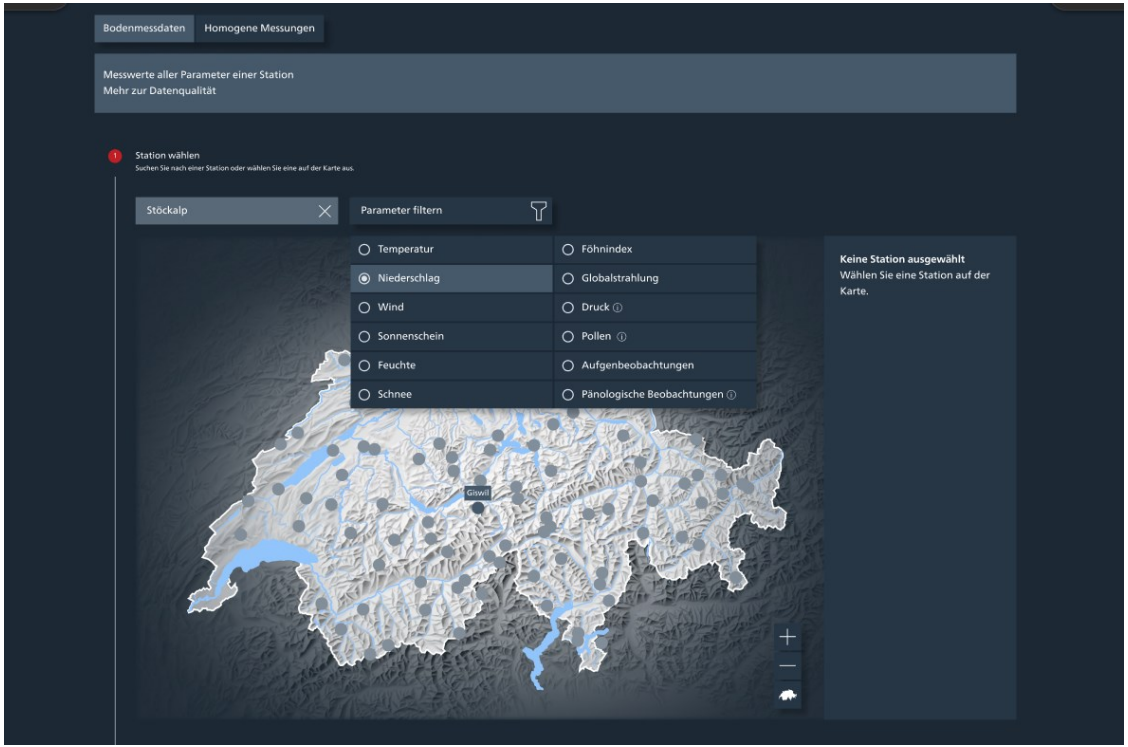
Now wa are ready to plot the monthly mean global radiation from Sept - Dec for 2023 and 2024

```python
def create_plot(df):
    """
    Plots global radiation monthly means
    """

    sns.set_theme(style="whitegrid")
    color_palette = sns.color_palette("deep", len(df["year"].unique()))

    plt.figure(figsize=(16, 9))
    ax = plt.gca()

    # Shading every 2nd Month for better readability.
    # Might need to be adapted for incomplete timeseries, or years with gaps, but works nice for our example
    for i in range(1, len(df["month_str"].unique()), 2):
        ax.axvspan(i - 0.5, i + 0.5, facecolor='gray', alpha=0.1)

    sns.barplot(data=df, x="month_str", y="gre000m0", hue="year", dodge=True, ax=ax, palette=color_palette)
    ax.set_xlabel("Month", fontsize=14, fontweight='bold')
    ax.set_ylabel("Mean Global Radiation [W/m²]", fontsize=14, fontweight='bold')
    ax.set_title("Monthly Mean Global Radiation\nSalen-Reutenen (station: HAI)", fontsize=16, fontweight='bold')

    ax.legend(title="Year", fontsize=12)

    plt.show()

create_plot(df)
```

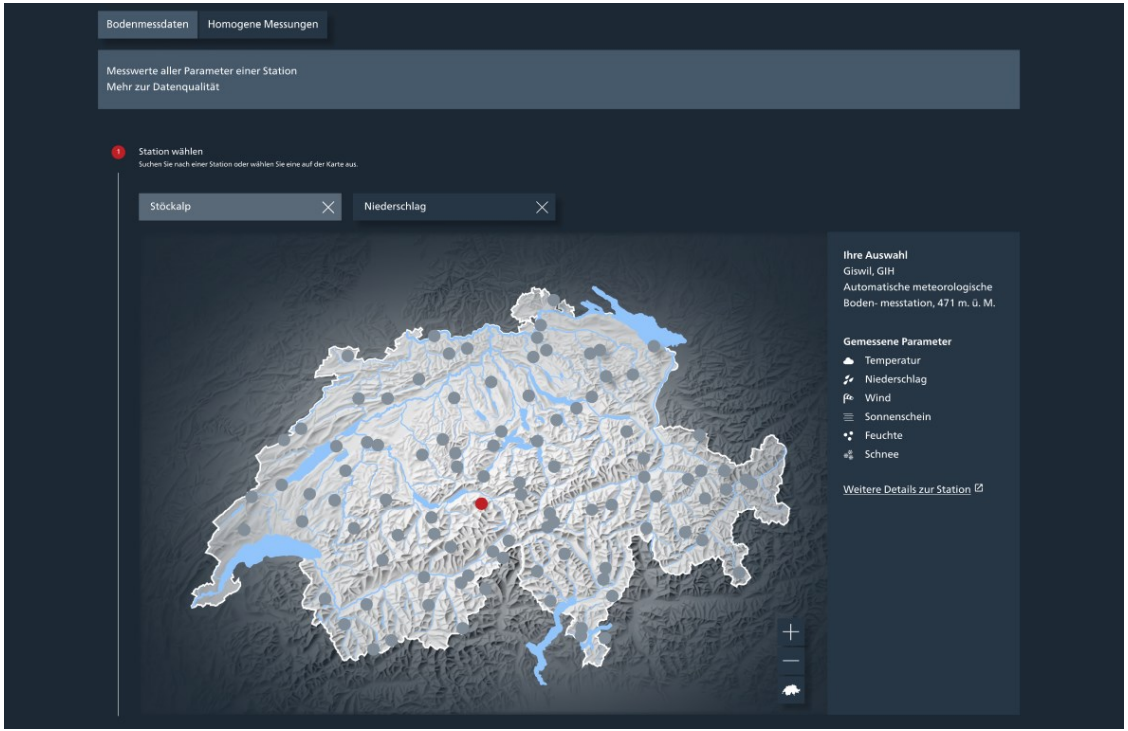
Open Data automatisch beziehen



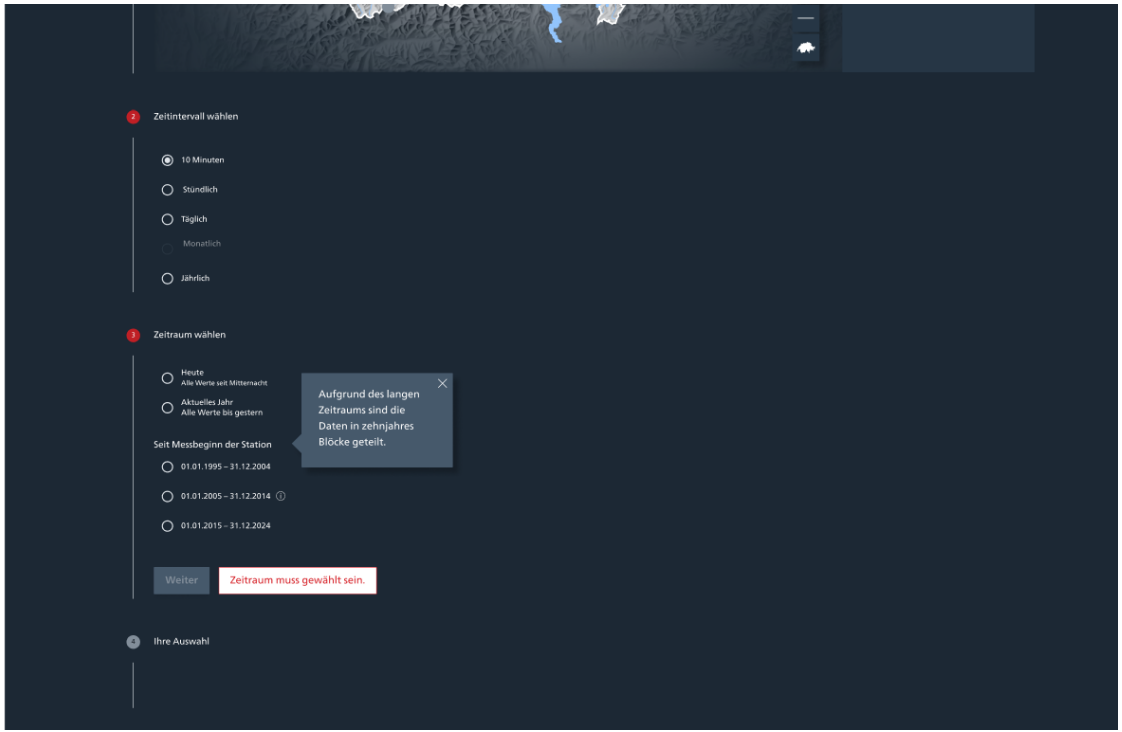
Open Data manuell herunterladen



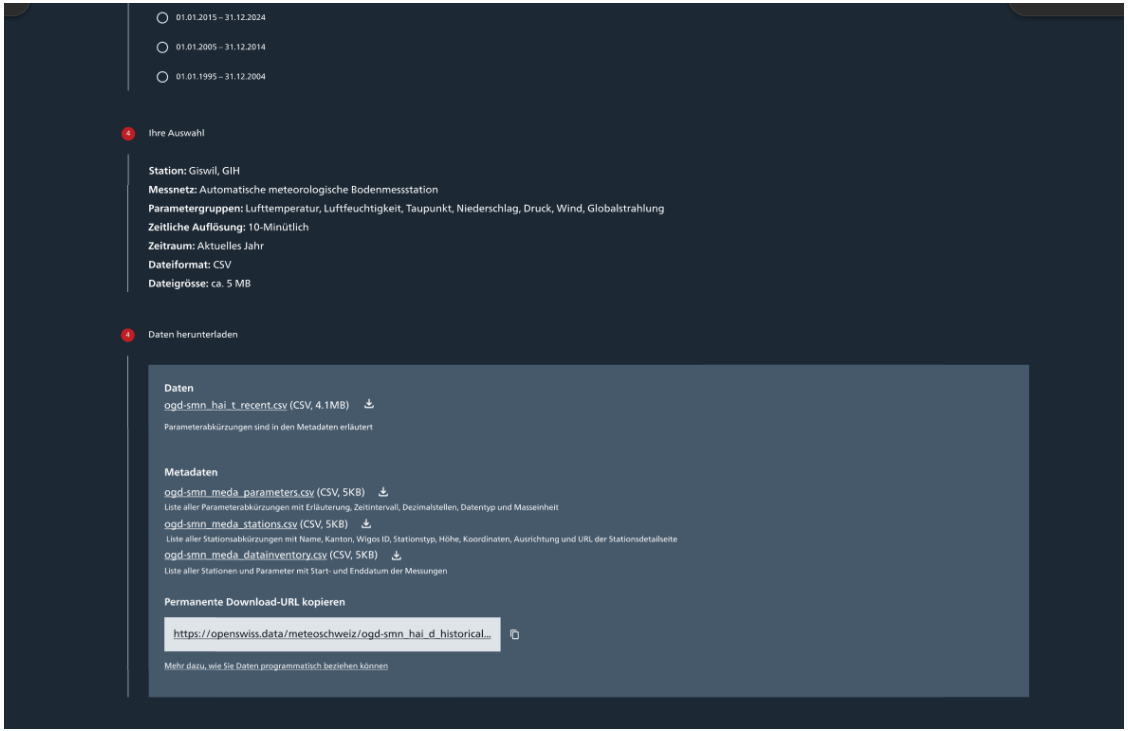
Open Data manuell herunterladen



Open Data manuell herunterladen



Open Data manuell herunterladen




Einführung Vorhaben
Neue Datenprodukte
Life of a Meteo Dataset
Technische Aspekte
Datenprodukte nutzen
Ausblick
API

Umsetzung OGD MeteoSchweiz: Ausblick

etappiert

EMBAG/MetG
Inkrafttreten ✓

β

1. Etappe
MetV = OGD MCH 1.4.25
STAC API

2. Etappe
API Features + EDR

EMBAG
Deadline Einführung
OGD Bund

1.1.24 | 24.3.25 | 1. Wochen Mai | 2026 | 31.12.2026

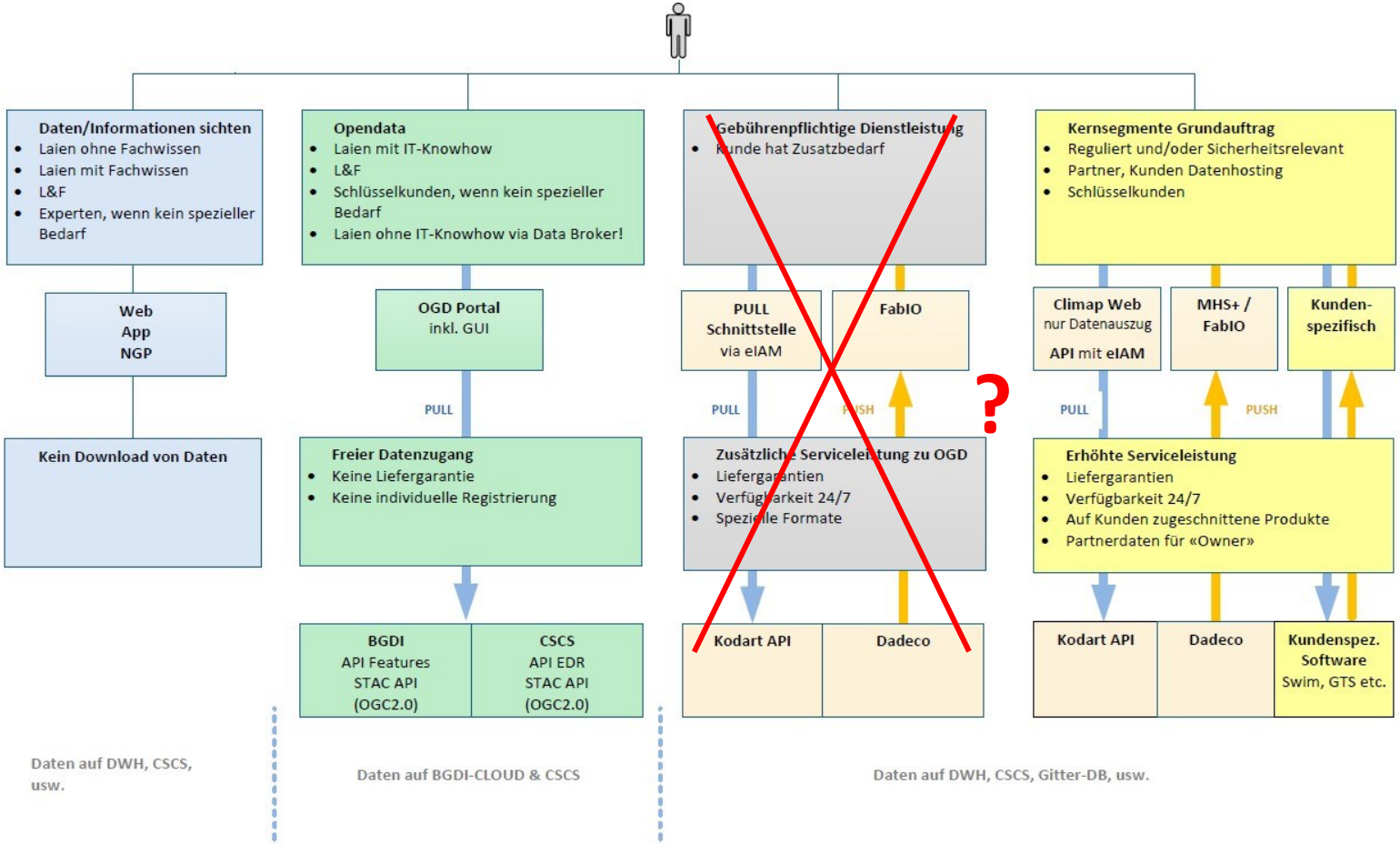
Go-Live-Kriterien (Punktdatensatz, Daten Browser, Prozesse)

Zusammen mit
- Swisstopo
- CSCS

----- Übergangsphase Push → Pull ----------

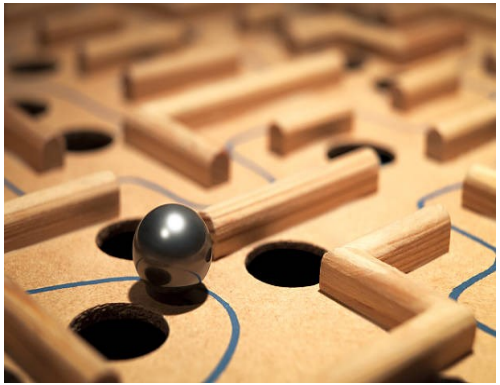
$ (Cloud)

Datendistribution extern «Einführungsphase OGD»



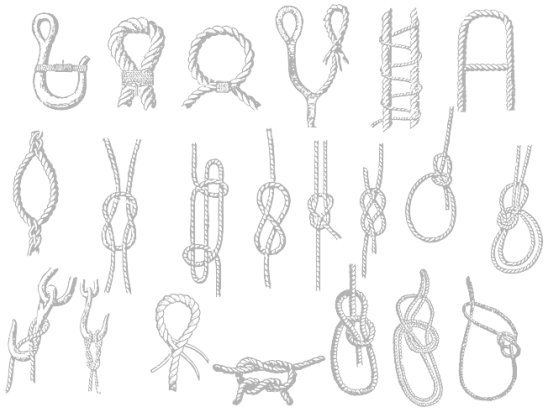
dwo / 19.03.2025

Herausforderungen Vorhaben



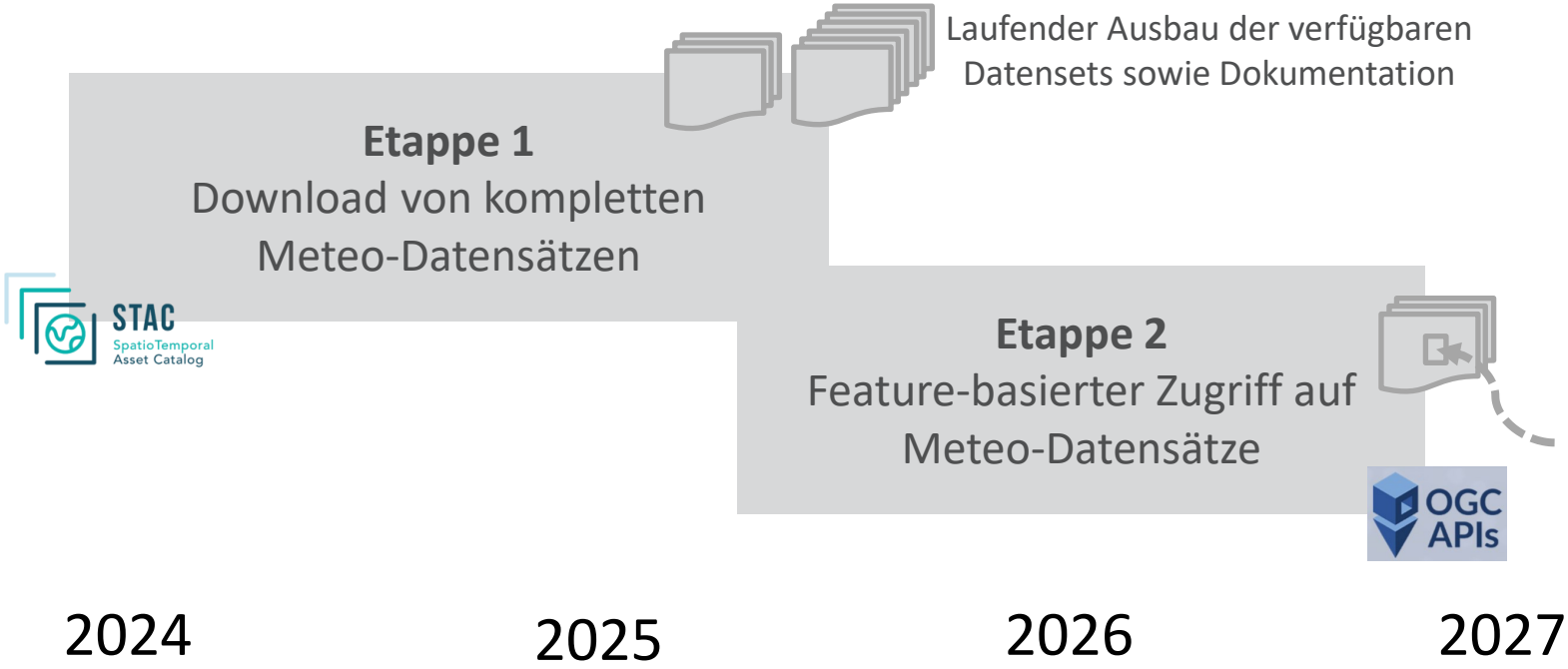
- Finanzierung
- Interne Ressourcen zur Datenbereitstellung, -integration sowie -publikation
- Priorisierung des Vorhabens
- Kollaborationstools

Herausforderungen Technisch/Betrieb

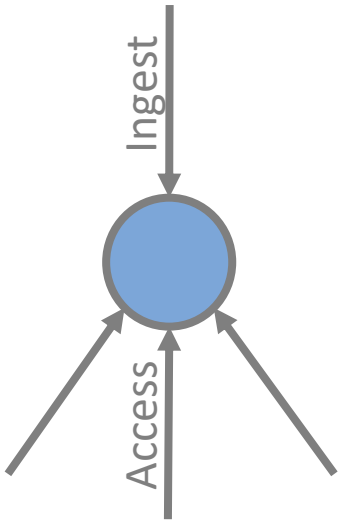


- Datenmengen und Updatefrequenz
- Betriebsbudget vs. Downloadmenge
- Internationale Meteo-Anforderungen vs. Betriebsumfeld BGDI

Gemeinsames Vorgehen in Etappen



Etappe 2: Ausbau PP BGDI



Ingest API

Etappierte Einführung einer generischen Schnittstelle für die standardisierte Datenanlieferung

OGC API – Features

Etappenweise Einführung


Kontakt
opendata.de@meteoschweiz.ch
webgis@swisstopo.ch