Grafana Dashboard for Phenological Data Monitoring

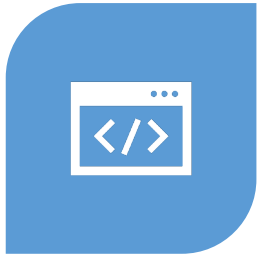
Aleksei Ivanov

Outline

CURRENT DEVELOPMENTS

APPLICATION DESIGN

METHODOLOGY



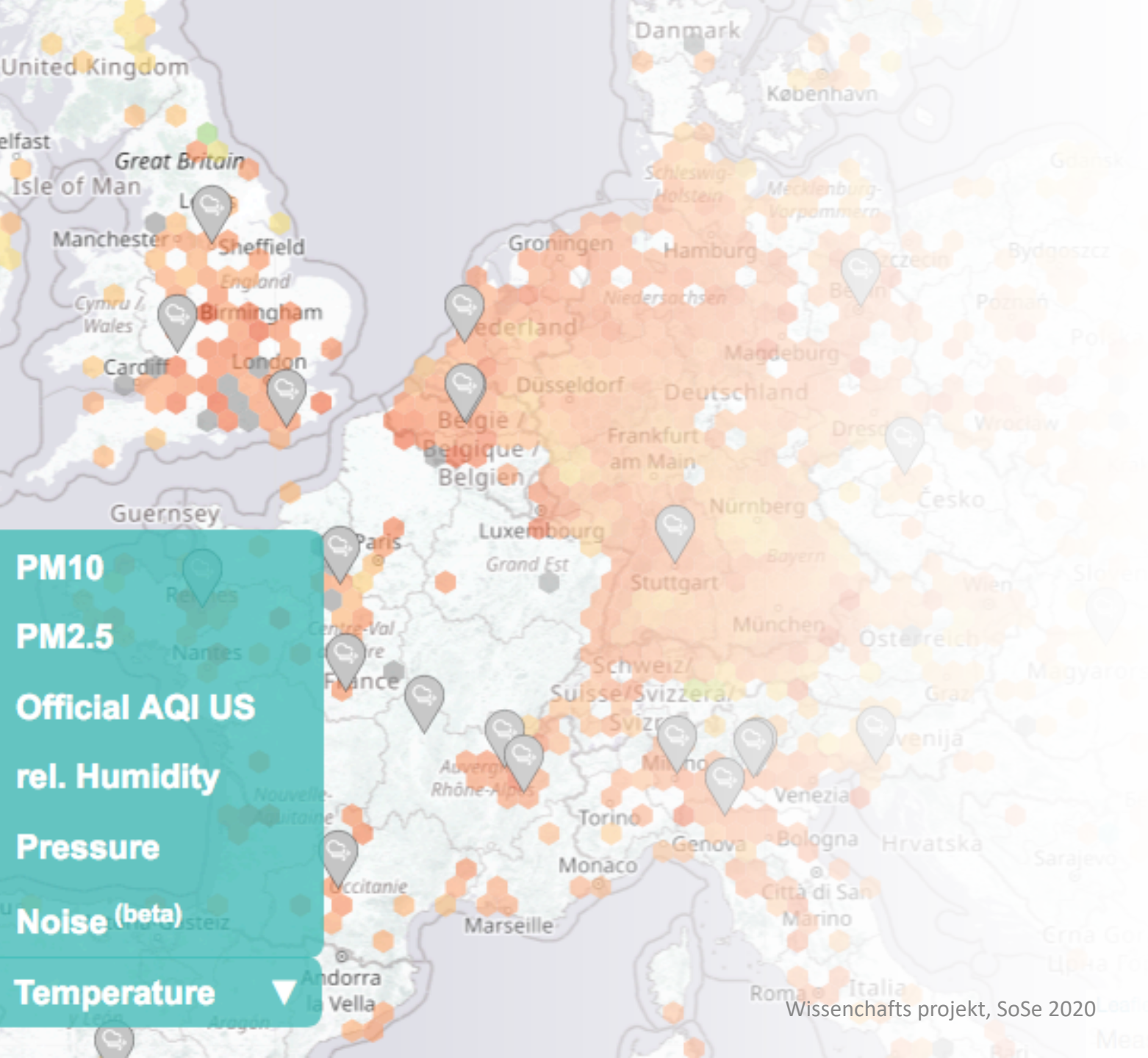
IMPLEMENTATION

DEMONSTRATION

Current developments

Open data sources

State of the art visualization tools

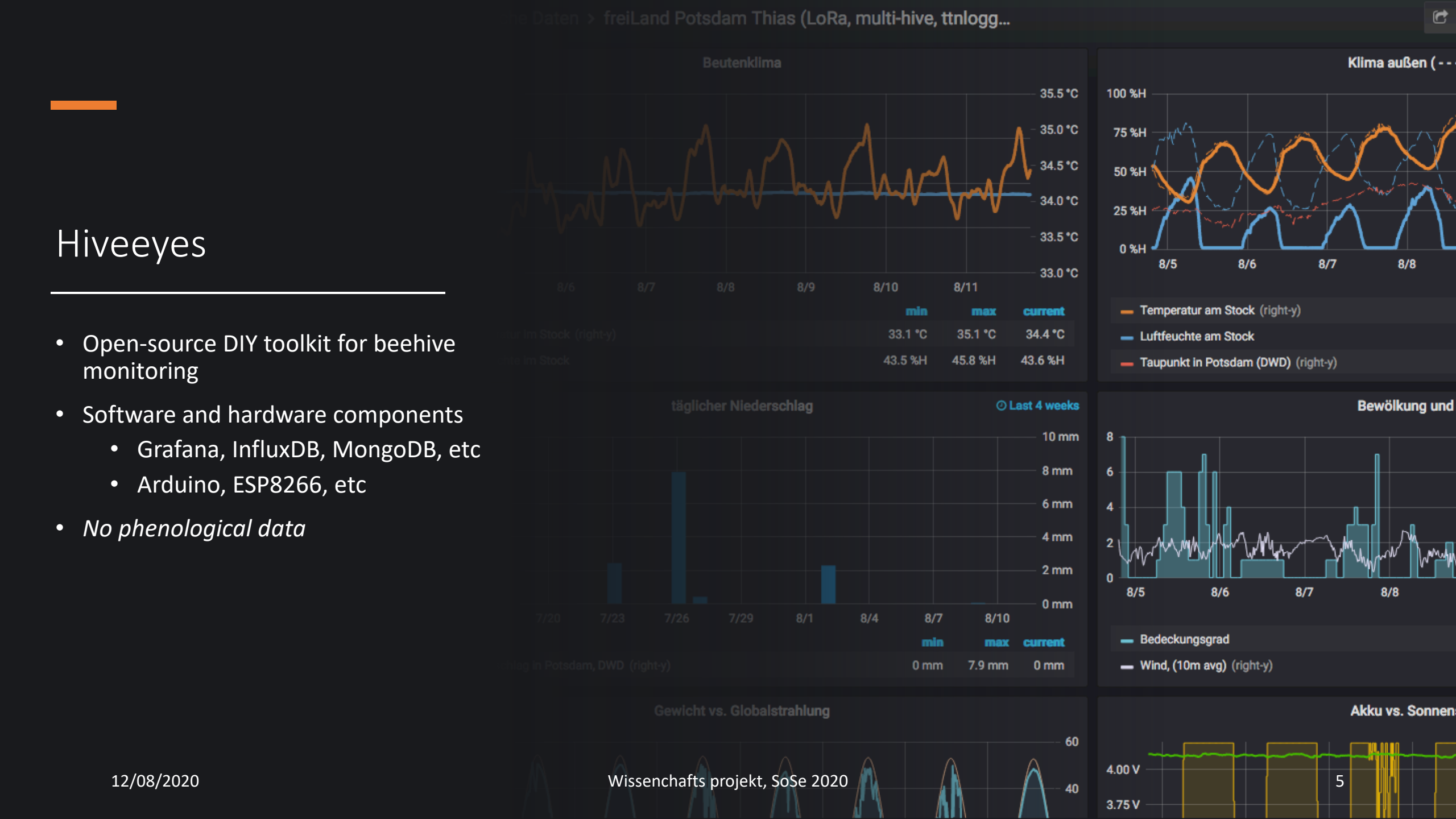


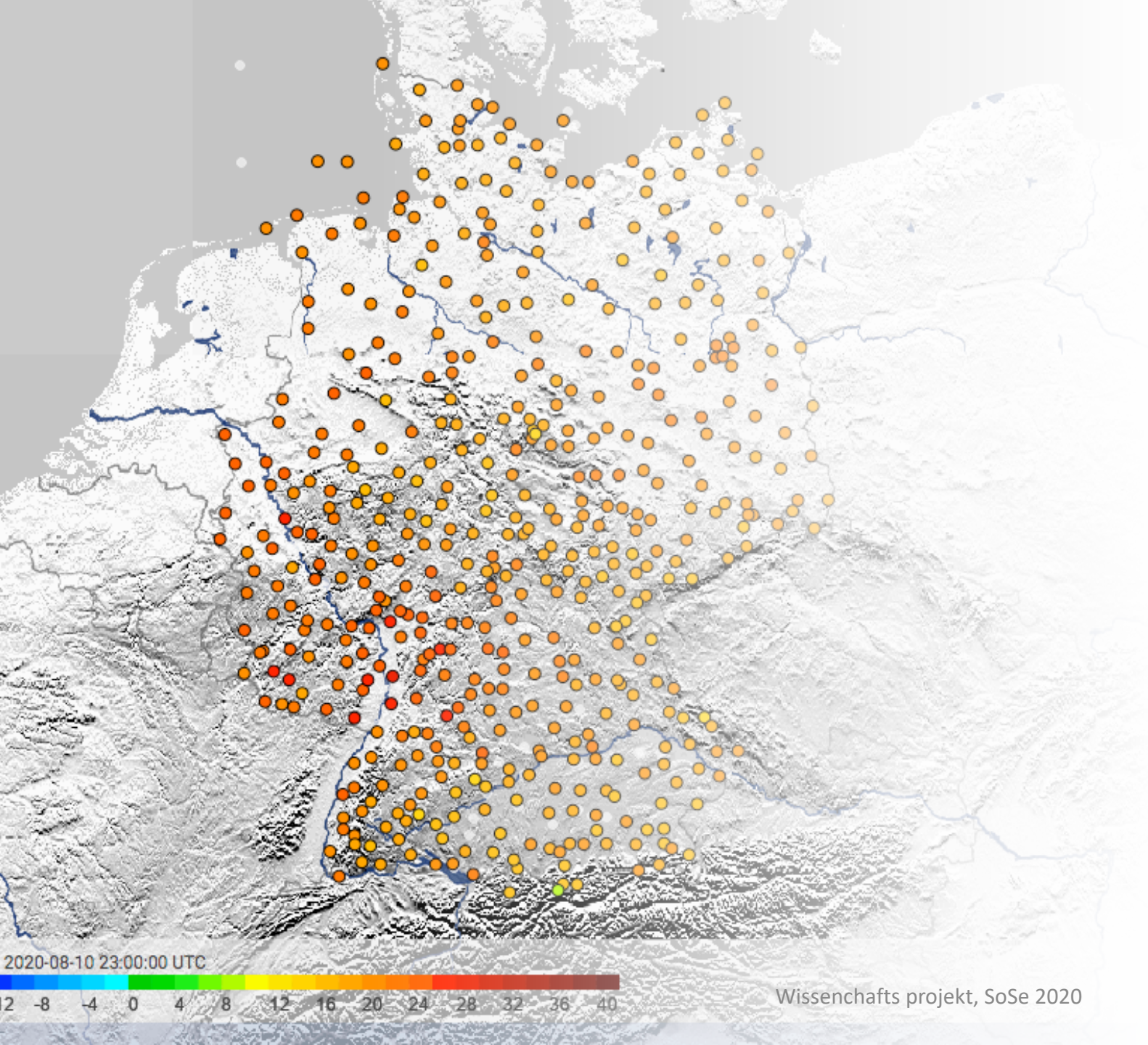
Sensor.Community

- Open Environmental Data
  - API and FTP archive
- Map visualization (no dashboard)
- *No phenological data*

Hiveeyes

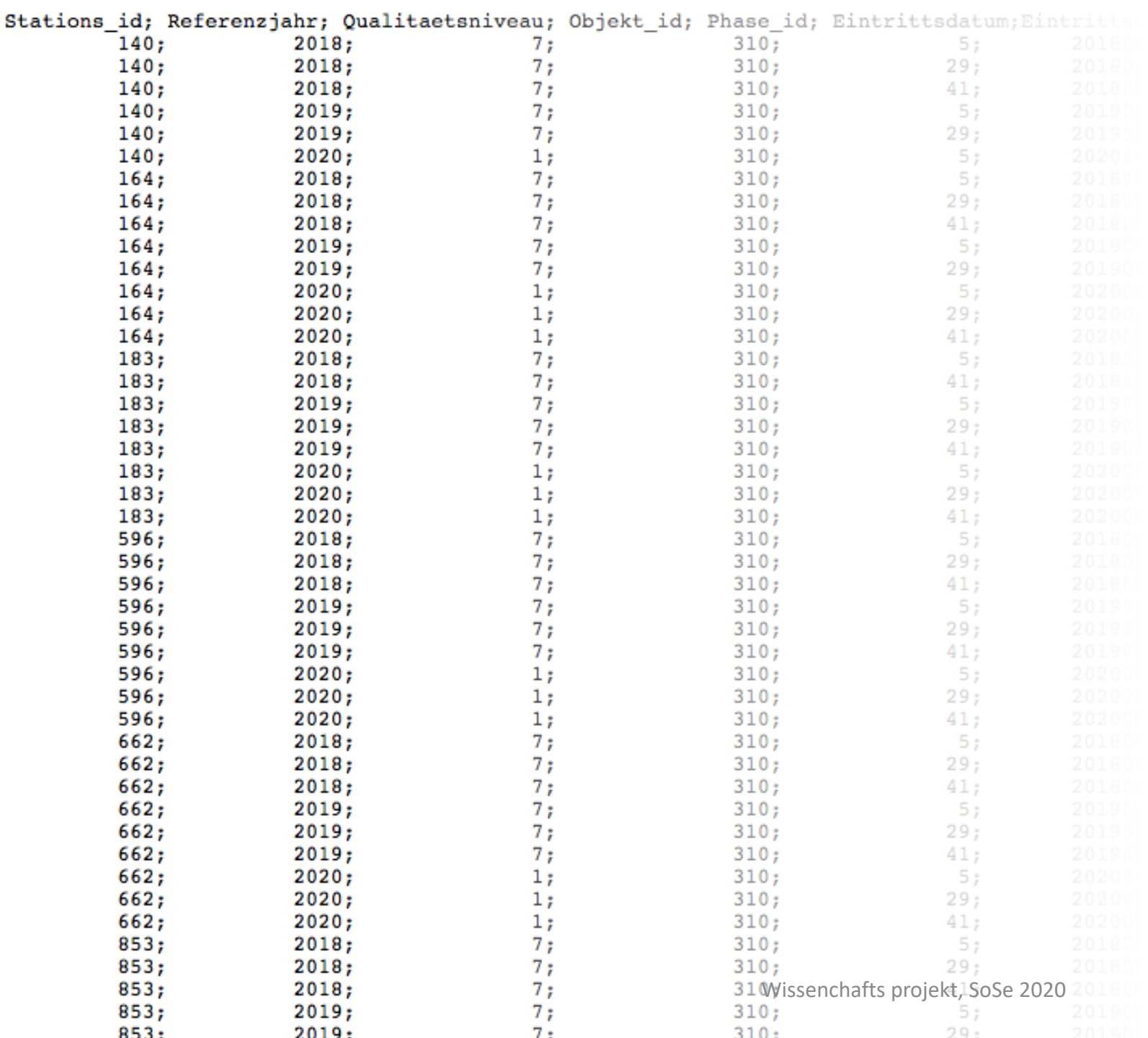
- Open-source DIY toolkit for beehive monitoring
- Software and hardware components
  - Grafana, InfluxDB, MongoDB, etc
  - Arduino, ESP8266, etc
- *No phenological data*





Deutscher Wetterdienst (DWD)

- Country-wide ecological data monitoring
- Climate Data Center (CDC)
  - Open data repository
  - Data gathered from stations
  - Accessible over HTTP/FTP
- Map visualization not available for all data in the repository



```
Stations_id; Referenzjahr; Qualitaetsniveau; Objekt_id; Phase_id; Eintrittsdatum;Eintritts
         140;          2018;               7;                310;               5;       2018
         140;          2018;               7;                310;              29;       2018
         140;          2018;               7;                310;              41;       2018
         140;          2019;               7;                310;               5;       2019
         140;          2019;               7;                310;              29;       2019
         140;          2020;               1;                310;               5;       2020
         164;          2018;               7;                310;               5;       2018
         164;          2018;               7;                310;              29;       2018
         164;          2018;               7;                310;              41;       2018
         164;          2019;               7;                310;               5;       2019
         164;          2019;               7;                310;              29;       2019
         164;          2020;               1;                310;               5;       2020
         164;          2020;               1;                310;              29;       2020
         164;          2020;               1;                310;              41;       2020
         183;          2018;               7;                310;               5;       2018
         183;          2018;               7;                310;              41;       2018
         183;          2019;               7;                310;               5;       2019
         183;          2019;               7;                310;              29;       2019
         183;          2019;               7;                310;              41;       2019
         183;          2020;               1;                310;               5;       2020
         183;          2020;               1;                310;              29;       2020
         183;          2020;               1;                310;              41;       2020
         596;          2018;               7;                310;               5;       2018
         596;          2018;               7;                310;              29;       2018
         596;          2018;               7;                310;              41;       2018
         596;          2019;               7;                310;               5;       2019
         596;          2019;               7;                310;              29;       2019
         596;          2019;               7;                310;              41;       2019
         596;          2020;               1;                310;               5;       2020
         596;          2020;               1;                310;              29;       2020
         596;          2020;               1;                310;              41;       2020
         662;          2018;               7;                310;               5;       2018
         662;          2018;               7;                310;              29;       2018
         662;          2018;               7;                310;              41;       2018
         662;          2019;               7;                310;               5;       2019
         662;          2019;               7;                310;              29;       2019
         662;          2019;               7;                310;              41;       2019
         662;          2020;               1;                310;               5;       2020
         662;          2020;               1;                310;              29;       2020
         662;          2020;               1;                310;              41;       2020
         853;          2018;               7;                310;               5;       2018
         853;          2018;               7;                310;              29;       2018
         853;          2018;               7;                310;              41;       2018
         853;          2019;               7;                310;               5;       2019
         853;          2019;               7;                310;              29;       2019
```

Deutscher Wetterdienst (DWD)

- Climate Data Center (CDC)
  - Open data repository
  - Data gathered from stations
  - Accessible over HTTP/FTP
- Map visualization not available for all data in the repository
- **Phenological data available**
  - Not visualized on the map
  - Accessible as plain text CSV
  - Text description of plant phases for every station
- *Data source for this project*

State of the art dashboard tools

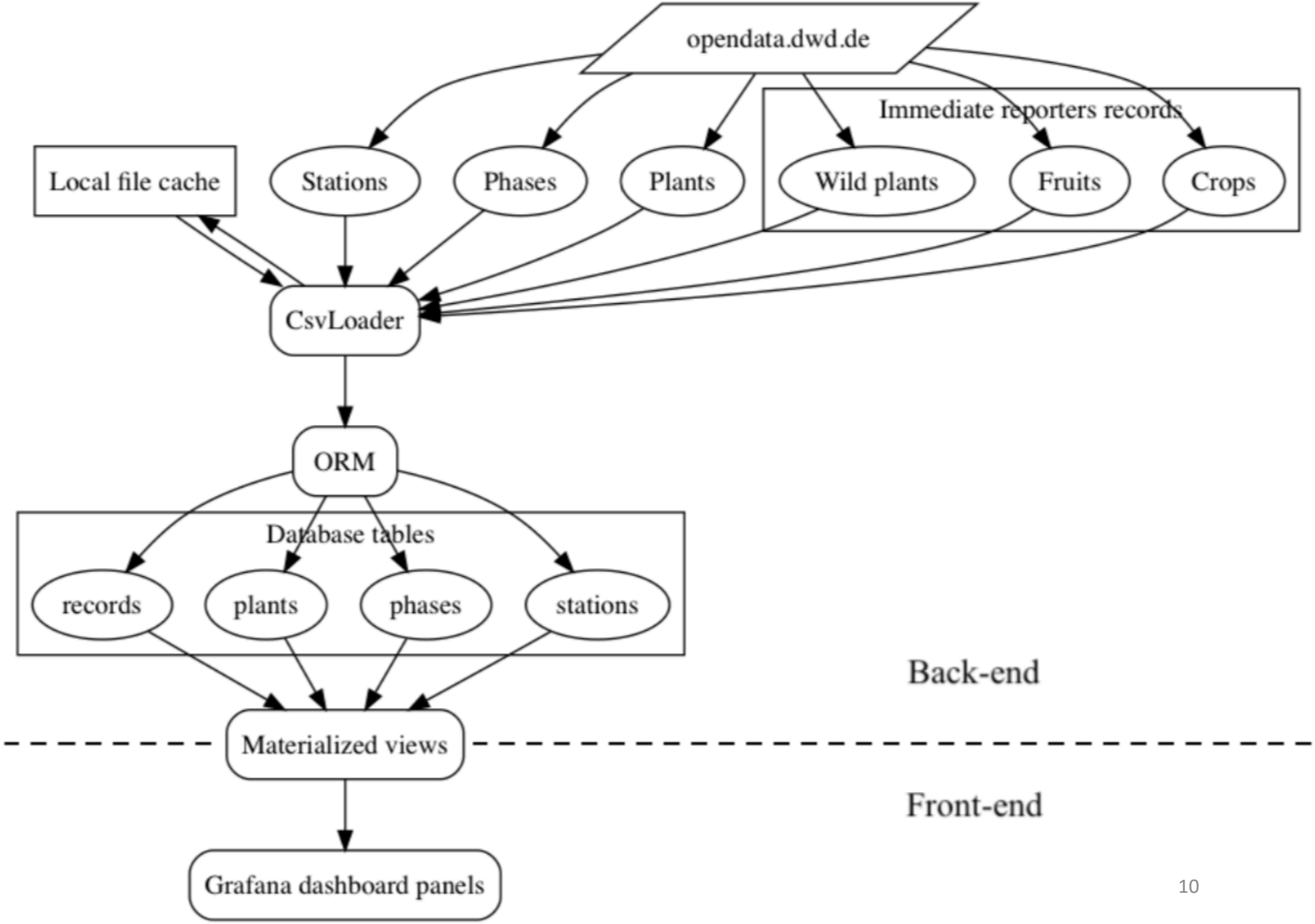
- Time-series data visualization
- InfluxDB
  - Originally built as time-series database
  - Fewer visualization options
  - Fewer supported data sources
- Grafana (was chosen)
  - Open-source
  - Many visualization choices
  - Custom plugins (e.g. world map panel)
  - Connection with many data sources

Application design

Data flow

Architecture

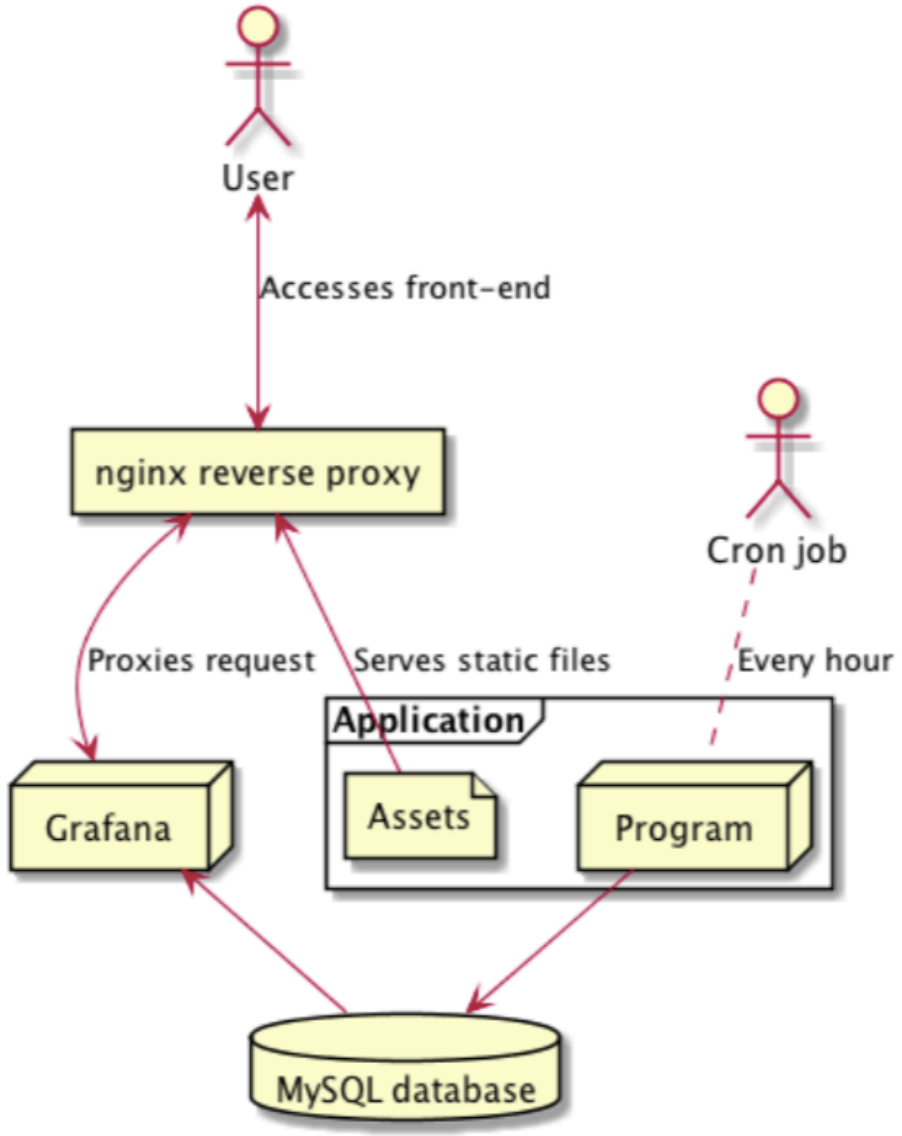
Data Flow


opendata.dwd.de
Immediate reporters records
Local file cache
Stations
Phases
Plants
Wild plants
Fruits
Crops
CsvLoader
ORM
Database tables
records
plants
phases
stations
Back-end
Materialized views
Front-end
Grafana dashboard panels

Architecture

Four core components:

1. MySQL database server;
2. Grafana web application;
3. Web server to host application assets (i.e. images for the use in the dashboard);
4. Software for phenological data download, processing, and upload to the database.

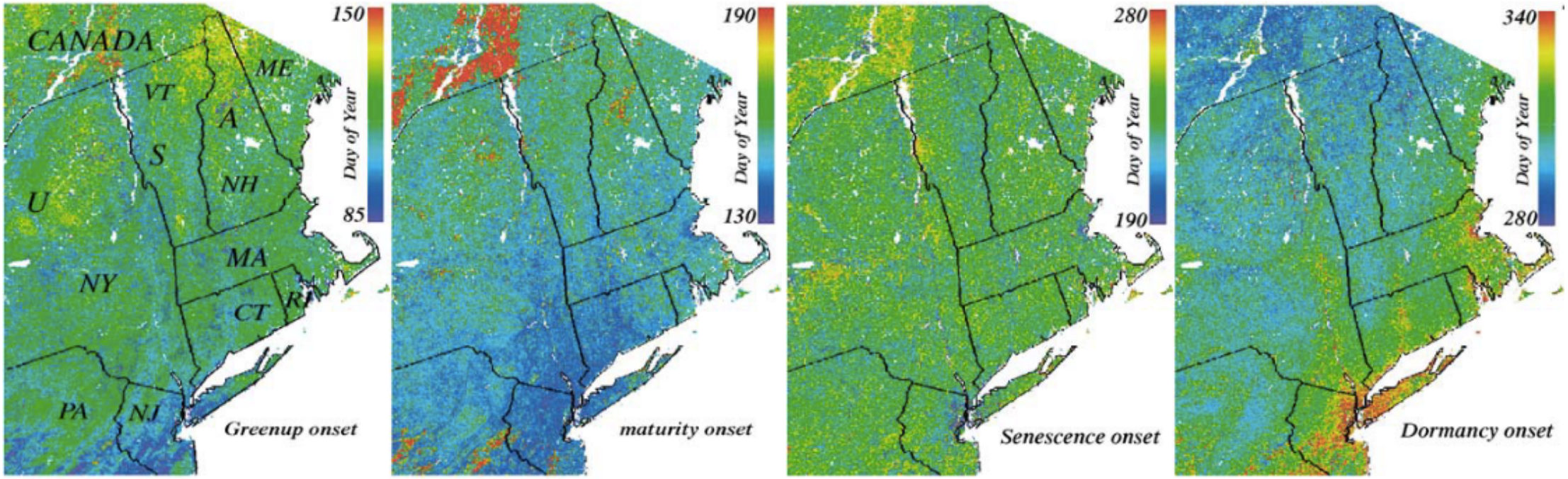


Methodology

Formalization

Data processing

Phase formalization

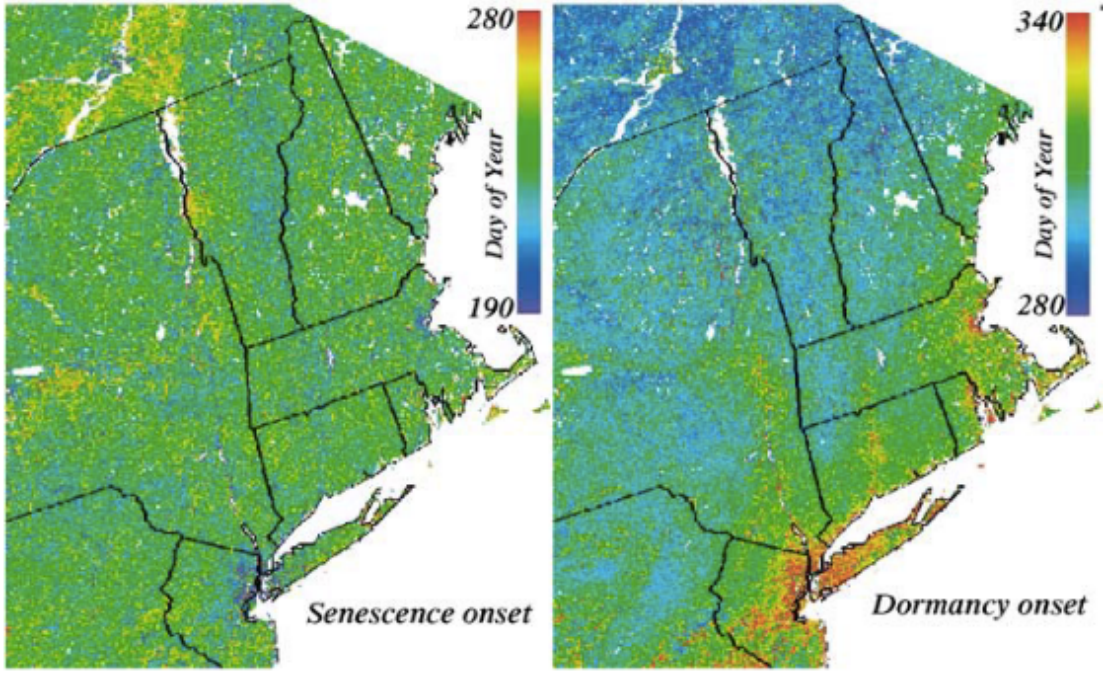


Location vs day of the year for a given phase. Source: [Zha03]

Phase formalization

Such approach:

- Gives more insight
- ***Harder to interpret***
- Three dimensions:
  1. Location
  2. Phase
  3. Value



Location vs day of the year for a given phase. Source: [Zha03]

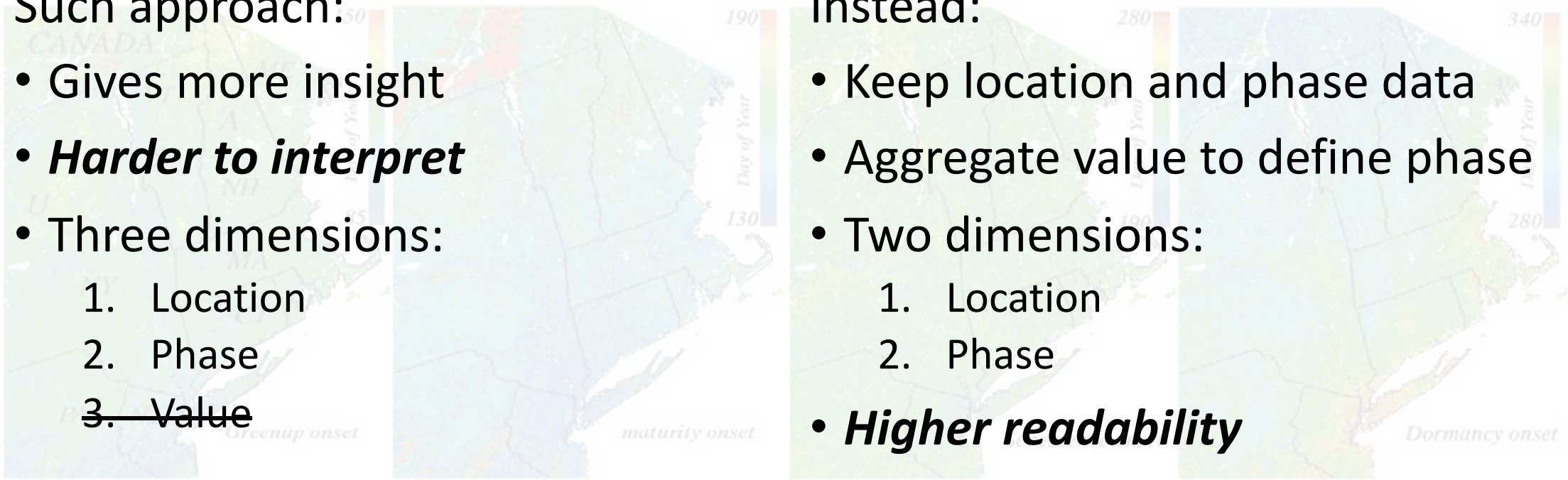
Phase formalization

Such approach:

- Gives more insight
- ***Harder to interpret***
- Three dimensions:
  1. Location
  2. Phase
  3. ~~Value~~

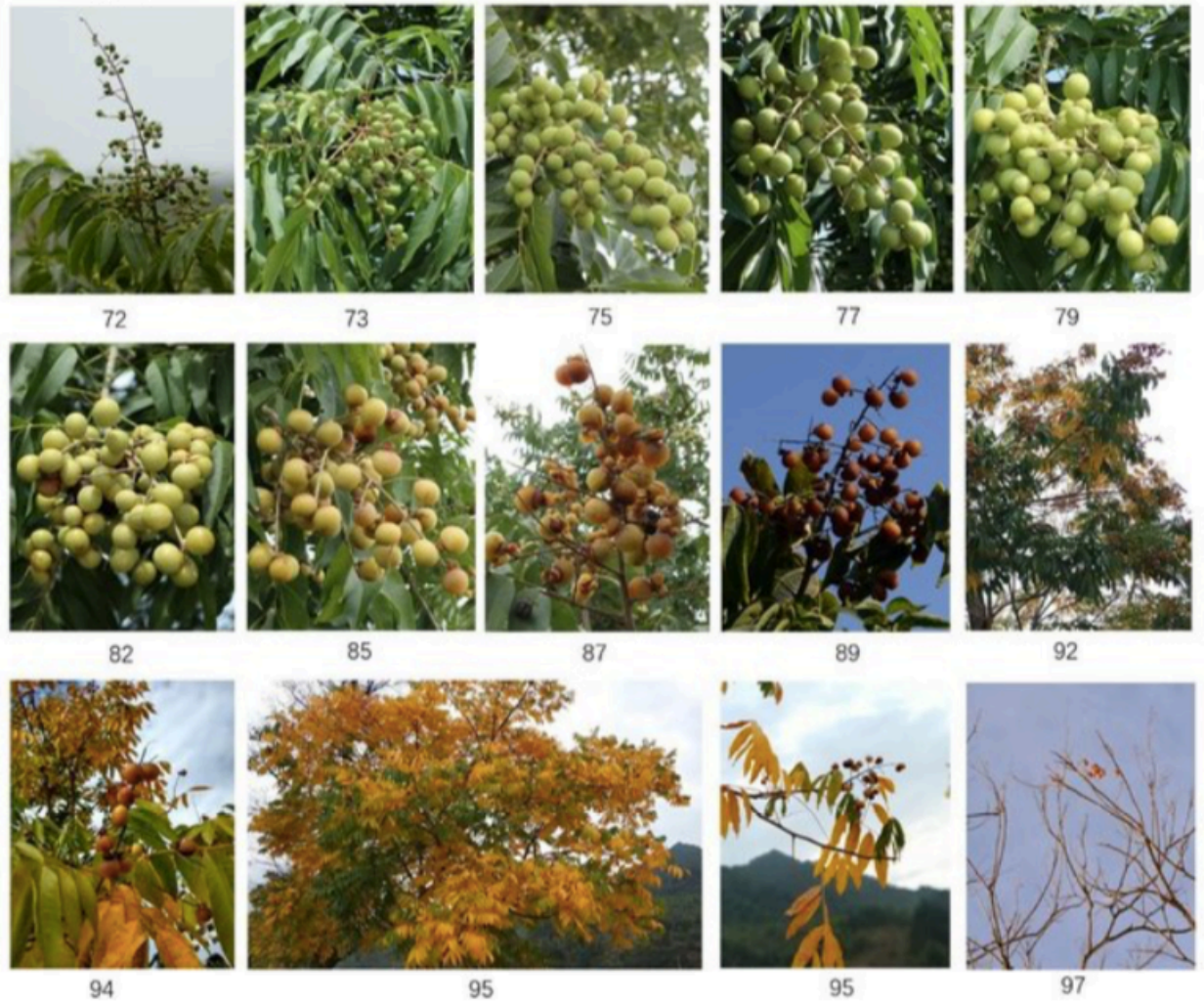
Instead:

- Keep location and phase data
- Aggregate value to define phase
- Two dimensions:
  1. Location
  2. Phase
- ***Higher readability***

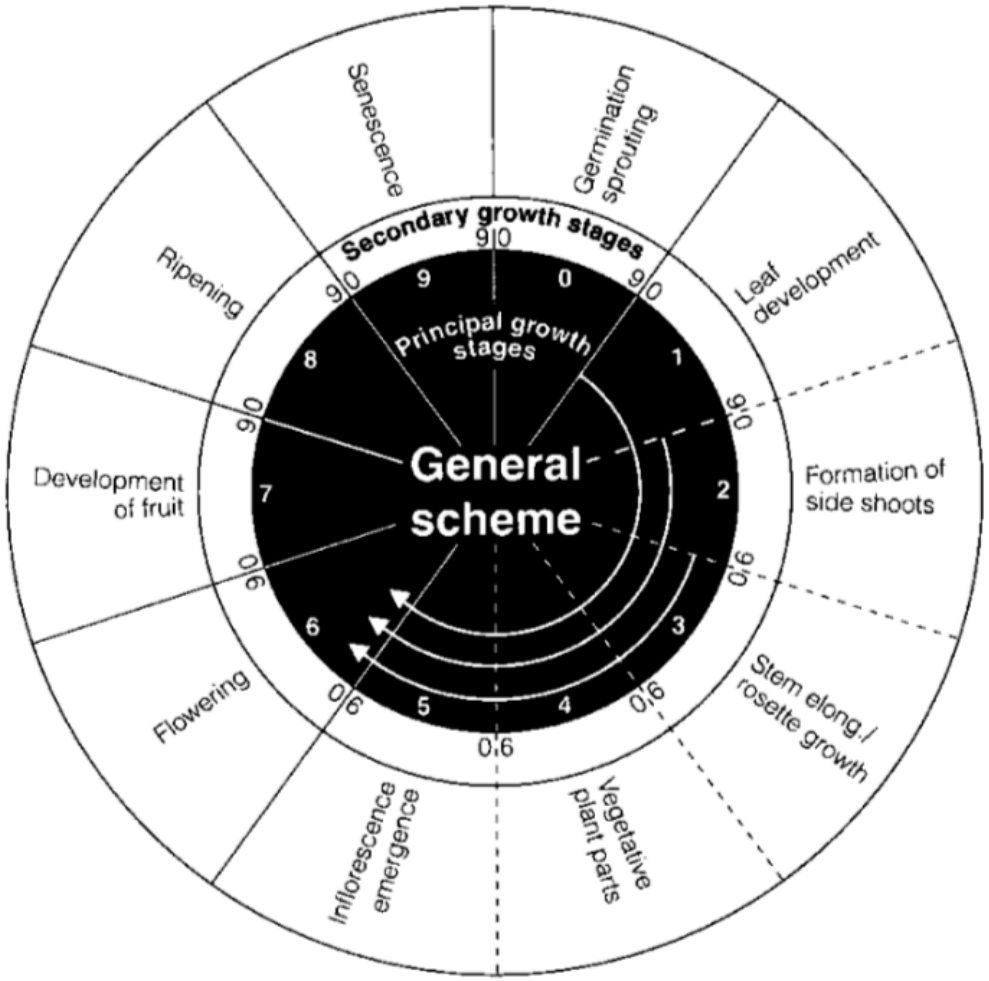


Location vs day of the year for a given phase. Source: [Zha03]

Phase formalization – BBCH scale



Source: [Yan19]

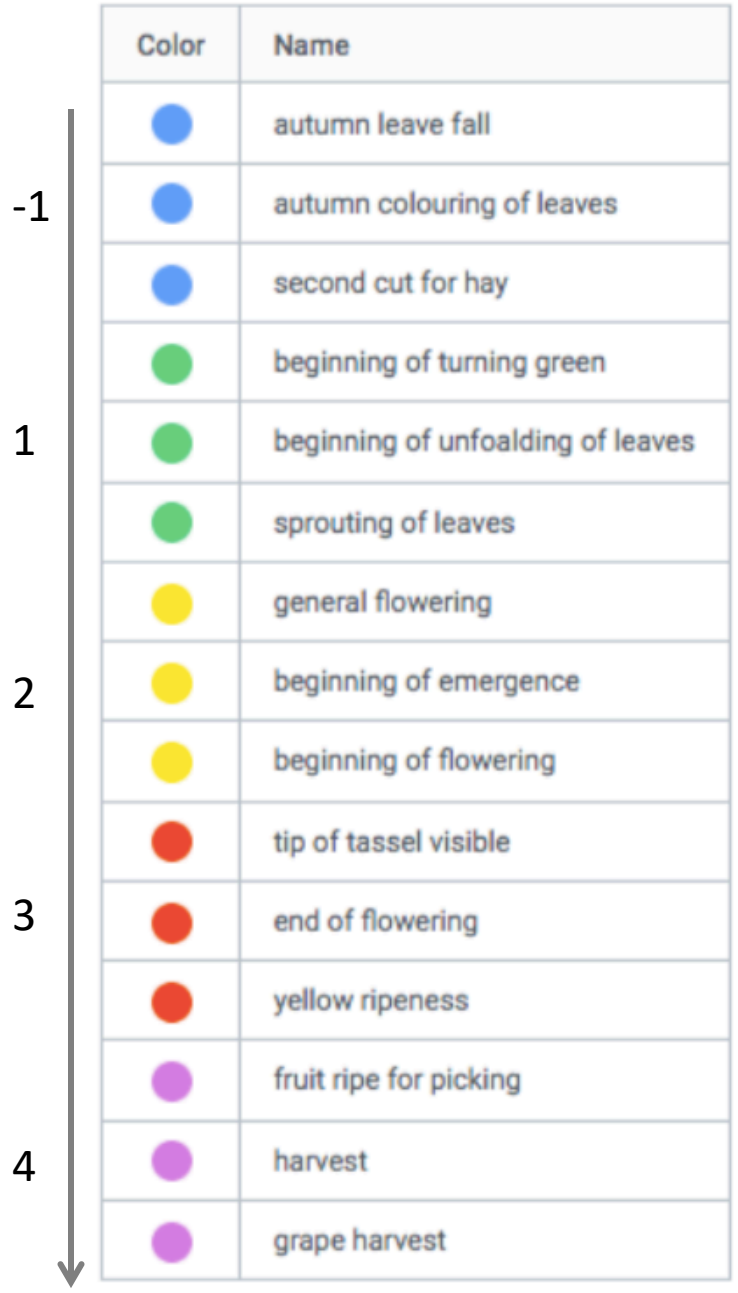


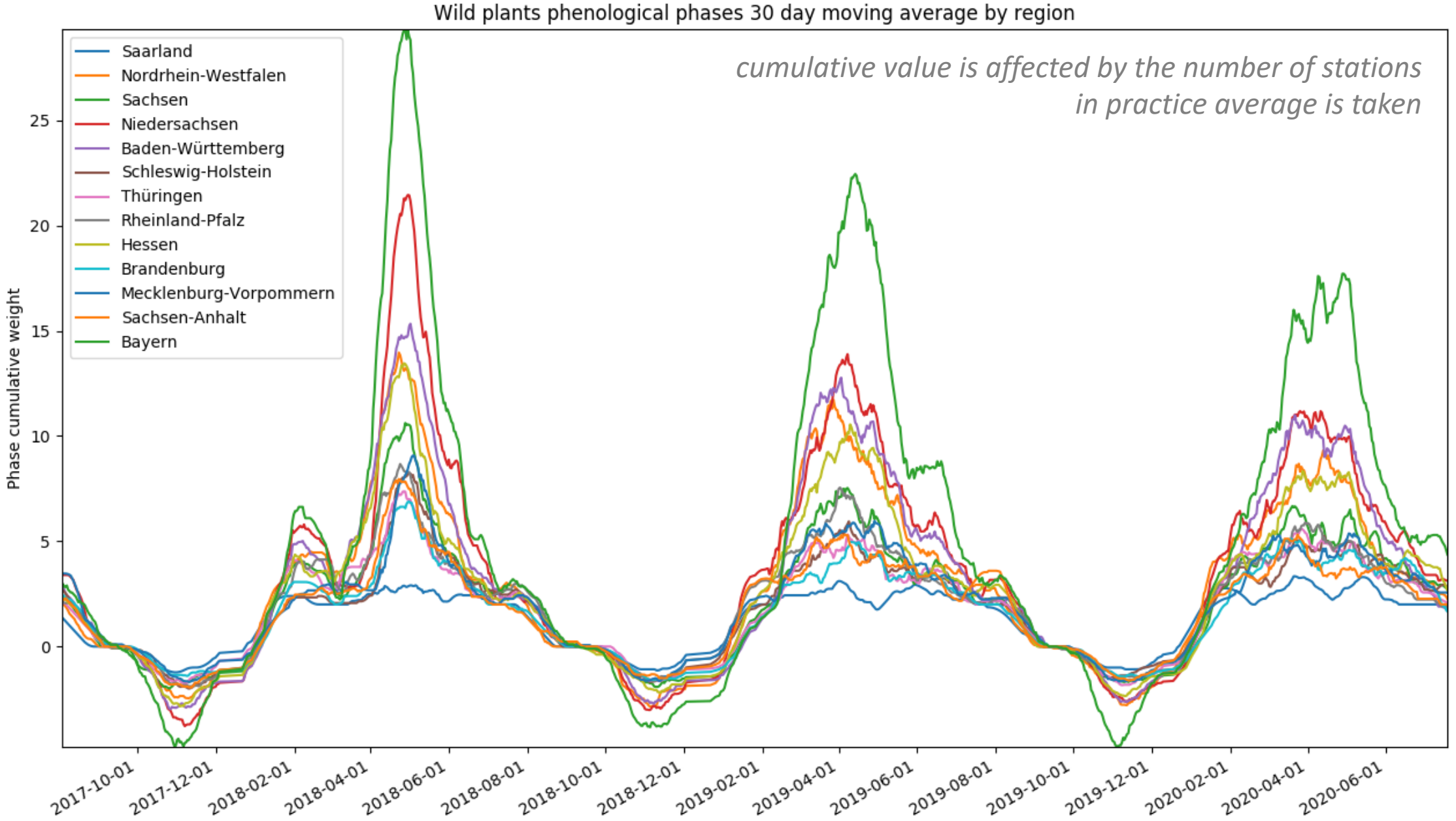
Source: [BBC01]

Phase formalization – weights

Weight	Phase ID	Name
1	1	beginning of turning green
2	5	beginning of flowering
3	7	end of flowering
4	24	harvest
-1	31	autumn coloring of leaves

Tab. 1: Phase weights depending on their description




Wild plants phenological phases 30 day moving average by region
cumulative value is affected by the number of stations
in practice average is taken
Phase cumulative weight
Saarland
Nordrhein-Westfalen
Sachsen
Niedersachsen
Baden-Württemberg
Schleswig-Holstein
Thüringen
Rheinland-Pfalz
Hessen
Brandenburg
Mecklenburg-Vorpommern
Sachsen-Anhalt
Bayern
25
20
15
10
5
0
2017-10-01
2017-12-01
2018-02-01
2018-04-01
2018-06-01
2018-08-01
2018-10-01
2018-12-01
2019-02-01
2019-04-01
2019-06-01
2019-08-01
2019-10-01
2019-12-01
2020-02-01
2020-04-01
2020-06-01

Data processing

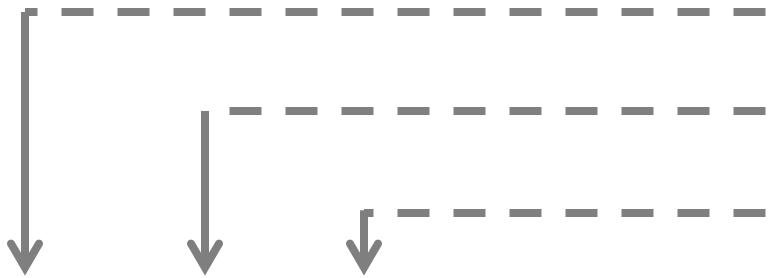
Conversion

Normalization

Data processing

- *Several steps are taken:*
    1. Convert non-standard CSV format into a table representation inside the program;
    2. Normalize column name (language and letter case);
    3. Strip whitespaces;
    4. Extract and convert to application data types from text

Data processing



```
"field_map": {
  "stations_id": "station_id",
  "referenzjahr": "year",
  "qualitaetsniveau": "data_quality_bit",
  "objekt_id": "object_id",
  "phase_id": "phase_id",
  "eintrittsdatum": "date",
  "eintrittsdatum_qb": "date_quality_bit",
  "jultag": "day_of_year"
```

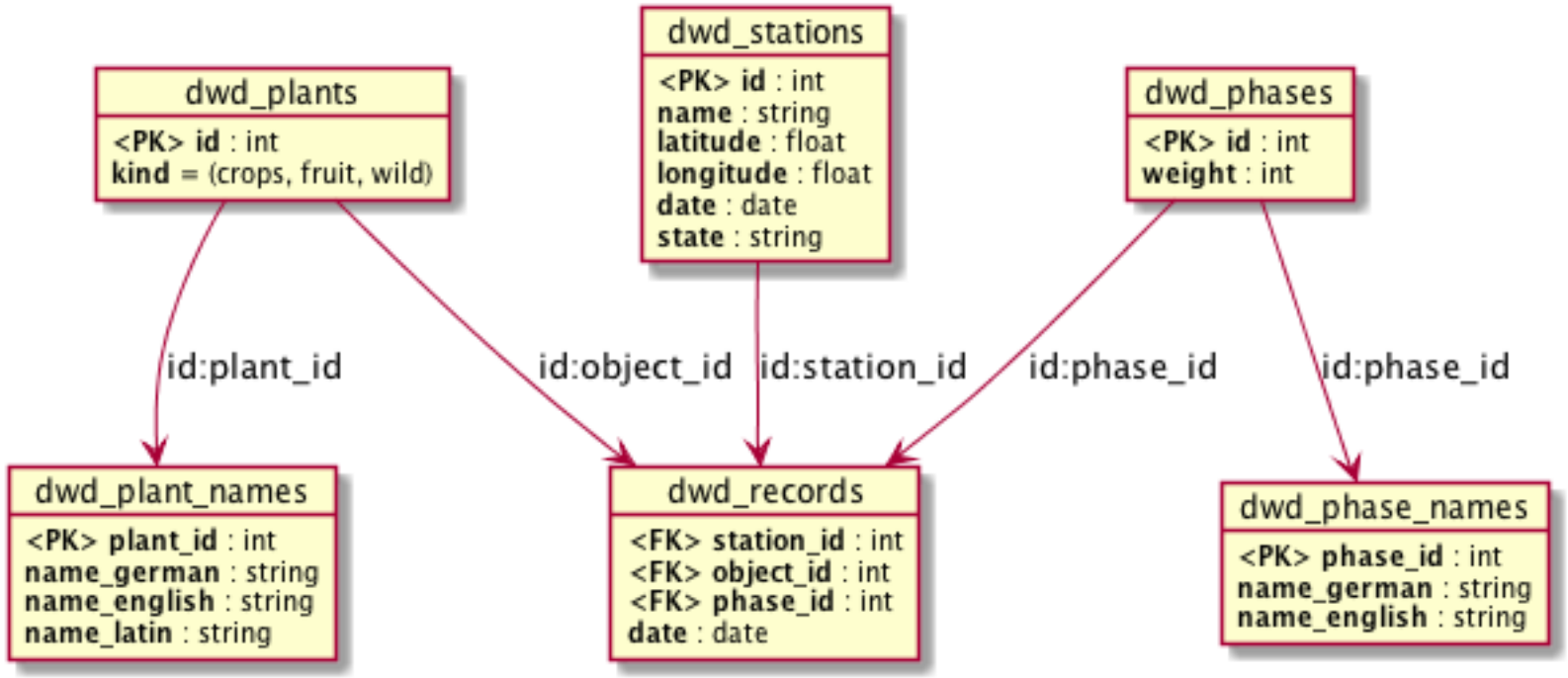
```
Stations_id; … Objekt_id; Phase_id; Eintrittsdatum; … eor;
        140; … 310;             29;       20170806; … eor;
        140; … 310;              5;       20180425; … eor;
```

field name and value transformation

```
 station_id … object_id  phase_id        date …
        140 …       310         29  2017-08-06 …
        140 …       310          5  2018-04-25 …
```

Database

Data processing – normalization



Implementation

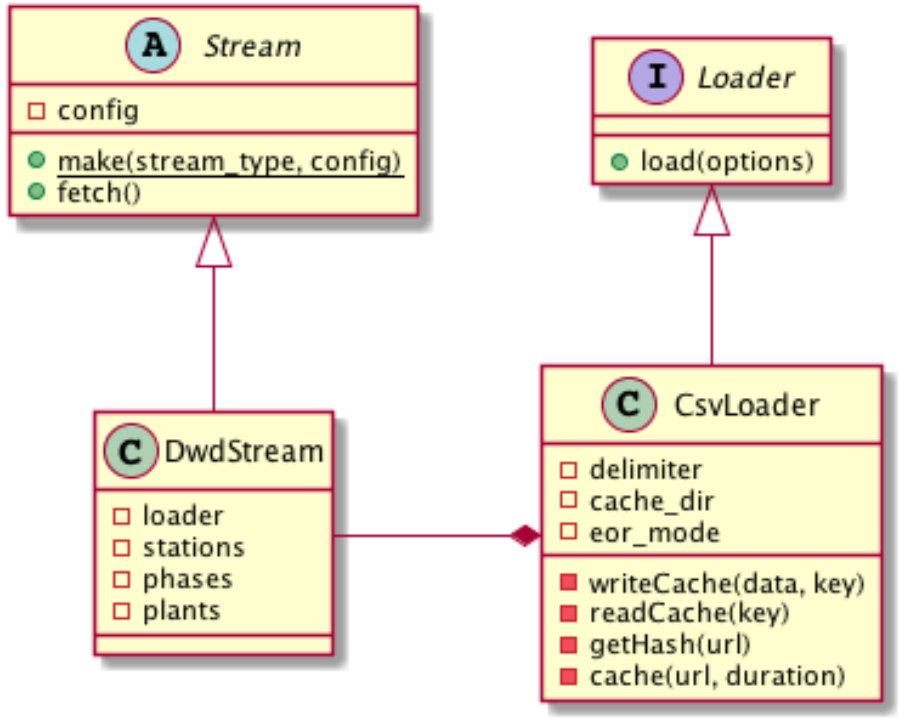
Program implementation & setup

Dashboard creation

Program – structure

Auxiliary functionality such as database migrations is not shown

- Streams are defined in configuration file (JSON format)
- Data loaders used by streams to load data in specific format




```
with open('config.json', 'r') as f:
    config = json.load(f)


    for options in config.get('streams', []):
        # Factory pattern
        stream = Stream.make(options.get('type'), options)
        stream.fetch()
```

Program – data models (peewee ORM)

Model is created for every entity

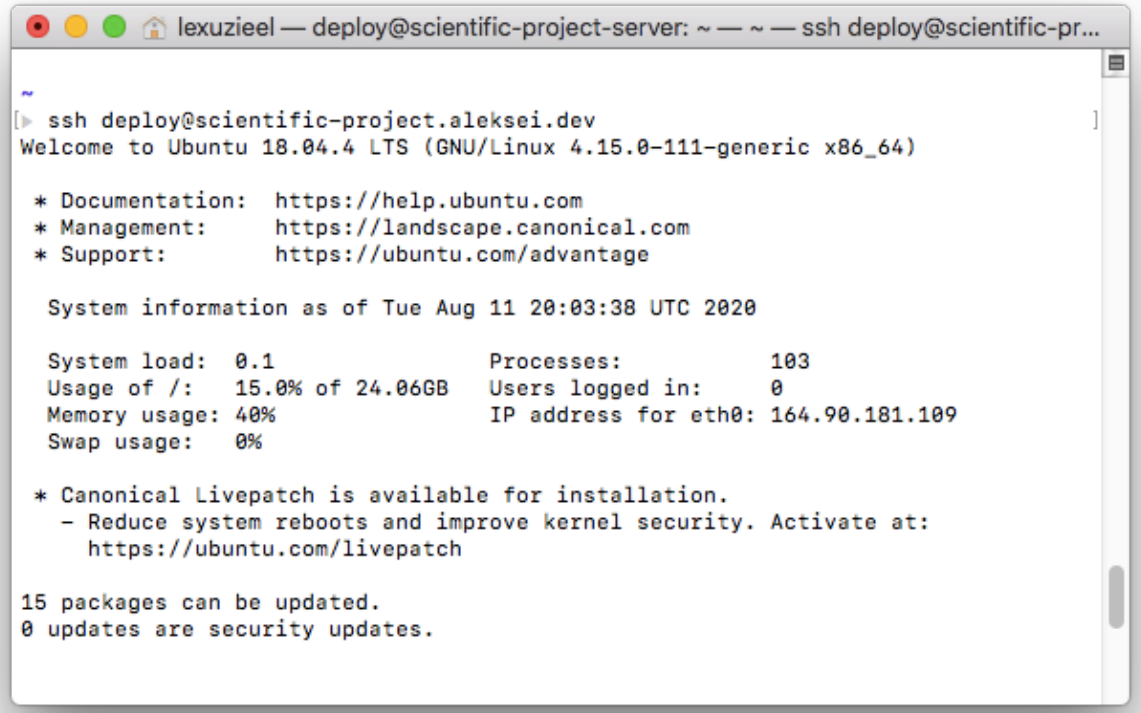
```
from peewee import *
from database.models import BaseModel

class Station(BaseModel):
    class Meta:
        table_name = 'dwd_stations'

    id = IntegerField(primary_key=True)
    name = CharField(null=True, index=True)
    latitude = FloatField(index=True)
    longitude = FloatField(index=True)
    height = FloatField(index=True)
    natural_region_group_code = IntegerField(index=True)
    natural_region_group = CharField(index=True)
    natural_region_code = IntegerField(index=True)
    natural_region = CharField(index=True)
    date = DateField(null=True, index=True, formats=['%d.%m.%Y'])
    state = CharField(index=True)
```

Program – migrations

```
from playhouse.migrate import *
from database.migrations import Migration as BaseMigration
from database.models.dwd import Station, Phase, PhaseName, Plant, PlantName, Record

class CreateTables(BaseMigration):

    def up(self):
        self.db.create_tables([
                Station, Phase, PhaseName, Plant, PlantName, Record
        ])

    def down(self):
        self.db.drop_tables([
                Station, Phase, PhaseName, Plant, PlantName, Record
        ])
```

Program – command-line interface

Server setup – Ubuntu 18.04 VPS on DigitalOcean



- MySQL 5.7.31
- nginx 1.14.0
- Grafana 7.1.0
- Apache 2.4.29
- Python 3.6.9 pipenv

Server setup – configuration

Reverse proxy according to application structure

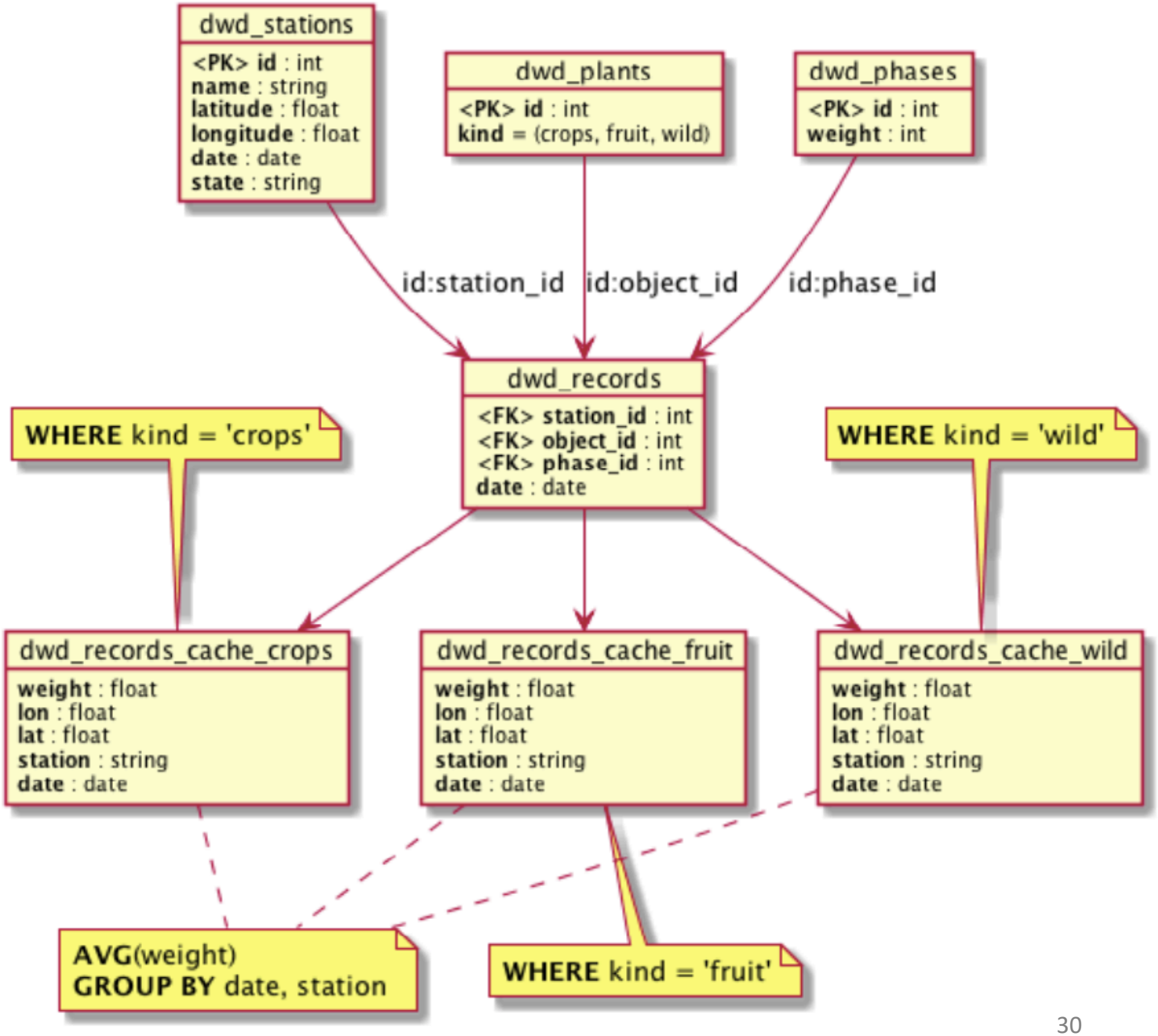
```
server {
        server_name scientific-project.aleksei.dev;

        location ^~ /data/ {
                alias /home/deploy/app/data/;      # Web assets
        }

        location / {
                proxy_pass http://127.0.0.1:3000;  # Grafana
        }
}
```

```
# m h  dom mon dow   command
  0 *   *   *   *    bash /home/deploy/app/run.sh
  0 0   *   *   *    bash /home/deploy/app/backup.sh
  * *   *   *   *    cd app && git pull > /dev/null 2>&1
```

Materialized views

increased read performance

Denormalized data using JOINs



```sql
drop table if exists dwd_records_cache_fruits;
create table dwd_records_cache_fruits
(
    key `weight` (`weight`),
    key `lat` (`lat`),
    key `lon` (`lon`),
    key `station` (`station`),
    key `state` (`state`),
    key `date` (`date`)
)
select avg(dwd_phases.weight)                                          as "weight",
       dwd_stations.longitude                                          as "lon",
       dwd_stations.latitude                                           as "lat",
       dwd_stations.name                                               as "station",
       substring_index(group_concat(dwd_stations.state), ',', 1) as "state",
       dwd_records.date                                                as "date"
from dwd_records
         join dwd_plants on dwd_plants.id = dwd_records.object_id
         join dwd_phases on dwd_phases.id = dwd_records.phase_id
         join dwd_stations on dwd_stations.id = dwd_records.station_id
where kind = 'fruit'
group by date, lon, lat, station;
```

Dashboard creation

Fig. 16: Defined panel query for fruit plants

Fig. 15: Worldmap panel plugin showing data from the database

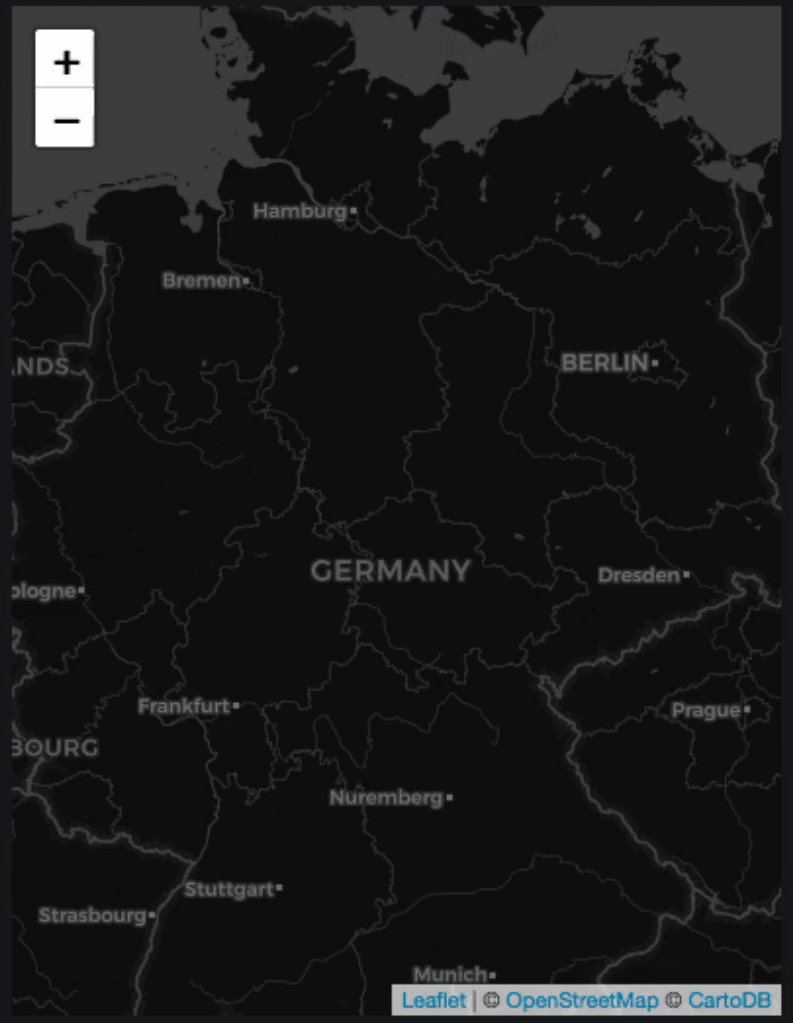
Demonstration

DWD Phenological Data

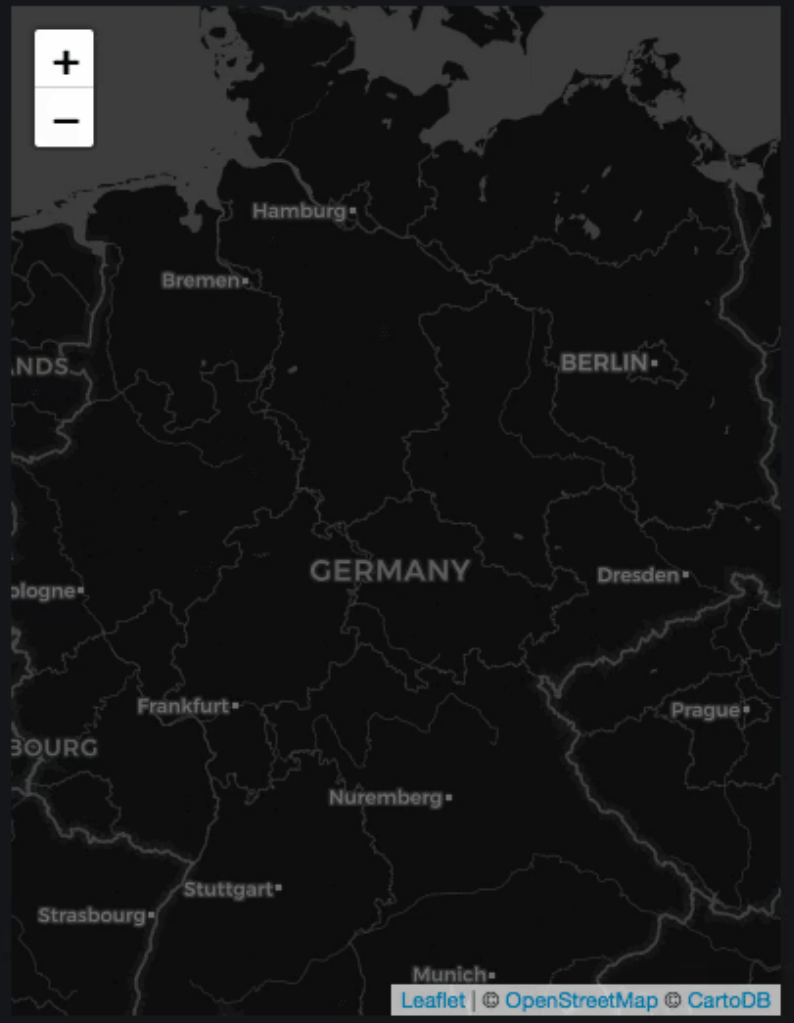
2018-01-02 00:00:00 to 2018-01-08 00:00:00

Autumn leave fall, cut for hay | Begining of turning green | Begining of flowering | Start of ripening | Ripe for picking, ready for harvest

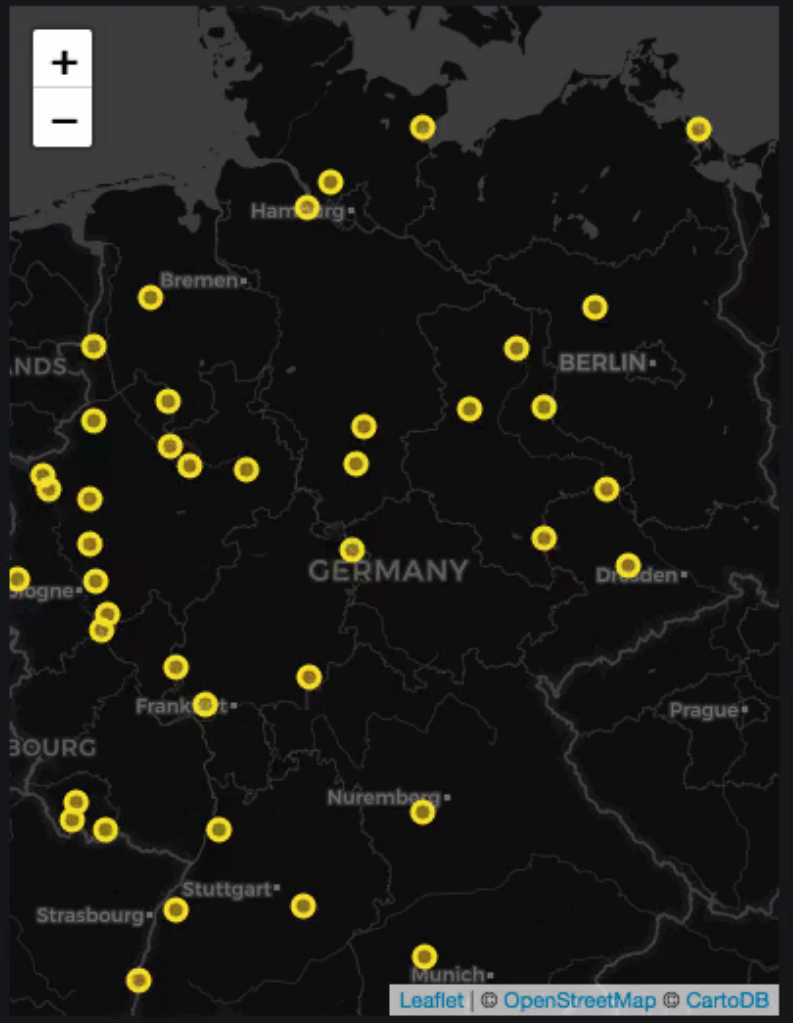
Crops



Fruits



Wild



DWD Phenological Data

Last 7 days

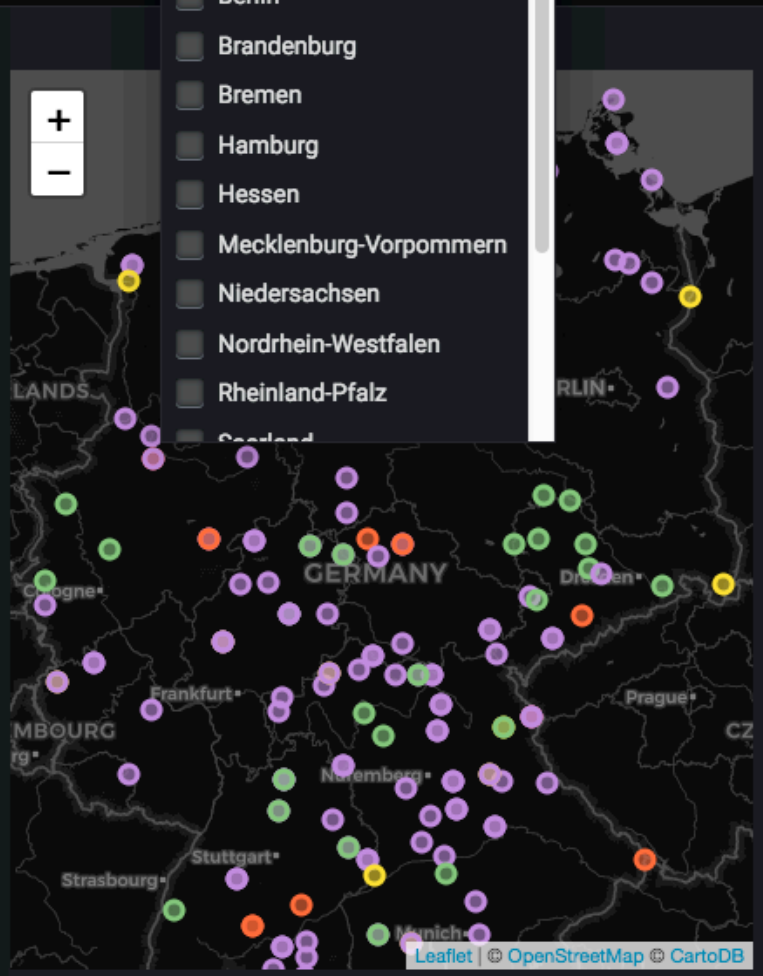
Filter by state

- Selected (1)
- All
- Baden-Württemberg
- Bayern
- Berlin
- Brandenburg
- Bremen
- Hamburg
- Hessen
- Mecklenburg-Vorpommern
- Niedersachsen
- Nordrhein-Westfalen
- Rheinland-Pfalz

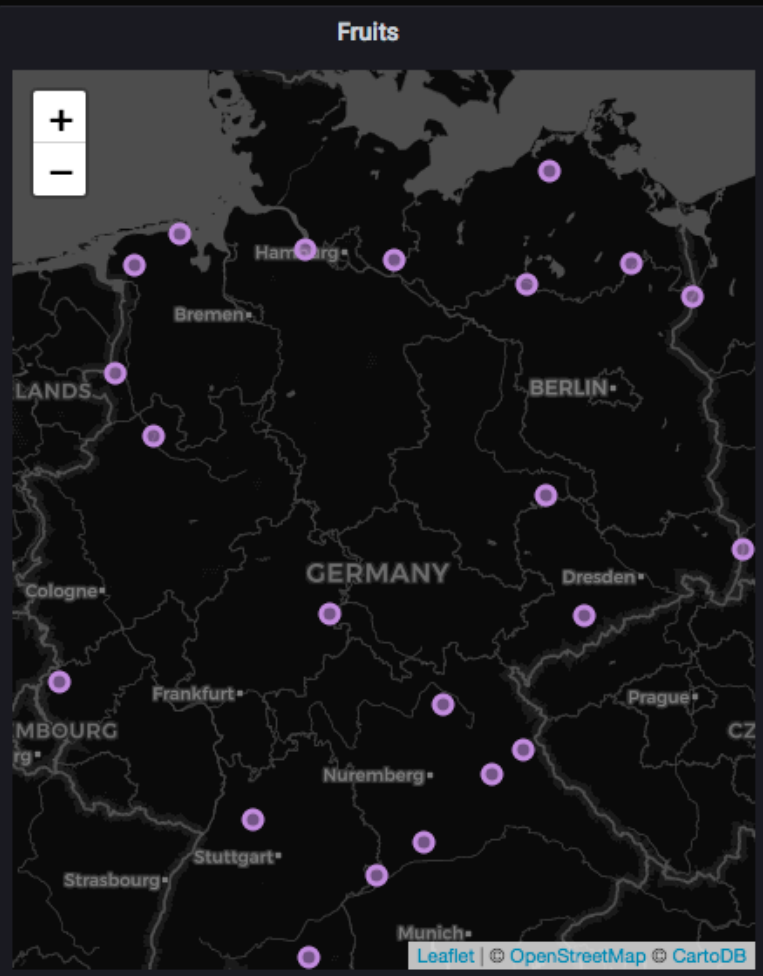
Description

This dashboa VD. Phenological phases are grouped into five categories by assigning a numeric weight to each phase depending on the progress it represents (see legend below).
Plants are divi eat, oat, potato, etc), ***fruit plants*** (cherry, apples, berries, etc) and ***wild*** (birch, hazel, etc). In order to narrow the results you can filter by state.

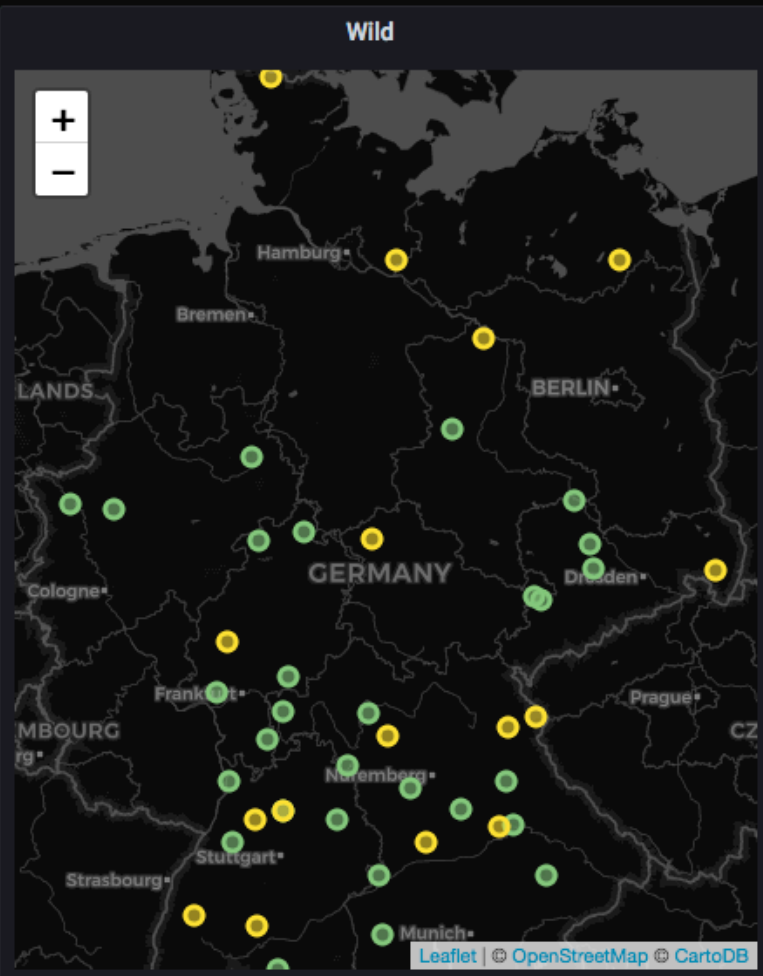
Autumn | of turning green | Begining of flowering | Start of ripening | Ripe for picking, ready for harvest



Fruits



Wild



Conclusion

- Application for ***phenological*** data monitoring
- Methodology was used to formalize data and present it in ***intuitive manner***
- ***Data processing*** pipeline established
- Implemented as ***cloud-based*** solution

Outlook

Phase weighting could be re-evaluated to improve resolution – more steps similar to BBCH scale

Other representations could be added
(i.e. filtering by plant type or specific plant)

Thank you

Live version of the application is available at
https://scientific-project.aleksei.dev

Source code is available at

https://github.com/lexuzieel/dwd-phenology-stream