CIVITAS CONNECT

Workshop 1:
Live-Umsetzung eines Smart City Anwendungsfalls mit CIVITAS/CORE

CIVITAS/CORE

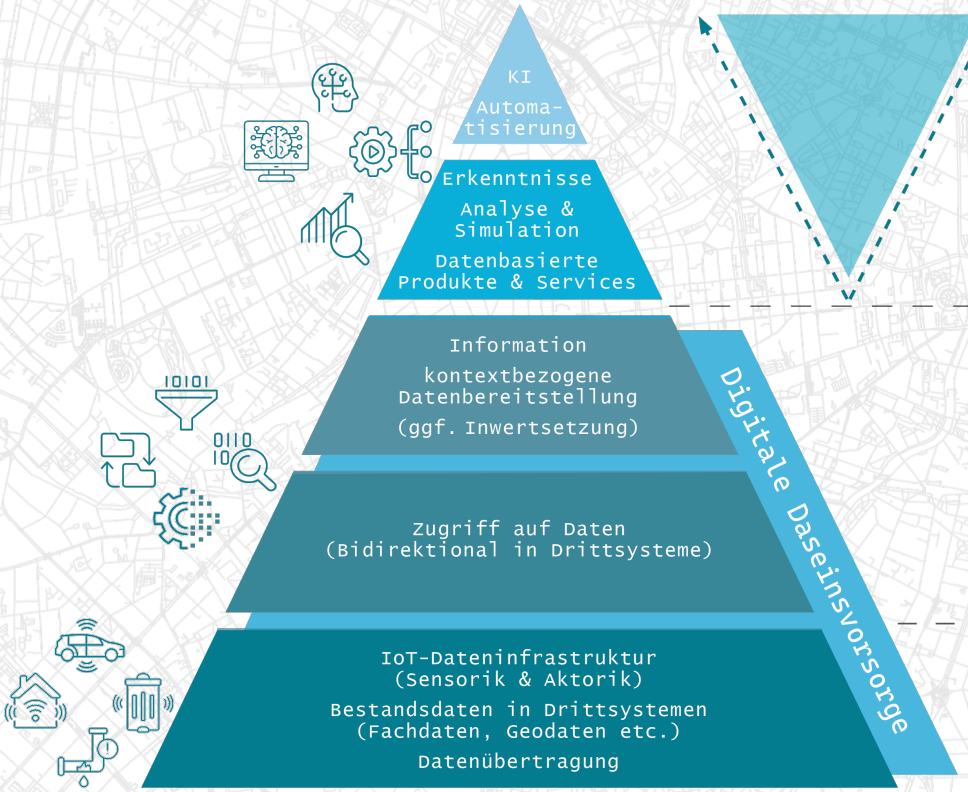
CIVICON

VERNETZT & NACHHALTIG LEBENSRÄUME ENTWICKELN

Agenda

1. Kurzvorstellung CIVITAS/CORE
2. Herangehensweise an Use Case Umsetzung
3. Vorstellung Use Case
4. Austausch

Datenplattformen sind zentral für die datengetriebene Wertschöpfung.



DATENGETRIEBENE WERTSCHÖPFUNG
in Fachverfahren, Services & Prozessen

DATENPLATTFORM

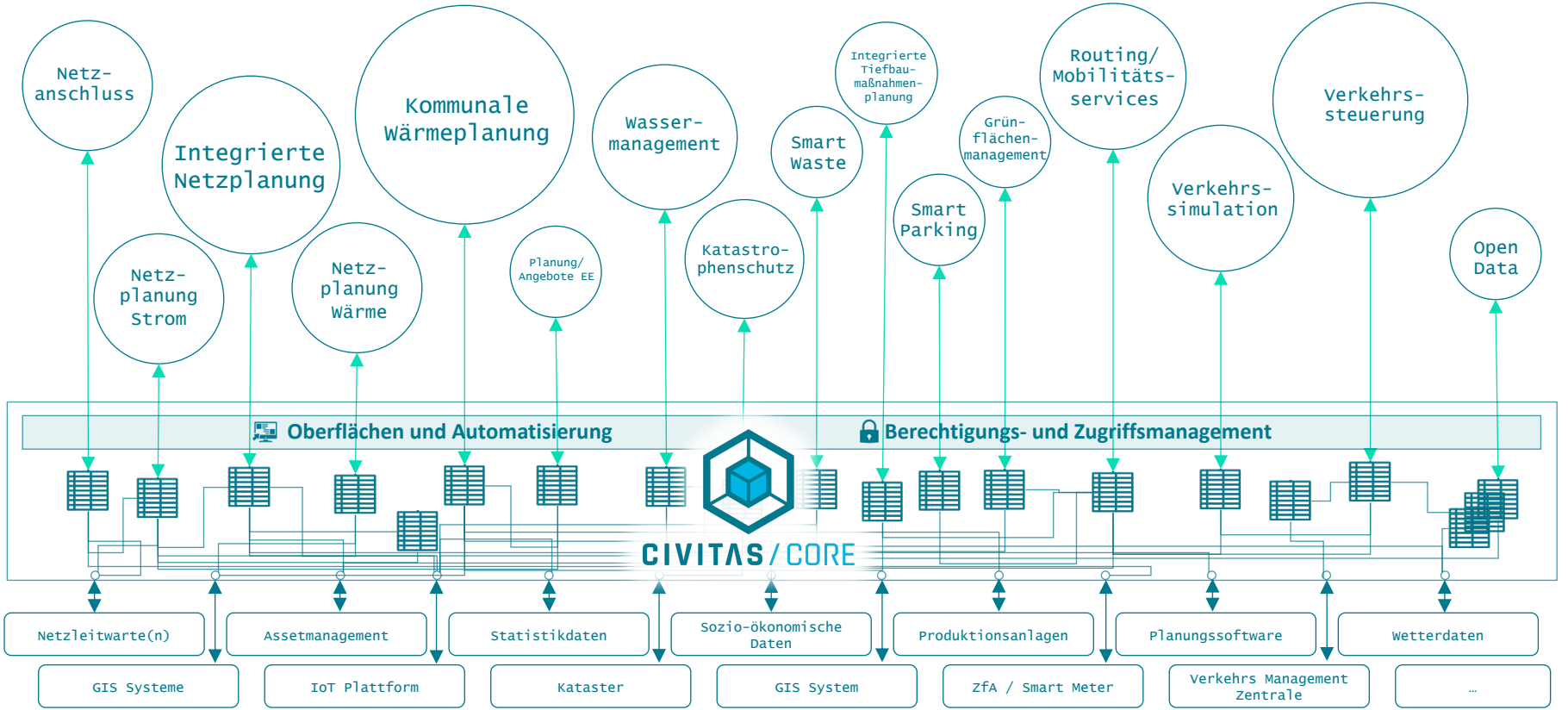
CIVITAS/CORE

DATENGRUNDLAGE

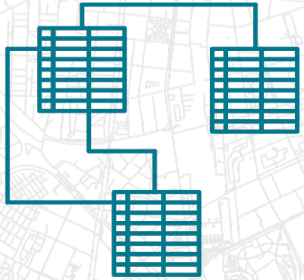
CIVITAS CONNECT

Mit Datenplattformen behalten Sie die Übersicht.

CIVITAS / CORE



Zentrale Funktionen (CIVITAS/CORE 2.0)



Modellzentriertes Datenmanagement

Zentrale Source of Truth,
Daten- und
Standardübergreifend

Oberflächen und Automatisierung

Self-service für den gesamten
Daten-Lebenszyklus
mit hoher Automatisierung

Berechtigungs- und Zugriffsmanagement

Multimandantenfähig
Feingranulare Steuerung auf
Datensatz-Ebene

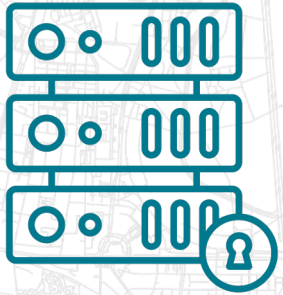
Unsere Grundwerte

Offene Plattform für alle

Use Case unabhängig und Open Source

Kommunale Kooperation

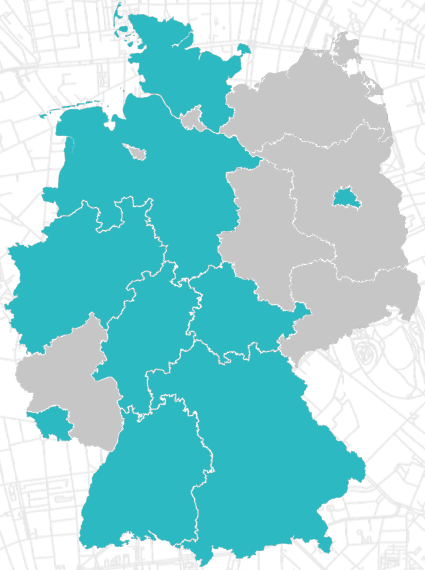
Entwickelt & langfristig gepflegt durch Community aus ausschließlich kommunalen Akteuren



Betreiberneutralität

Eigenbetrieb (on-prem) als auch SaaS aus Dienstleister-Ökosystem

CIVITAS/CORE Community

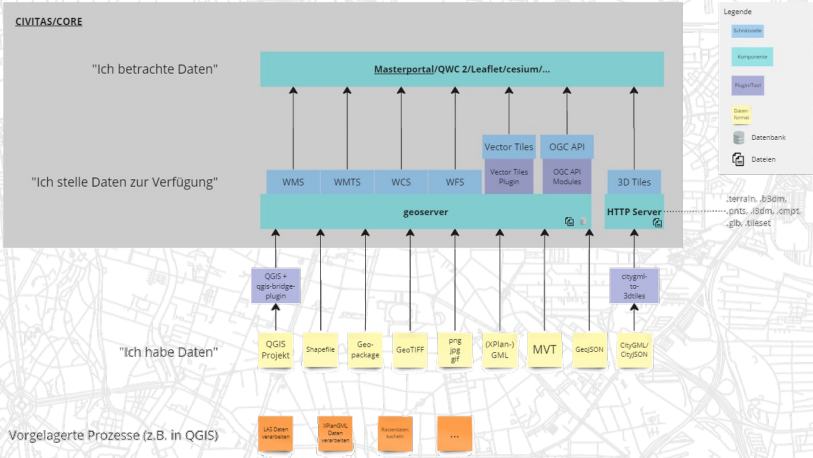


12 Mitglieder

5 Kommunale Unternehmen
7 Städte und Kreise

2. Herangehensweise an Use Case Umsetzung

- UDP bilden die **Infrastruktur** für die Umsetzung von Smart City Use Cases
- Durch die Umsetzung verschiedener Use Cases ergeben sich **Synergien** für Plattform
- UDP sollte möglichst **generisch** aufgestellt sein



Use Case Vorlage

"Use Case Titel"

Anforderungen

User Stories (Rolle, Anforderung, Nutzen):

- als [Nutzerrolle, z. B. Bürger] möchte ich [Aktion, z. B. freie Parplätze auf einer Karte sehen], um [die Zeit der Parplatzsuche zu reduzieren].
- ...

Performance (falls relevant)

- ...

Datenschutz (falls relevnt)

- ...

IT-Security (falls relevant)

- ...

Qualität (falls relevant)

- ...

Weitere Anforderung X

- ...

optional NICHT-Ziele
Welche innerhalb der fachlichen Domäne vorhandenen oder erwartbaren Anforderungen sollen hier NICHT umgesetzt werden:

- ...

Weitere Informationen (existierende Mockups, Diagramme etc.):

Umsetzung

Input

Datenquelle (z. B. REST-API von System X, Umsetzung z. B. mit einem "Konnektor", falls nötig: Transformationen (Datenumwandlungen):

- ...

Verwaltung

Datenmodell (welche Informationen muss ich vorhalten, Datentypen, ...)

- ...

Anforderung an Datenverarbeitung nach Dateneingang (z. B. Daten müssen aggregiert werden):

- ...

Output

Was ist das Zielsystem (z. B. falls Daten an externes System weitergeleitet werden sollen, wie z. B. ERP-System, Umsetzung z. B. mit einem "Konnektor", muss auch gesteuert werden?)

- ...

Anforderungen

User Stories (Rolle, Anforderung, Nutzen)

1. Als [Nutzerrolle, z. B. Bürger] möchte ich [Aktion, z. B. freie Parplätze auf einer Karte sehen], um [die Zeit der Parkplatzsuche zu reduzieren].
2. …

Nicht-Funktionale Anforderungen

Performance, Datenschutz, IT-Security, Qualität, …

Optional: Nicht-Ziele, Weitere Informationen

Umsetzung

Input

Datenquellen (z. B. REST-API von System X, Umsetzung z. B. mit einem "Konnektor", falls nötig: Transformationen (Datenumwandlungen)

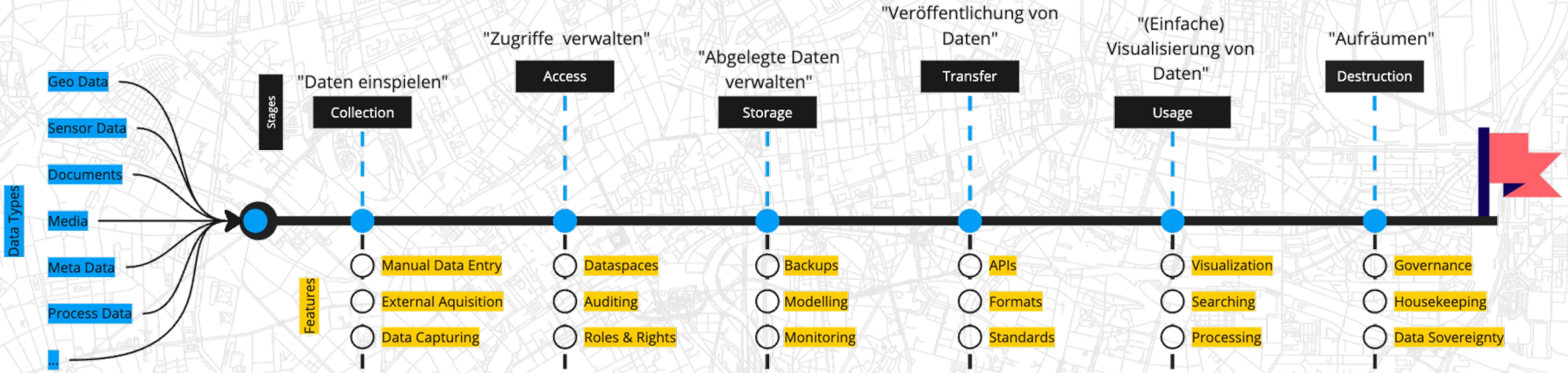
Verwaltung

- Datenmodell (welche Informationen muss ich vorhalten, Datentypen, ...)
- Anforderung an Datenverarbeitung nach Dateneingang (z. B. Daten müssen aggregiert werden)

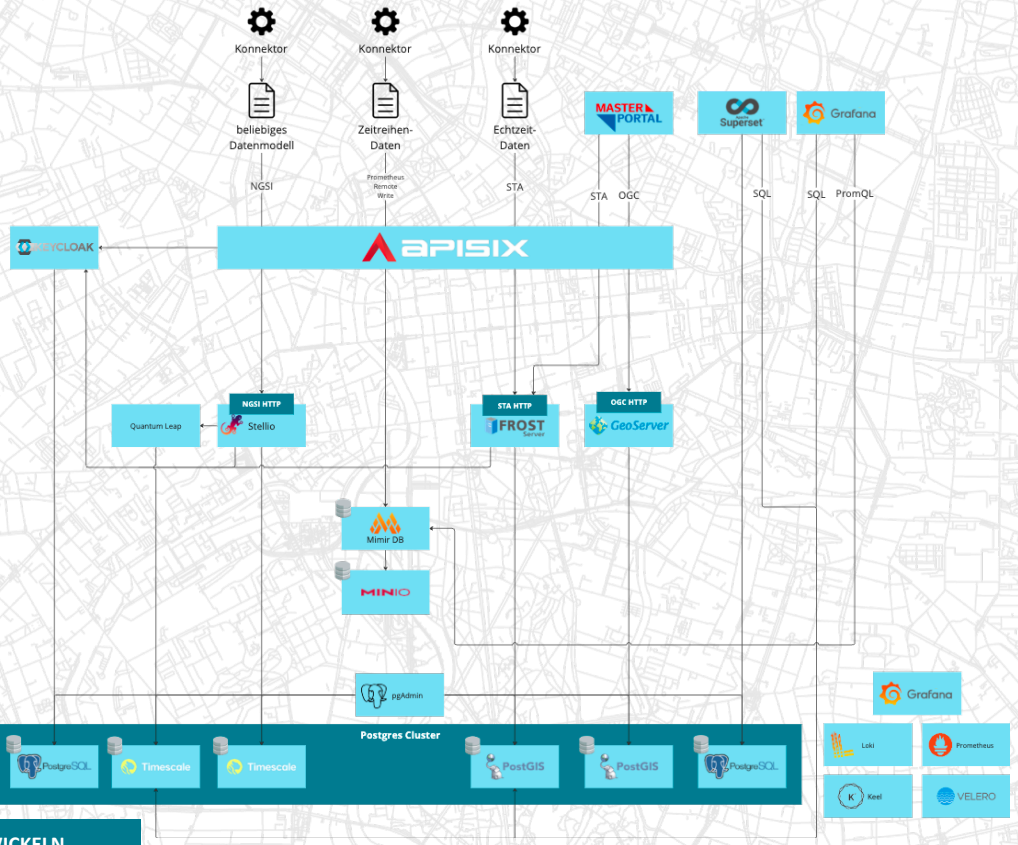
Output

Was ist das Zielsystem (z. B. falls Daten an externes System weitergeleitet werden sollen, wie z. B. ERP-System, Umsetzung z. B. mit einem "Konnektor", muss auch gesteuert werden?)

Data Management Lifecycle



Tech-Stack Version 1.0



3. Use Case: Energiewetteruhr

- User Story: Als Bürger möchte ich wissen, wann viel erneuerbare Energie bei mir verfügbar ist, um Geräte, die viel Strom verbrauchen, dann zu nutzen.

- Nutzt Daten über Erzeugungskapazitäten und Wetterprognose für Prognose auf PLZ-Ebene

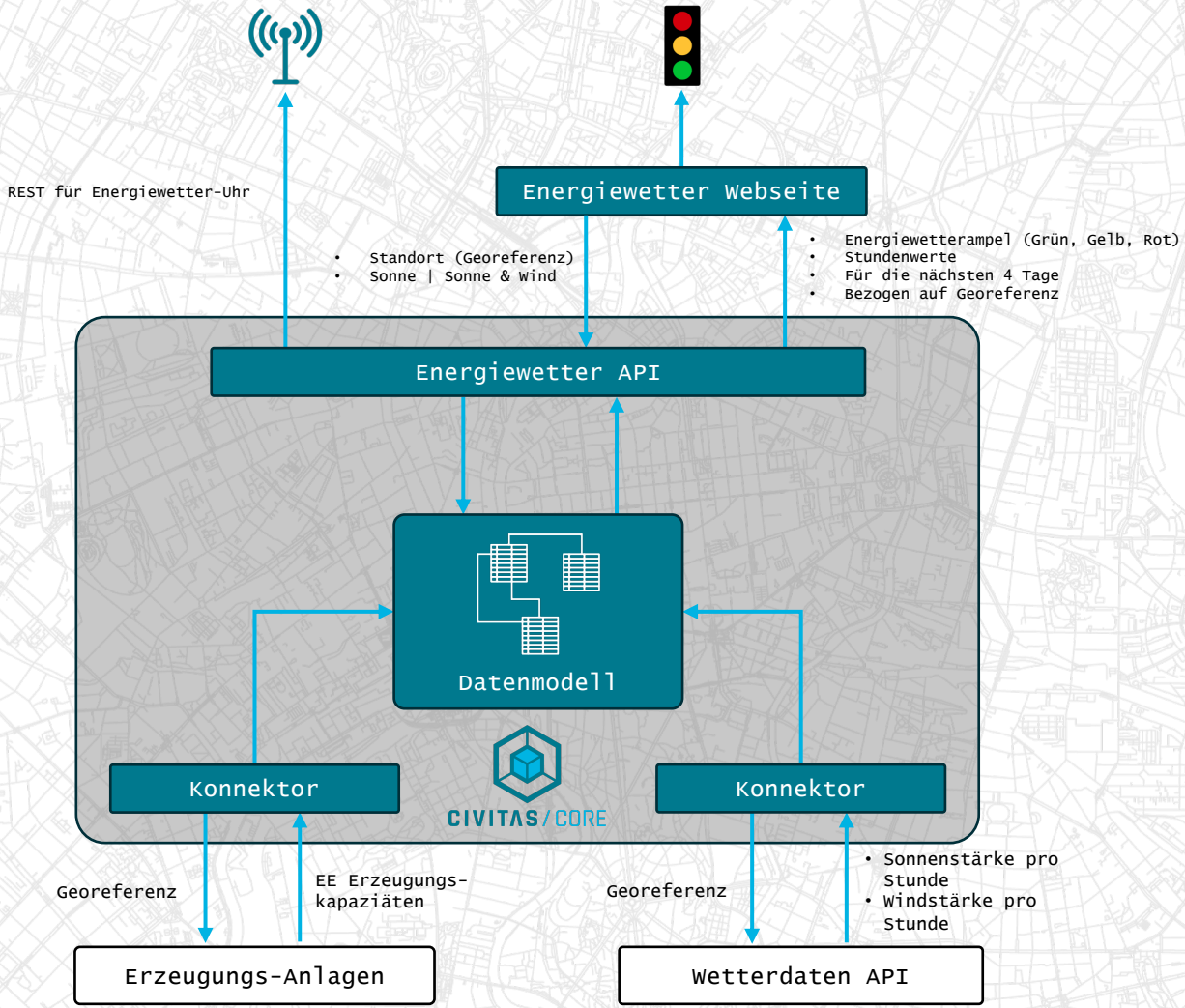
- Stellt Prognose auf 3D-gedruckter Uhr dar

- Basiert auf Projekt von WSW

https://www.wsw-online.de/energie/services/energiewetter/

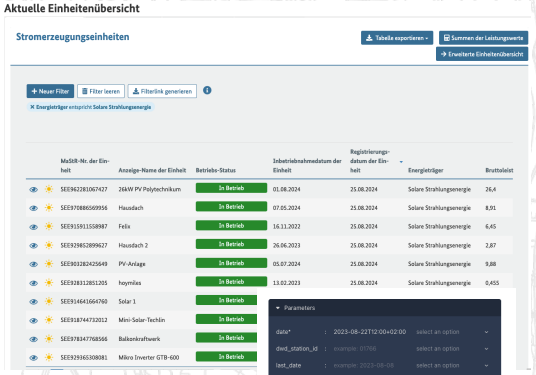
Hands-On

- Sie werden die Energiewetteruhren gleich selbst zusammenbauen
- Dafür sammeln wir die Postleitzahlen, für die Daten erhoben werden sollen


REST für Energiewetter-Uhr
Energiewetter Webseite
• Standort (Georeferenz)
• Sonne | Sonne & Wind
• Energiewetterampel (Grün, Gelb, Rot)
• Stundenwerte
• Für die nächsten 4 Tage
• Bezogen auf Georeferenz
Energiewetter API
Datenmodell
Konnektor
Konnektor
CIVITAS/CORE
Georeferenz
EE Erzeugungs-kapaziäten
Georeferenz
• Sonnenstärke pro Stunde
• Windstärke pro Stunde
Erzeugungs-Anlagen
Wetterdaten API

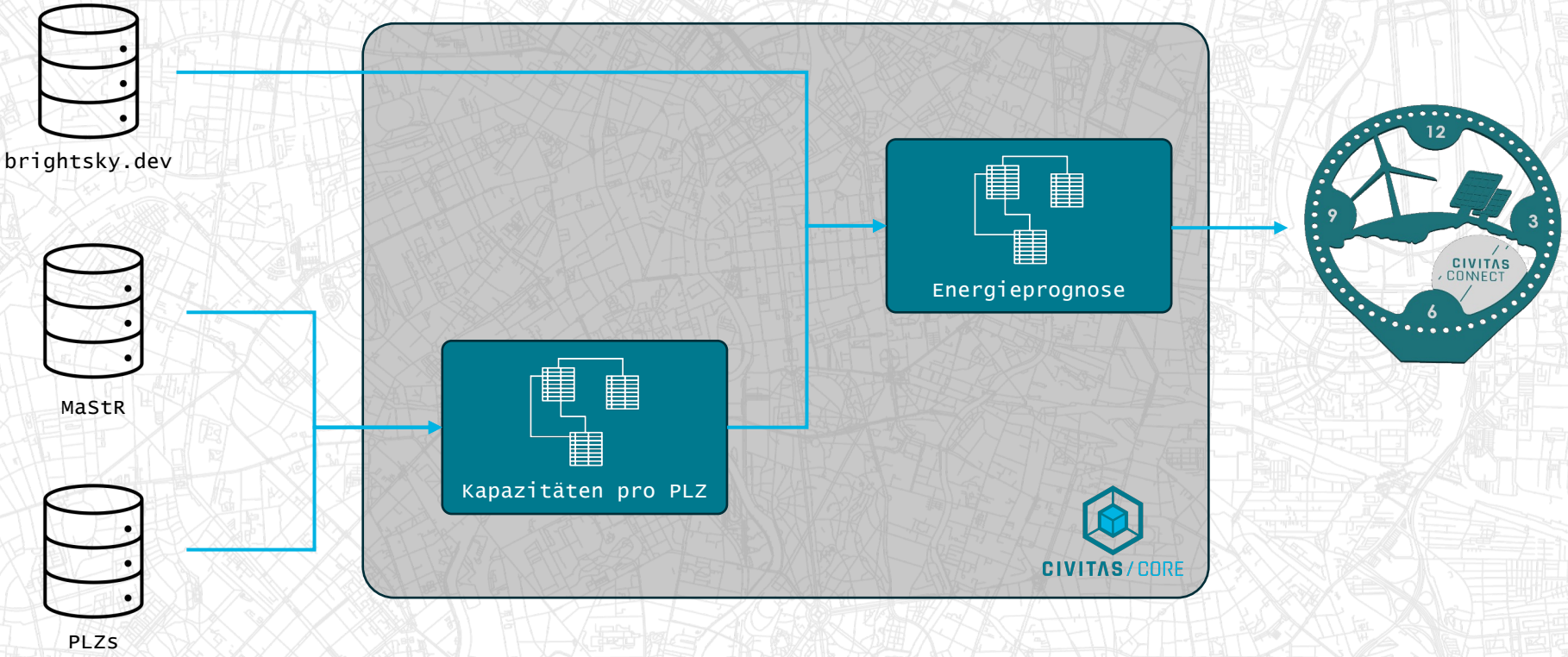
Datenquellen

- Erzeugungskapazitäten: Marktstammdatenregister
- API für Wetterprognose mit Sonneneinstrahlung und Windgeschwindigkeit: brightsky.dev
- Georeferenzierung von Postleitzahlen: public.opendatasoft.com



	Postleitzahl / Post code	PLZ Name (short)	PLZ Name (long)	Kreis code
1	84076	Pfeffenhausen	84076 Pfeffenhausen	09274
2	93195	Wolfsegg	93195 Wolfsegg	09375
3	07557	Gera, Zedlitz u.a.	07557 Gera, Zedlitz u.a.	16052
4	18069	Rostock, Lambrechtshagen	18069 Rostock, Lambrechtshagen	13072
5	92711	Parkstein	92711 Parkstein	09374
6	93138	Lappersdorf	93138 Lappersdorf	09375
7	84427	Sankt Wolfgang	84427 Sankt Wolfgang	09177
8	18147	Rostock	18147 Rostock	13003
9	08523	Plauen	08523 Plauen	14523
10	95685	Falkenberg	95685 Falkenberg	09377

Dataflow


brightsky.dev
MaStR
PLZs
Kapazitäten pro PLZ
Energieprognose
CIVITAS/CORE
12
3
6
9
CIVITAS CONNECT

Datenquellen

MaStR

- XML-Dateien
- Schemata als XSDs, umfangreiche Doku
- für uns statisch
- Mehrere GB an Daten

brightsky.dev

- REST-API
- JSON formatiert, Schema dokumentiert
- Echtzeitanspruch
- leichtgewichtig

opendatasoft PLZs

- CSV Datei
- Dokumentation im Datenportal
- Statisch
- 90 MB

Data Preparation

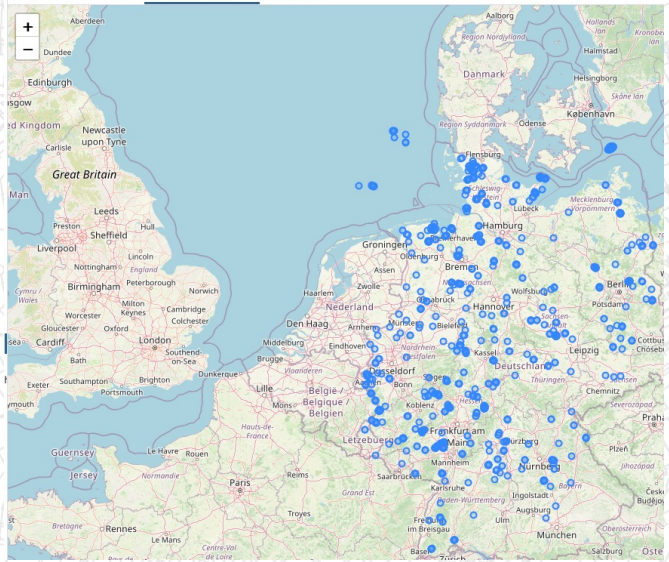
- Der MaStR Datensatz muss auf Georeferenzen der Anlagen untersucht werden
  - Nicht alle Anlagen haben Postleitzahlen. Wie kann man auf diese schließen?
- Ansatz: Daten in PostGIS DB von CIVITAS/CORE einspielen
  - Hier SQL + Geobasierte Funktionen zum Vorbereiten der Daten nutzen
- Tabellen anlegen: aus XSD-Dateien SQL zum Erstellen der Tabellen nutzen mittels XSLT
- Daten einspielen mittels Skript

MaStR Datensatz

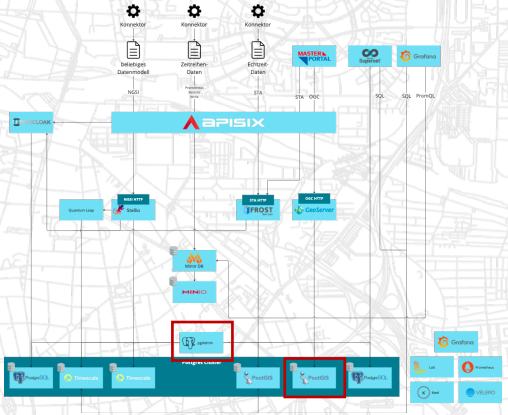
Query | Query History

```
1  SELECT * FROM einheitwind WHERE postleitzahl is null;
2
3  SELECT ST_SetSrid(ST_MakePoint(laengengrad, breitengrad), 4326) FROM einheitwind;
4
5  SELECT * FROM einheitsolar WHERE postleitzahl is null;
6
7  SELECT ST_SetSrid(ST_MakePoint(laengengrad, breitengrad), 4326) FROM einheitsolar;
8
9
```

Data Output | Messages | Geometry Viewer | Notifications



weic character varying (255)	weic_nv smallint	weicdisplayname character varying (255)	kraftwerksnummer character varying (255)	kraftwerksnummer_nv smallint	energietraeger smallint	bruttoleistung real	nettonennleistung real
[null]	0	[null]	[null]	0	2497	15000	15000
[null]	1	[null]	[null]	0	2497	4200	4200
[null]	0	[null]	[null]	0	2497	15000	15000
11WD2OWVM000355S	0	VEJA MATE	[null]	0	2497	6300	6300
[null]	1	[null]	[null]	0	2497	4200	4200
[null]	1	[null]	[null]	0	2497	3780	3780
11WD8BALT3W---6	0	BALTICERZ	[null]	0	2497	3780	3780
[null]	1	[null]	[null]	0	2497	3780	3780
[null]	0	[null]	[null]	0	2497	15000	15000
[null]	1	[null]	[null]	0	2497	4200	4200
11WD8BALT3W---6	0	BALTICERZ	[null]	0	2497	3780	3780
11WD8BALT3W---6	0	BALTICERZ	[null]	0	2497	3780	3780
11WD8BALT3W---6	0	BALTICERZ	[null]	0	2497	3780	3780
11WD8BALT3W---6	0	BALTICERZ	[null]	0	2497	3780	3780
11WD8BALT3W---6	0	BALTICERZ	[null]	0	2497	3780	3780

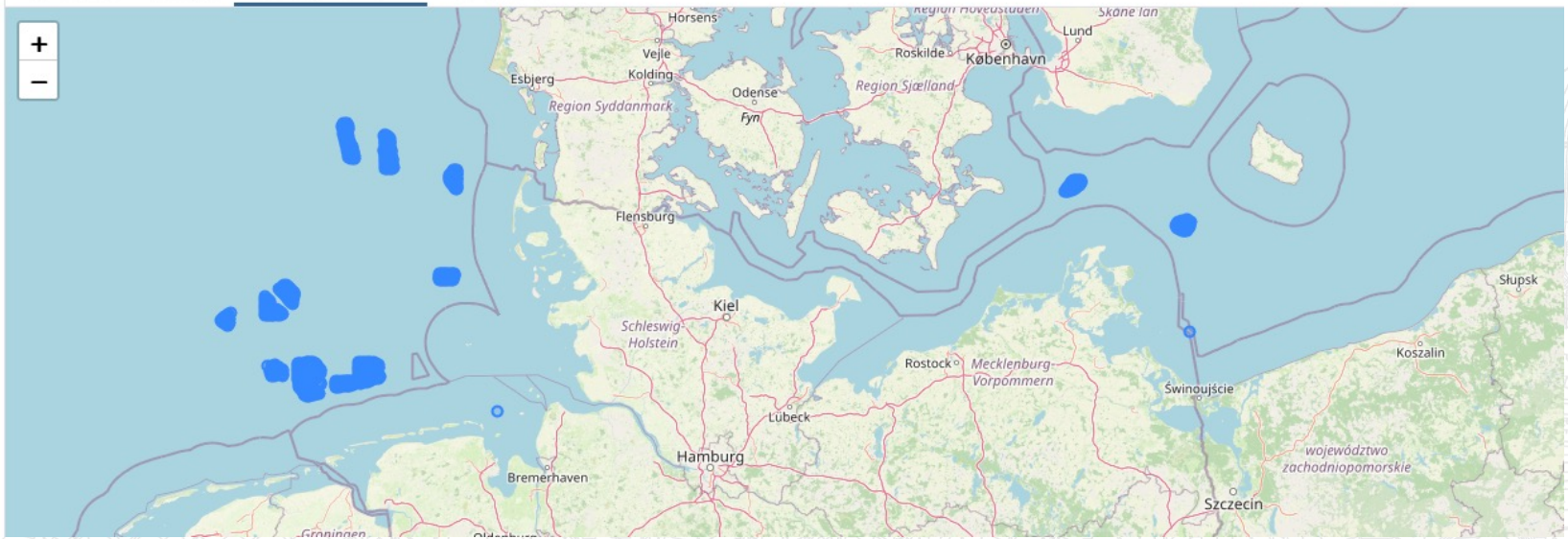


MaStR Datensatz

Query | Query History

```
1  SELECT e.einheitmastrnummer, ST_SetSrid(ST_MakePoint(e.laengengrad, e.breitengrad), 4326) FROM einheitwind e WHERE e.postleitzahl is null;
2
```

Data Output | Messages | Geometry Viewer | Notifications



Data Preparation

- Die meisten Anlagen haben gepflegten Postleitzahlen
- -> Die, die keine haben, können über ihre Koordinaten einer PLZ zugeordnet werden
- Dazu nutzen wir den PLZ-Datensatz und PostGIS

PLZ Datensatz

- Datensatz ist CSV Datei mit 10 Spalten -> wesentlich weniger Komplexität
- Per Skript Daten einlesen und dann in PostGIS importieren
- Dabei können Geoinformationen in PostGIS Datentypen umgewandelt werden

```
async function importPLZs() {
    const fileContent = fs.readFileSync('georef-germany-postleitzahl.csv', { encoding: 'utf

    return (await parsePromise(fileContent, {
        delimiter: ';',
        columns: ['Name', 'PLZ_Name_short', 'PLZ_Name_long', 'Geometry', 'Postleitzahl', 'K
        from: 2,
        cast: (value, context) => {
            if (context.column === 'Geometry' && value !== 'Geometry') {
                return JSON.parse(value)
            } else if (context.column === 'geo_point_2d' && value !== 'geo_point_2d') {
                return JSON.parse('[' + value + ']')
            } else {
                return value
            }
        }
    })).records;
}
```

```
async function exportPLZs(records: any) {
    const sql: string[] = records.map(({ Name, Geometry, geo_point_2d }: { Name: string, Geometry: any, geo_point_2d: any }) =>
        `INSERT INTO postleitzahlen (plz, area, center) VALUES (${Name}, ST_GeomFromGeoJSON('${JSON.stringify(Geometry)}'),

    console.log('Generated sql statements.')

    const client = new Client({
        connectionString: 'postgres://test:ChangeMe@localhost:5432/MaStr'
    })
    await client.connect()

    let i = 0
    for (const line of sql) {
        console.log(`Insert line ${i++}`)
        await client.query(line)
    }
    console.log('Inserted lines.')

    await client.end()
}
```

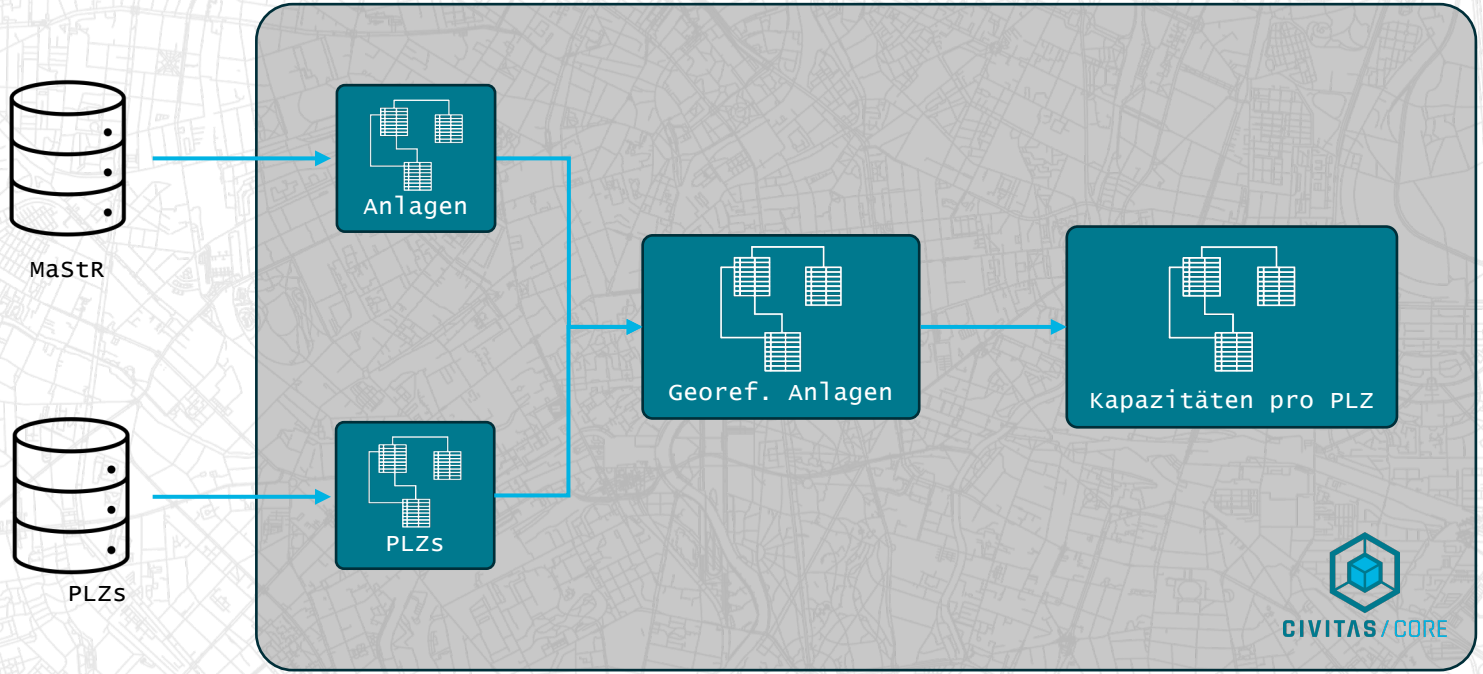
Datenfusion

- Ziel: mit PLZ-Datensatz MaStR-Datensatz vervollständigen

- Mit PostGIS können wir zu jeder Anlage die nächstgelegene Postleitzahl finden:

```
SELECT e.nettonennleistung, p.plz, p.area, ST_SetSrid(ST_MakePoint(e.laengengrad, e.breitengrad), 4326) as point FROM einheitsolar e
CROSS JOIN LATERAL (
  SELECT plz, area, area <-> ST_SetSrid(ST_MakePoint(e.laengengrad, e.breitengrad), 4326) AS dist
  FROM postleitzahlen AS p
  ORDER BY dist
  LIMIT 1
) p
 WHERE e.postleitzahl is null;
```

Data Lineage Kapazitäten



Berechnung Prognose

- Im MaStR sind nicht alle Informationen zur Berechnung vorhanden
  -> Anlagen werden pro PLZ aggregiert: Durchschnittswerte annehmen

- Folgende Informationen werden verwendet:
  - Wetter: Sonneneinstrahlung, (Sonnenschein pro Stunde), Windgeschwindigkeit
  - Anlagen: Nettonennleistung, Leistung Wechselrichter (Solar), Rotorblattradius (Wind)

- Für statische Daten (Anlagen) wird Tabelle angelegt

Speicherung der Prognosedaten

- Zur Verfügung gestellt werden sollen immer Prognosedaten für die nächsten Stunden pro registrierter PLZ

- SensorThingsAPI Datenmodell eignet sich gut für zeitbasierte Werte
  - Daten lassen sich über REST und MQTT Schnittstellen zur Verfügung stellen
  - -> wir müssen weder Datenmodell noch Schnittstellen selbst bauen

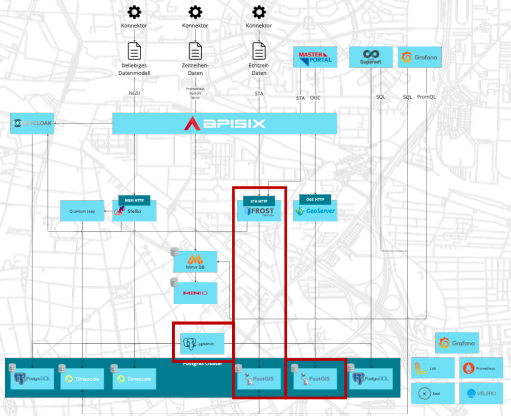
Daten Integration in FROST-Server

```
name: `Erzeugungskapazität ${p.PLZ_Name_short}`,
description: `Erzeugungskapazität für ${p.PLZ_Name_long}`,
properties: {
    usecase: 'Energiewetteruhr',
    plz: p.Name,
    solar: s,
    wind: w
},
Locations: [
    {
        name: `${p.PLZ_Name_long}`,
        description: `Mittelpunkt von ${p.PLZ_Name_long}`,
        encodingType: "application/vnd.geo+json",
        location: {
            type: "Point",
            coordinates: [
                p.geo_point_2d[1],
                p.geo_point_2d[0]
            ]
        }
    }
]
```

- Daten aus PostGIS in FROST-Server exportieren
- Daten sind statisch -> in „Things“ Entities speichern
  - “Location“ des Things für Georeferenz der PLZ nutzen
- Ganzer Datensatz kann per Batch-Aufruf importiert werden

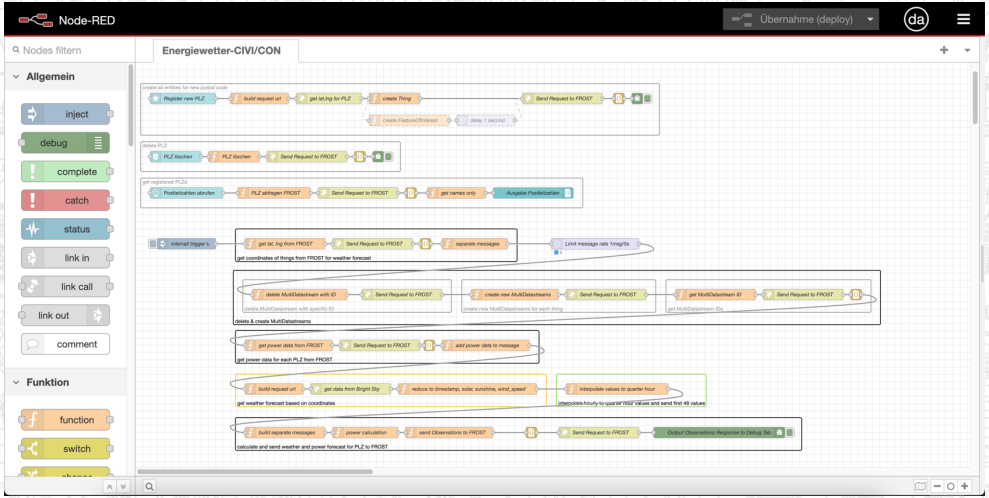
  `https://api.civitasconnect.cloud/context/sensorthings/v1.1/$batch`

- Wenn wir Anlagendaten regelmäßig aktualisieren würden, würden wir eher Datastream nutzen



Node-RED

- Low-Code-Plattform
- Nutzung vorgefertigter Bausteine
- Browser-basierter Editor



Stärken Node-RED

- Visuelle Programmiermöglichkeit
- Schneller und niedrigschwelliger Einstieg in die Programmierung
- Schnelle Entwicklung von Prototypen
- Große Auswahl vorgefertigter Nodes
- Einfache Erweiterbarkeit durch Erweiterungen von Dritten
- Einfache Integration von APIs

Grenzen Node-RED

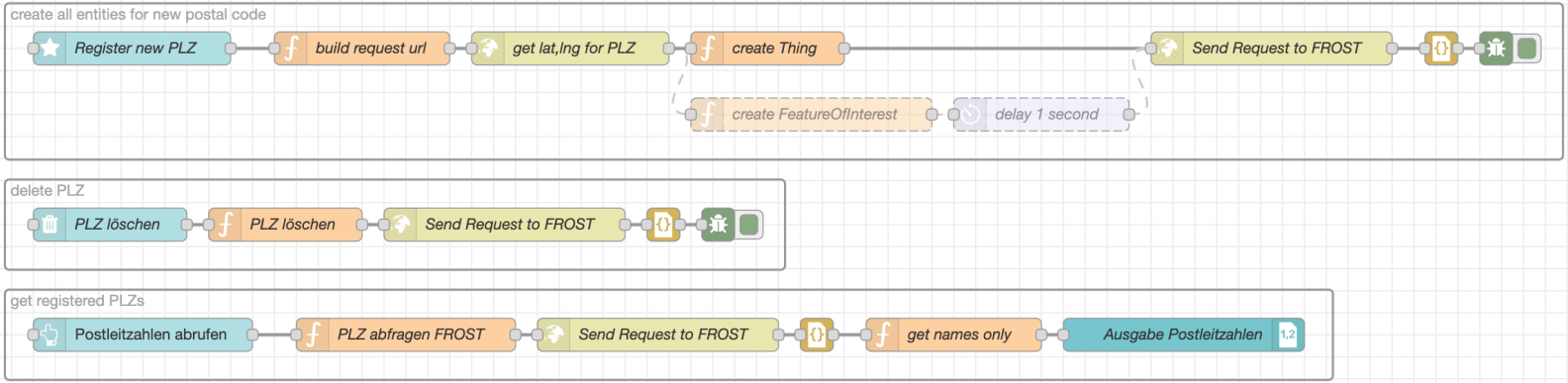
- Limitierungen bei sehr komplexen Berechnungen
- Begrenzte Möglichkeiten zur Fehlerbehebung und Debugging
- Abhängigkeiten bei Nutzung von Drittanbieter-Erweiterungen

Einsatzmöglichkeiten Node-RED

- Anbindung von IoT-Hardware
- Automatisierung von Prozessen
- Umsetzung einfacher Berechnungsmodelle
- Datenharmonisierung & -speicherung
- Integration von APIs (Wetterdienste, FROST / Sensorthings API, …)

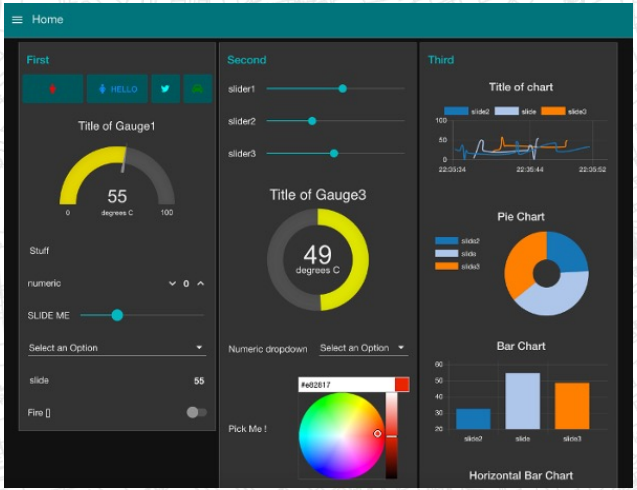
Genutzte Flows

- Flows zur Verwaltung (Anlegen, Löschen, …) von Postleitzahlenbereichen
- Nutzung von UI-Erweiterung zur Ein- und Ausgabe via separate UI-Oberfläche

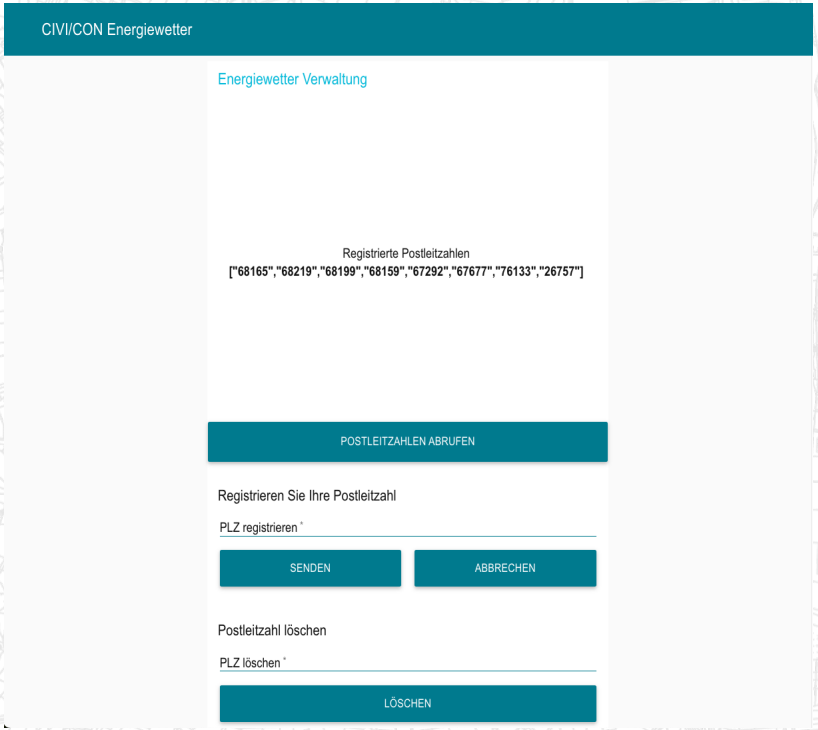


UI Erweiterung

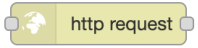
- "node-red-dashboard (3.6.5)"-Erweiterung zur Realisierung der UI



https://flows.nodered.org/node/node-red-dashboard

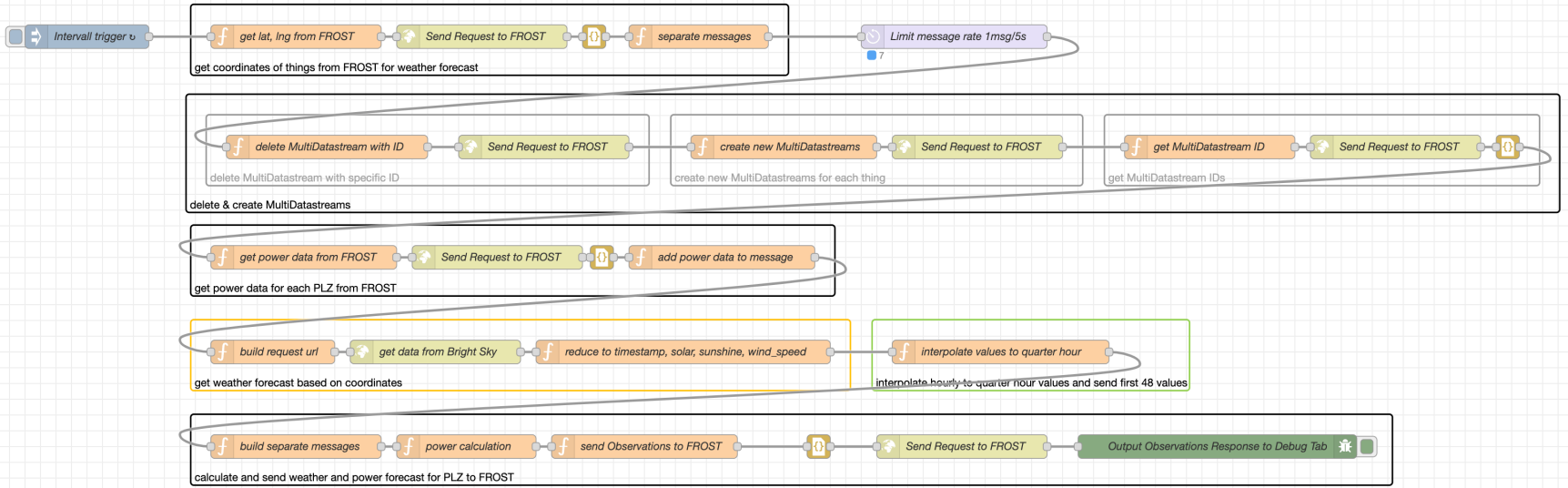


Genutzte Flows



- Flow zur Arbeit mit Leistungs- und Wetterdaten und Integration des Berechnungsmodells
- Inject-Node als periodischer Trigger
- http-Nodes für API-Calls
- Function-Nodes
  - Zur dynamischen Erstellung der Request-URLs
  - Zur Extraktion von Daten
  - Zur Realisierung von Berechnungen
- Delay-Node zur Spreizung der API Calls

Genutzte Flows



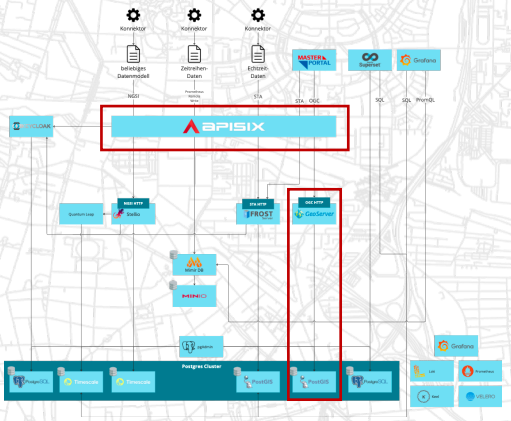
Datenbereitstellung

- Request zur Abfrage:

```
https://api.civitasconnect.cloud/context/sensorthings/v1.1/MultiDatastreams?
$filter=
    substringof(,<PLZ>', name) and
    Observations/phenomenonTime lt now() add duration'PT12H' and
    Observations/phenomenonTime ge now()
&$expand=Observations($select=result, phenomenonTime)
&$select=Observations/result, Observations/phenomenonTime
```

  - relativ komplex und nur mit Authentifizierung zugänglich

➔ Mit APISIX eigene API für Use Case definieren



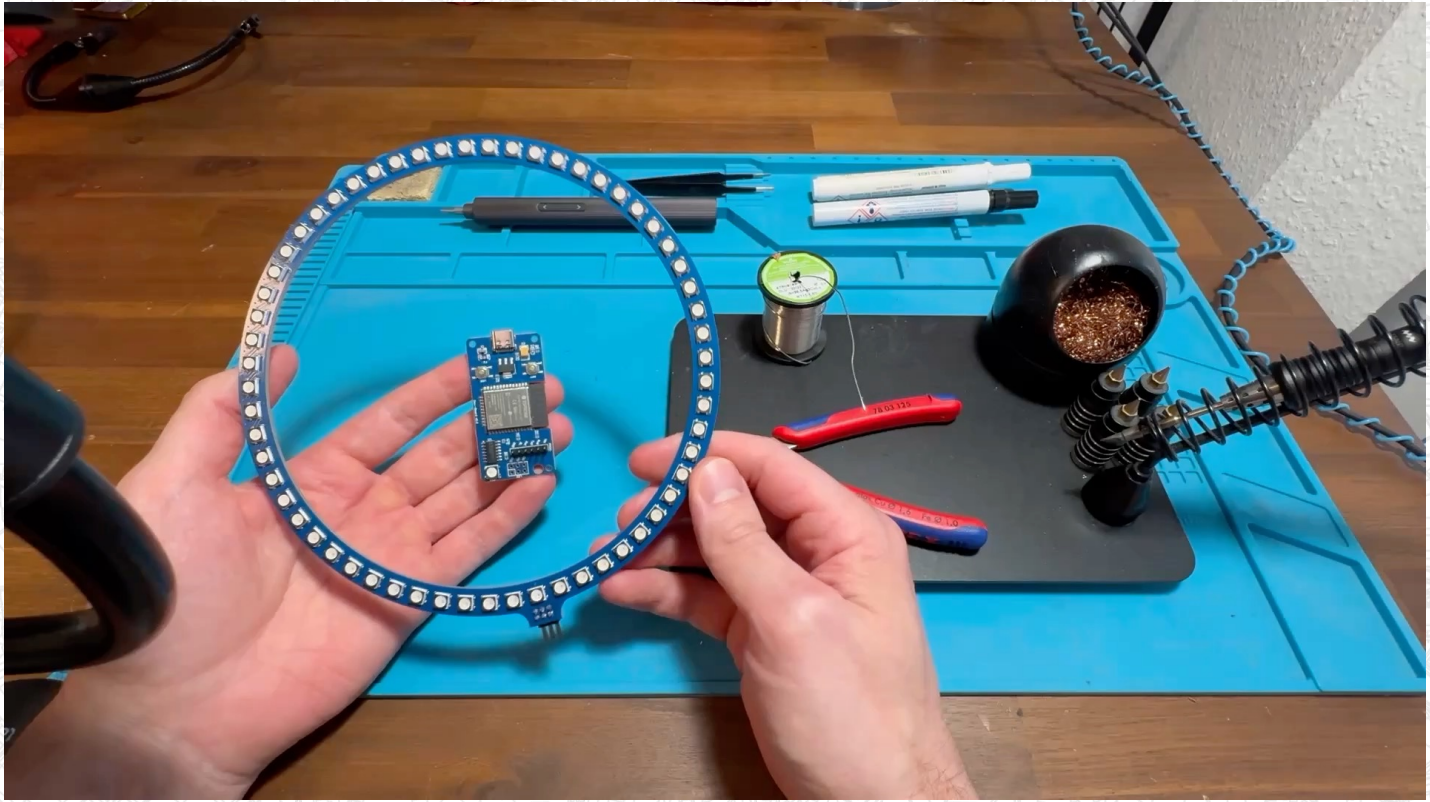
- Dann ist API einfach zu verwenden:

  https://api.civitasconnect.cloud/energiewetter/ >

  WLAN
  SSID: Tenda
  Password: 41681797

- WLAN
- SSID: Tenda
- Password: 41681797
- URL: energiewetter-<ID siehe Uhr>.local
- IP: 192.168.4.1

Zusammenbau Hardware



Learnings

- Datenimport erfordert verschiedenste Formate und Tools
- Datenvorbereitung macht größten Teil aus
  - Daten müssen vor Nutzung umfassend untersucht werden können
  - Daten müssen immer bereinigt werden
- Use Case bestehen meistens aus einer Kette von Datensätzen und Datenmodellen
  - Transformation und Fusion von Daten sind nötig
- Standardisierte Datenmodelle und Schnittstellen sparen Aufwände

Ausblick

- Mit CIVITAS/CORE 1.0 müssen einige Schritte der Use Case Umsetzung noch außerhalb durchgeführt werden

- CIVITAS/CORE 2.0 wird hier gezielt unterstützen
  - Konnektor-Baukasten ermöglicht Import verschiedenster Datenformate
  - Benutzeroberfläche deckt gesamten Prozess der Use Case Erstellung mit Datenmodellen ab
  - Datensätze lassen sich unabhängig von Quelle/Format/Modell miteinander verschneiden
  - Alle Daten lassen sich über mehrere Standard Schnittstellen abrufen

CIVITAS CONNECT

Folien und Materialien

VERNETZT & NACHHALTIG LEBENSRÄUME ENTWICKELN

Robin Lamberti
Lead Architect CIVITAS/CORE

r.lamberti@civitasconnect.digital
Mobil +49 171 9668 718

CIVITAS / CORE

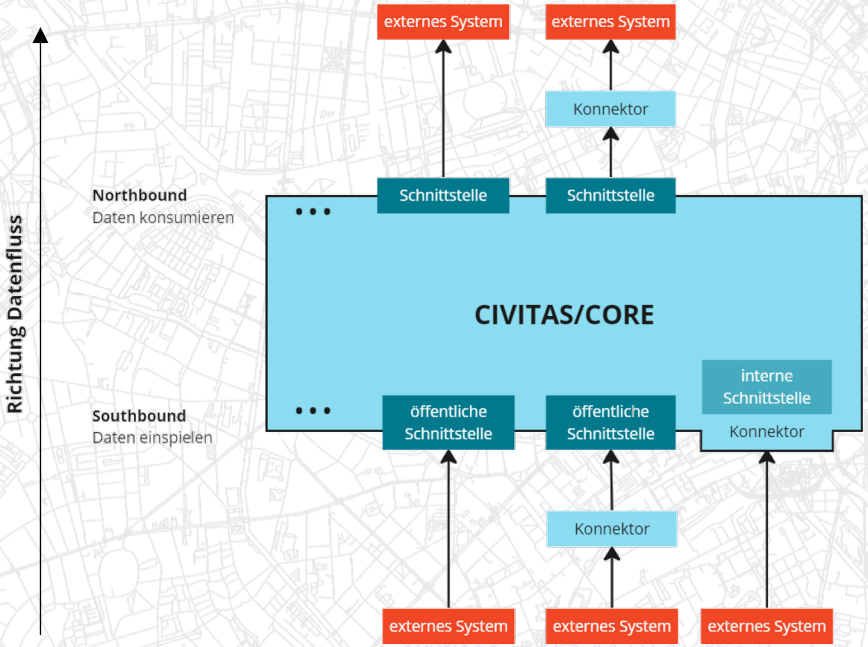
Balthasar Weitzel
Abteilungsleiter Smart City Engineering
Fraunhofer IESE

balthasar.weitzel@iese.fraunhofer.de

Steffen Hupp
Senior Full-Stack Developer
Fraunhofer IESE

steffen.hupp@iese.fraunhofer.de

Schnittstellen & Konnektoren



Abgrenzung

Öffentliche Schnittstellen bietet CIVITAS/CORE standardisiert an, für den Datenaustausch zu externen Systemen. Sind von außerhalb der Plattform erreichbar

Konnektoren "ergänzen Schnittstellen". Es sind optionale, individuelle Adapter, die die Anschlussfähigkeit zu speziellen externen IT-Systemen herstellen, die nicht über die existierenden Schnittstellen angebunden werden können. Sie können, einmal konfiguriert, unter Plattform-Betreibern geteilt werden.

Arten von Konnektoren

intern vs. extern

- interne Konnektoren werden innerhalb der Umgebung von CIVITAS/CORE betrieben und haben Zugriff auf interne Komponenten
- externe Konnektoren können unabhängig von der Infrastruktur von CIVITAS/CORE betrieben werden. Sie haben ausschließlich Zugriff auf die öffentlichen Schnittstellen von CIVITAS/CORE

Push vs. Pull

- Push-Konnektoren kriegen die Daten, die sie einspielen geschickt. Sie sind passiv
- Pull-Konnektoren fragen aktiv Daten bei externen Systemen an, z.B. in kontinuierlichem zeitlichem Abstand

Schnittstellenliste

Geodaten
- OGC Web Services (WM(T)S, WFS, WCS)
- OGC API
- 3D Tiles
- Vector Tiles

Meta Daten
- CSW

Sensor Daten
- Sensor Things API
- FIWARE NGSI-LD
- PromQL, Prometheus Remote Write

generisch
- REST
- GraphQL
- MQTT (+Web Socket Transport)

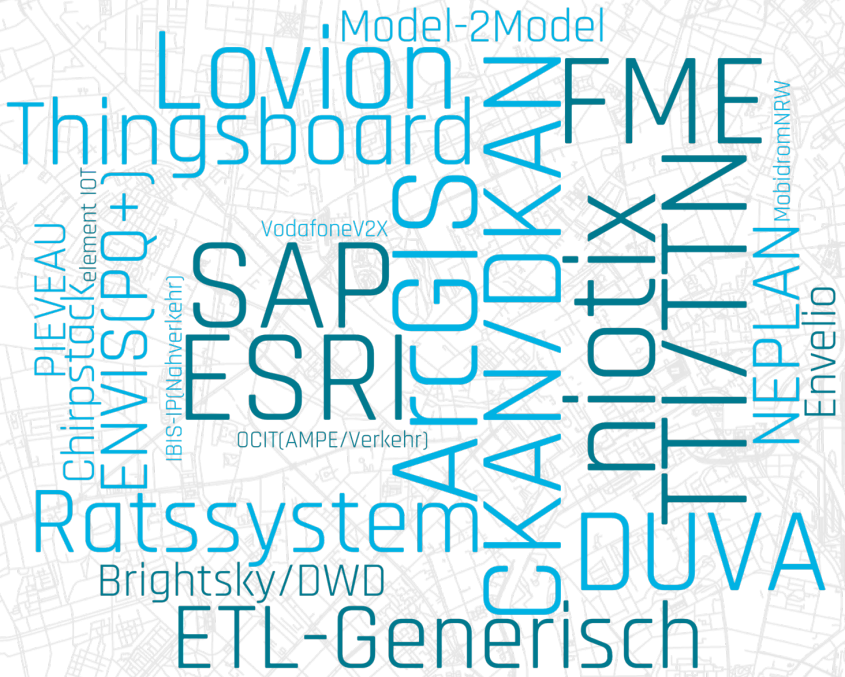
administrativ
- Data Management
- User Management

Diskussion:
- CDE-Schnittstellen im BIM-Kontext
- SOAP relevant?
- 3D-Daten - wie angebunden?
- Anbindung an BI-Tools/Daten
- XMLA, SPARQL

-> Schnittstellen sind im Nachhinein individuell durch Add-Ons erweiterbar

Kooperationspotenziale bei Konnektoren

Systeme und Produkte



Protokolle, Formate und Standards

Tech-Stack Version 2.0 (Release Mitte 2025)