

Das Energiewende-Dashboard

Ein Erfolgsbeispiel interkommunaler Kooperation

Partnerstädte:



Landeshauptstadt
München

Gefördert durch:



- Connected Urban Twins (CUT) – Urbane Datenplattformen und Digitale Zwillinge für integrierte Stadtentwicklung
- Smart-City-Modellprojekten des Bundesministeriums für Wohnen, Stadtentwicklung und Bauwesen (BMWSB)
- Hamburg, München, Leipzig
- Entwicklung Digitaler Zwillinge
- Open Source Paradigma



3D-Stadtmodelle von Hamburg, Leipzig und München

Als Sebastian in CUT angefangen hat...



Anforderungen zum Projektstart:

- Wir bauen replikationsfähige Softwareprodukte.
- Alles Open Source.
- Zum Projektende sollen die entwickelten Lösungen produktionsreif sein.
- Wir wollen „Was-wäre-wenn“-Szenarien und Digitale Zwillinge.

Ziel

Darstellung des aktuellen Ausbaus erneuerbarer Energien im Stadtgebiet, um die Fachämter des Stadtplanungsamtes, Referat Nachhaltigkeit und Klimaschutz, Amt für Umweltschutz bei der Planung der Energiewende zu unterstützen (mit Bezug zu Kommunalen Wärmeplanung, energetischen Quartierskonzepten etc.).

Zielgruppe

- Klimaschutzmanagende
- Stadtplanende

Datenquelle

Öffentlich und frei verfügbare Daten des Marktstammdatenregisters der Bundesnetzagentur.

Energiewende-Dashboard Leipzig

Das Energiewende-Dashboard richtet sich an Klimaschutzmanagende der Kommunen und zeigt die aktuelle installierte Leistung an Erneuerbarer Energie an. Solare Strahlungsenergie hat aufgrund seiner geringen Voraussetzungen an Flächen eine besondere Bedeutung für das Gelingen der Energiewende, daher analysieren wir den historischen Zubau und prognostizieren diesen bis ins Jahr 2030, um den Entscheidungsträgern und Entscheidungsträgerinnen aus Städten und Kommunen eine Datengrundlage anbieten zu können.

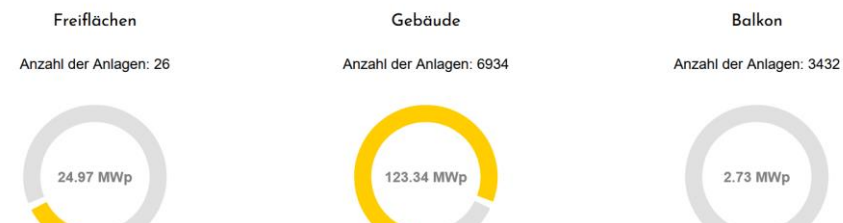
Aktueller Stand der erneuerbaren Energien

	Solare Strahlungsenergie	Windkraft	Biomasse	Wasserkraft
Installierte Leistung (MW(p)/a)	151.04	20.80	13.82	0.10
Jährliche Produktion (GWh/a)	155.22	26.69		



Solare Strahlungsenergie

Übersicht zur Lage der Photovoltaikanlagen



<https://leipzig.energiewende-dashboard.platforming.io/>

Die Entwicklungsgemeinschaft setzt auf Microservices und CI/CD.

Frontend

Frontend (React.js)

Backend

Backend in Python

Data Storage

PostgreSQL

Data Analysis Jobs

KPI Berechnung (Python)

Data Inbound Service

Datenintegration (Python, HTTPS-Webrequests)

MaStR

Externe API des Marktstammdatenregisters der Bundesnetzagentur

Entwicklung

Die Wartung und Weiterentwicklung des Energiewende-Dashboards wird durch die MPSC Community der Arbeits- und Entwicklungsgemeinschaft Energiedatenplattformen sichergestellt.

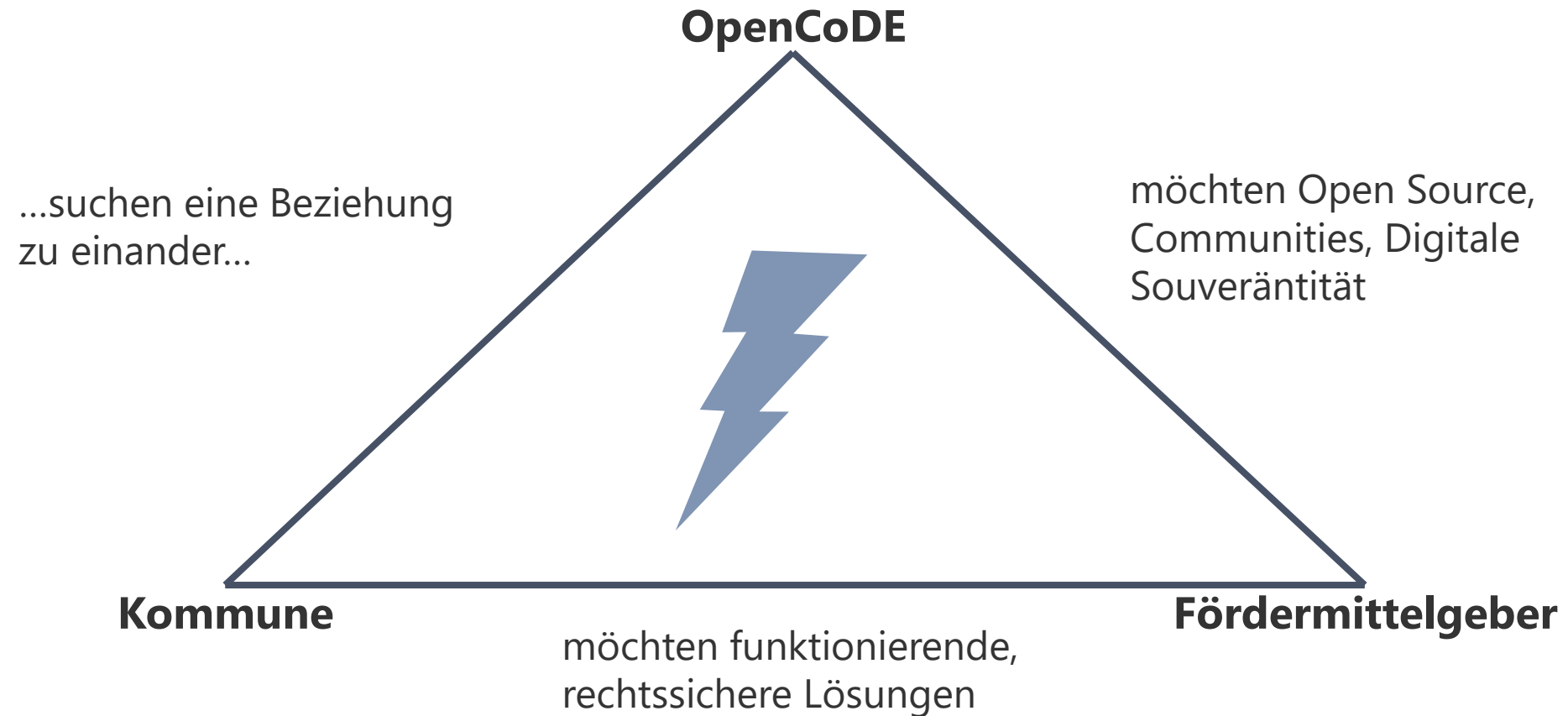
Meilensteine 2025

- Betriebskonzept für dauerhafte Nutzung
- Weiterentwicklung mit Fokus auf Fragestellungen aus Klimaschutz und Stadtplanung
- Veröffentlichung auf GitLab und openCode



<https://www.smart-city-dialog.de/raum/113/beitraege/11456>

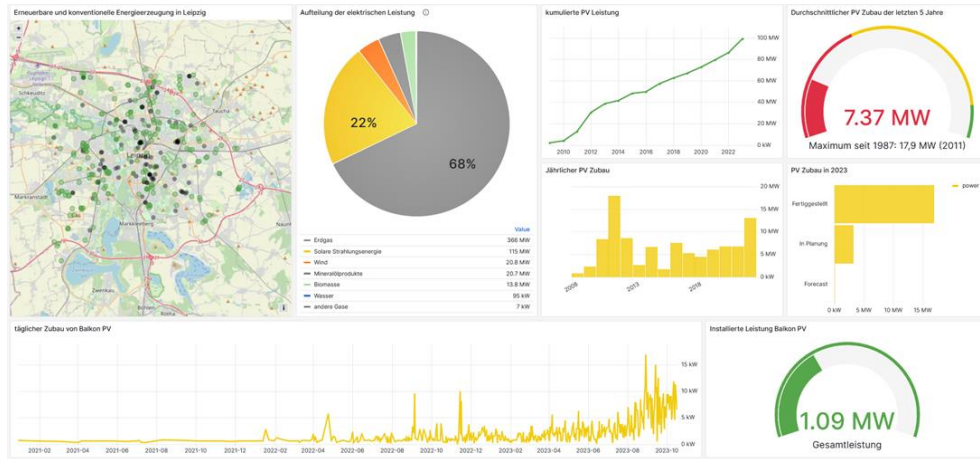
„Hätte ich damals gewusst, was ich heute weiß, hätte ich es nie getan.“



**Erste Schritte, nächste
Schritte**

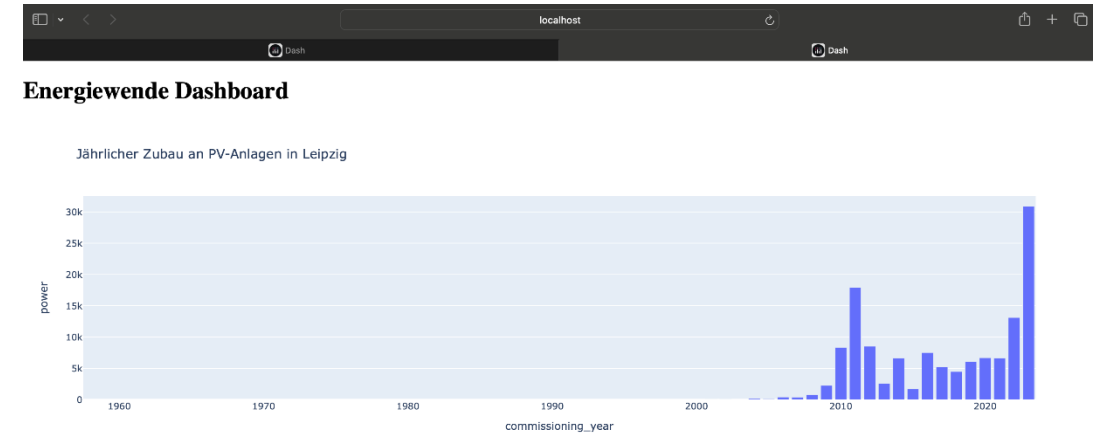
Entwicklung es Energiewende-Dashboards Teil 1

Juni 2023: PoC und erste Analysen in grafana



#Proof of Concept
#statische Daten
#100% Open Source
#1 Entwickler

Dezember 2023: erster PyDash PoC



#Replikationsreifes Softwareprodukt
#Technologischer Durchstich
#Datenaktualisierung

Entwicklung es Energiewende-Dashboards Teil 2

Juni 2024: erste PyDash Variante, dockerisiert, ins Internet exposed

Erreichung der Zubau-Ziele für erneuerbare Energien in Leipzig

Datenstichtag: 24.09.2024 erstellt vom **Datenlabor**, Referat Digitale Stadt

Zur aktiven Gestaltung der Energiewende im urbanen Raum hat sich die Stadt Leipzig das Ziel gegeben, von 2023 bis 2030 mindestens 400 MW regenerative Stromerzeugungskapazität zu installieren. Das wurde im "Energie- und Klimaschutzprogramm 2030" (EKSP 2030) vom 01.06.2023 beschlossen. Dieses Dashboard bietet einen Überblick über den aktuellen Stand und die Entwicklung der Stromerzeugung und Stromerzeugungskapazität der Stadt Leipzig.

Aktueller Stand der erneuerbaren Energien in Leipzig



Solare Strahlungsenergie: Historie und Prognose Übersicht zur Lage der Photovoltaikanlagen



#AgileEntwicklung
#TDD
#OOP
#2 Entwickler*innen

Dezember 2024: React-Version, Microservices, Python-Backend, Tests

Energiewende-Dashboard Leipzig

Das Energiewende-Dashboard richtet sich an Klimaschutzmanager der Kommunen und zeigt die aktuelle installierte Leistung an Erneuerbarer Energie an. Solare Strahlungsenergie hat aufgrund seiner geringen Voraussetzungen an Flächen eine besondere Bedeutung für das Gelingen der Energiewende, daher analysieren wir den historischen Zubau und prognostizieren diesen bis ins Jahr 2030, um den Entscheidungsträgern und Entscheidungsträgerinnen aus Städten und Kommunen eine Datengrundlage anbieten zu können.

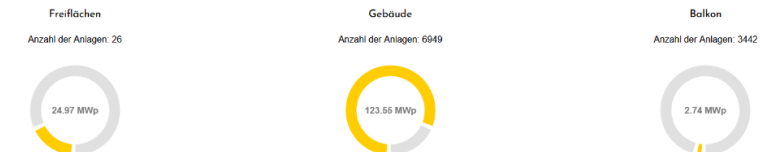
Aktueller Stand der erneuerbaren Energien

	Solare Strahlungsenergie	Windkraft	Biomasse	Wasserkraft
Installierte Leistung (MWp/a)	151.26	20.80	13.82	0.10
Jährliche Produktion (GWh/a)	155.44	26.69		

■ Solare Strahlungsenergie ■ Windkraft ■ Wasserkraft ■ Biomasse ■ Nicht-erneuerbare Energie

Solare Strahlungsenergie

Übersicht zur Lage der Photovoltaikanlagen



#Frontend
#Backend
#Data Jobs
#Datenbank

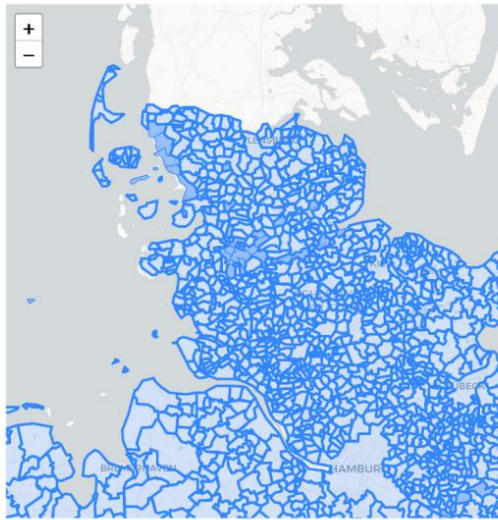
Entwürfe für die nächste Version

Energiewende-Dashboard Nordfriesland

Das Energiewende-Dashboard richtet sich an Klimaschutzmandate und Stadtplanende der Kommunen. Wir möchten Ihnen damit eine Datengrundlage für die Entscheidungsfindung im Ausbau der erneuerbaren Energien geben. Die Datengrundlage hierfür ist das [Marktstammdatenregister](#) der Bundesnetzagentur. Diese Daten sind öffentlich verfügbar.

Status Quo Vorhersage Simulation

Aktueller Stand und historische Zubauzahlen der erneuerbaren Energien

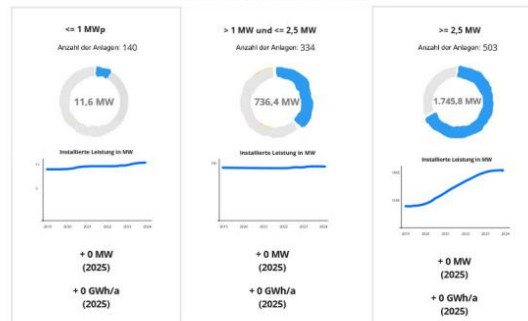


	Solare Strahlungsenergie	Windkraft	Biomasse	Wasserkraft
Installierte Leistung (MW(p))	512,89	2.493,71	113,59	0,00
Jährliche Produktion (GWh/a)	527,06	3.199,51		



Solare Strahlungsenergie Windkraft Wasserkraft Biomasse

Übersicht zur Größenordnung der Anlagen



erstellt am 02.01.2025



Energiewende-Dashboard Leipzig

Das Energiewende-Dashboard richtet sich an Klimaschutzmandate und Stadtplanende der Kommunen. Wir möchten Ihnen damit eine Datengrundlage für die Entscheidungsfindung im Ausbau der erneuerbaren Energien geben. Die Datengrundlage hierfür ist das [Marktstammdatenregister](#) der Bundesnetzagentur. Diese Daten sind öffentlich verfügbar.

Status Quo Vorhersage Simulation

Aktueller Stand und historische Zubauzahlen der erneuerbaren Energien



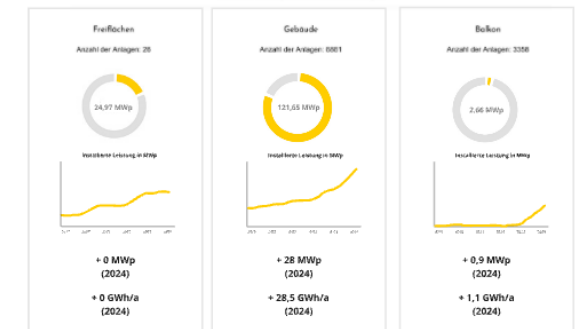
	Solare Strahlungsenergie	Windkraft	Biomasse	Wasserkraft
Installierte Leistung (MW(p))	148,25	29,30	13,82	0,10
Jährliche Produktion (GWh/a)	113,40	28,89		



Solare Strahlungsenergie Windkraft Wasserkraft Biomasse

Verteilung der nach Lage, Einspeisung, Größenordnung

Übersicht zur Lage der Photovoltaikanlagen



erstellt am 12.11.2024

- Trotz organisatorischer **Herausforderungen** aus dem Ökosystem:
Das Dashboard ist nur so gut wie es ist, weil es in einer Community entwickelt wurde.
- Die Community zieht die guten Leute an, weil sie einen Raum zum Lernen und für Weiterentwicklung bietet.
- Die Community zwingt einen zu professionellen Arbeitsweisen und bringt Methoden mit sich, die man nutzen muss, damit man überhaupt eine sinnvolle Lösung entwickeln kann.
- Software-Qualität und moderne IT (Microservices, Kubernetes, TDD, ...) sind kein Selbstzweck. Sie stellen sicher, dass die Lösung langfristig funktionieren kann
- Jedes Feedback verbessert die Lösung. Replikation generiert Feedback.

Wie arbeitet unsere Community zusammen?

#Nützlichkeit

#Verfügbarkeit

#NoExcuses

```

1 pv_only = MaStReg.query('source == "Solare Strahlungsenergie"')
2 print('Number of photovoltaic assets (all status) in Leipzig: ' + str(pv_only.shape[0]))
3 print(pv_only.groupby('status').agg({'status':'count', 'power':{'sum', 'mean', 'median'}}))
4
5 # filter only for assets which are "In Betrieb", because we want to work with the assets which are producing now
6 pv_only_active = pv_only.query('status == "In Betrieb"')
7 pv_annual_addition = pv_only_active.groupby('commissioning_year').agg(['power':'sum'])
8
9 pv_annual_addition['cum_sum'] = pv_annual_addition['power'].cumsum()
10 pv_daily_addition = pv_only_active.groupby('commissioning_date').agg({'power':'sum'})
11

```

Kein Spaghetti-Code

```

1 pv_annual_addition['returns'] = pv_annual_addition['power'].pct_change()
2 # pv_annual_addition['returns'].hist()
3 print('--- key statistics on annual pv addition returns ---')
4 print(pv_annual_addition['returns'].agg([np.mean, np.median]))
5 print('25%-quantile ' + str(pv_annual_addition['returns'].quantile(0.25)))
6 print('25%-quantile ' + str(pv_annual_addition['returns'].quantile(0.75)))
7
8 print('all time HIGH annual addition: ' + str(pv_annual_addition['power'].max()))
9 print('all time LOW annual addition: ' + str(pv_annual_addition['power'].min()))
10 print('annual addition of the last 5 years: ' + str(pv_annual_addition['power'].tail(5)))
11 print('5 year mean: ' + str(np.mean(pv_annual_addition.loc[2018:2022, 'power'])))
12
13 five_yr_mean = np.mean(pv_annual_addition.loc[2018:2022, 'returns'])
14 print('5 yr mean (returns): ' + str(five_yr_mean))
15 print('5 yr median (returns): ' + str(np.median(pv_annual_addition.loc[2018:2022, 'returns'])))

```

- Prinzipien aus der OOP helfen dabei die Komplexität zu managen
- Anstatt langer, umständlicher Abfrage-Konstrukte oder Modelle kann man die Aufgaben in Klassenmethoden kapseln
- Vorteile:
 - einheitliche Schnittstellen
 - gut zu testen

```
### Asset Class
import pandas as pd

from datetime import datetime

class Asset:
    """defines a class for general energy asset"""
> def __init__(self, max_power, status, commissioning_date): ...

    # Add a decorated max_power() method returning _power
    @property
> def max_power(self): ...

    # Add a setter max_power() method
    @max_power.setter
> def max_power(self, new_max_power): ...

    # Add a decorated status() method returning _status
    @property
> def status(self): ...

    # Add a setter status() method
    @status.setter
> def status(self, new_status): ...

    # Add a decorated is_active() attribute
    @property
> def is_active(self): ...
    @property
> def commissioning_date(self): ...
```


Kein Weg führt an Tests vorbei



- Automatisches Testen sichert langfristig die Funktionalität und Qualität des Codes
- Anfangs dauert es länger, aber im Laufe der Zeit gibt es euch Geschwindigkeit

```
===== test session starts =====
platform win32 -- Python 3.11.1, pytest-8.1.1, pluggy-1.4.0
rootdir: C:\Users\BoehmSe03\Analyse_Tools\Code\energiewende-dashboard
plugins: dash-2.16.1
collected 40 items

tests\asset_modelling\test_asset_modelling.py ..... [ 97%]
tests\test_main.py x [100%]

===== warnings summary =====
tests/asset_modelling/test_asset_modelling.py::test_AssetPortfolio_annual_growth_path_nonempty_portfolio
  C:\Users\BoehmSe03\Analyse_Tools\Code\energiewende-dashboard\tests\asset_modelling\test_asset_modelling.py:519: FutureWarning: The 'kind' keyword in DataFrame.resample is deprecated and will be removed in a future version. Explicitly cast the index to the desired type instead
    expected_result = helper_df.resample('YE', on = 'commissioning_date', kind = 'period').sum()

tests/asset_modelling/test_asset_modelling.py::test_AssetPortfolio_annual_growth_path_nonempty_portfolio
  C:\Users\BoehmSe03\Analyse_Tools\Code\energiewende-dashboard\src\asset_modelling\asset_modelling.py:251: FutureWarning: The 'kind' keyword in DataFrame.resample is deprecated and will be removed in a future version. Explicitly cast the index to the desired type instead
    df = df.resample('YE', on = "commissioning_date", kind='period').sum()

-- Docs: https://docs.pytest.org/en/stable/how-to/capture-warnings.html
===== 39 passed, 1 xfailed, 2 warnings in 4.56s =====
```

Wie wir Tests in die Entwicklung einbinden



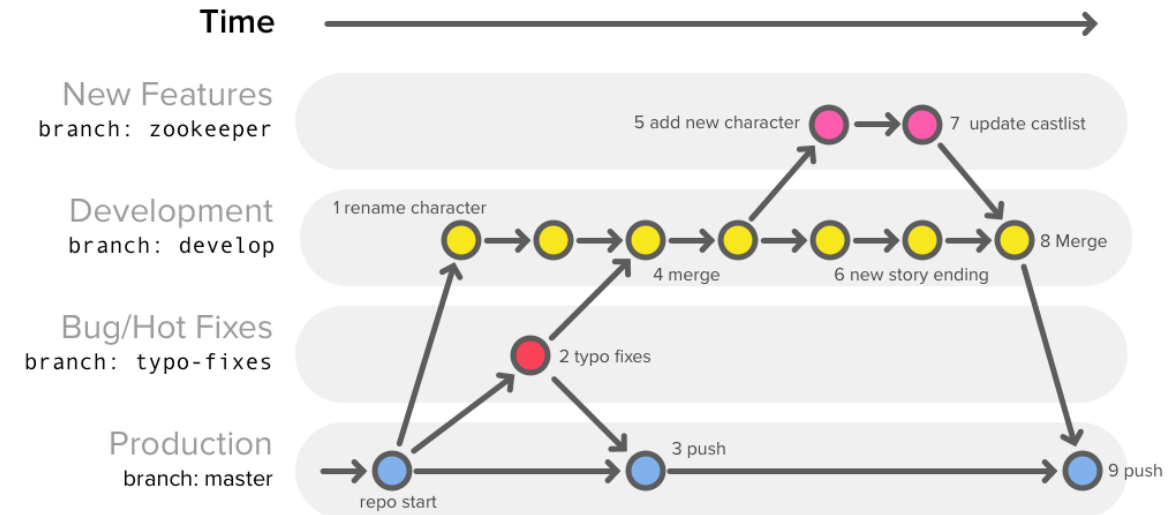
- Wir nutzen TDD
- TDD ... test driven development
- Zuerst Testfälle definieren
- Dann diese implementieren
- Dann Code entwickeln
- Nach jeder Änderung die Tests laufen lassen

```
20
27 ✓ def test_Asset_init_error_with_negative_max_power():
28     """negative max_power input raises a ValueError"""
29
30     # define test inputs
31     invalid_max_power = -1
32     valid_status = 'In Betrieb'
33     valid_commissioning_date = datetime(2024, 2, 28)
34
35     # define expected error message
36     expected_error_msg = "Power can not be negative for this asset type."
37
38     # if Asset raises a ValueError then store exception as exception_info
39 ✓ with pytest.raises(ValueError) as exception_info:
40     |     test_asset = Asset(invalid_max_power, valid_status, valid_commissioning_date)
41
42     # check if ValueError contains correct message
43     assert exception_info.match(expected_error_msg)
44
45
```

„Keine Versionskontrolle – kein Mitleid“



- Eine Versionskontrolle ermöglicht das kollaborative Arbeiten an einem Projekt
- Alle Änderungen werden nachvollziehbar verwaltet
- Durch das aufteilen der Entwicklung auf verschiedene Branches wird sichergestellt, dass nur funktionstüchtige Software ausgeliefert wird



Continuous Integration und Continuous Deployment



- CI / CD stellt sicher, dass die Software kontinuierlich und standardisiert ausgeliefert wird
- In einer Pipeline werden die einzelnen Schritte definiert, die für die Auslieferung notwendig sind
- Zum Beispiel sollte es immer auch einen Test-Schritt geben

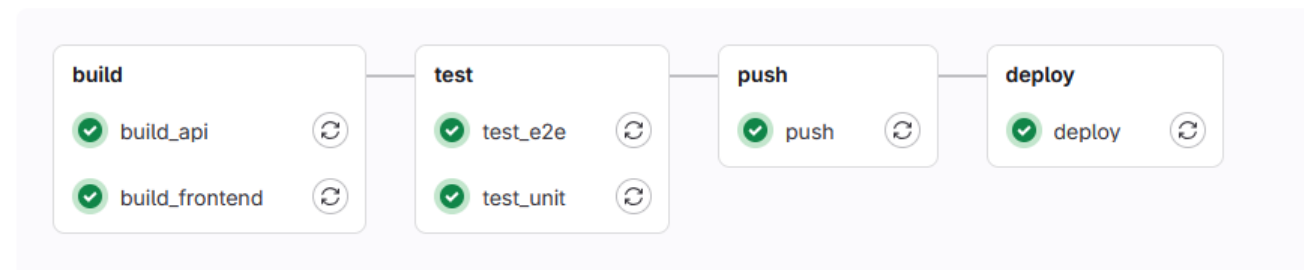
Merge branch 'ED-131' into 'main'

✓ Passed Sebastian Böhm created pipeline for commit `b58dbb36` 5 days ago, finished 5 days ago

For `main`

latest 60 6 jobs ⌚ 4 minutes 37 seconds, queued for 3 seconds

Pipeline Jobs 6 Tests 0

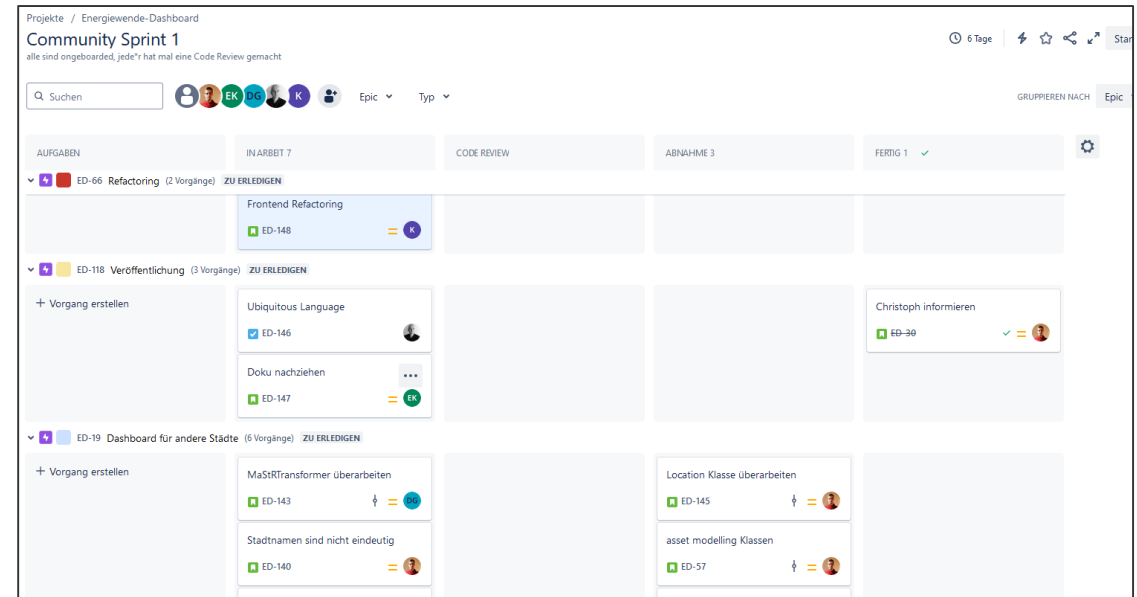


- Container stellen sicher, dass der Code überall läuft – unabhängig von Betriebssystem, lokal installierten Paketen usw.
- Außerdem sind sie eine zusätzliche Sicherheitsschicht im Rechenzentrum

```
docker-compose.yml 1.15 KiB
1  services:
2    backend:
3      build: .
4      environment:
5        - DATABASE_URL=postgresql://user:password@db:5432/energiewende_db;
6        - PAGE_SIZE=25000 # size of buckets for data download
7      ports:
8        - "8080:8000"
9      healthcheck:
10        test: ["CMD-SHELL", "curl http://localhost:8000/db_health"]
11        interval: 10s
12        timeout: 5s
13        retries: 5
14        start_period: 10s
15      depends_on:
16        db:
17          condition: service_healthy
18      links:
19        - db
20
21    frontend:
22      build:
23        context: frontend
24      args:
25        REACT_APP_API_BASE_URI: http://localhost:8080
26      ports:
27        - "5001:80"
28      depends_on:
29        backend:
30          condition: service_healthy
31      links:
32        - db
33
34
35    db:
36      image: postgres:13
37      environment:
```

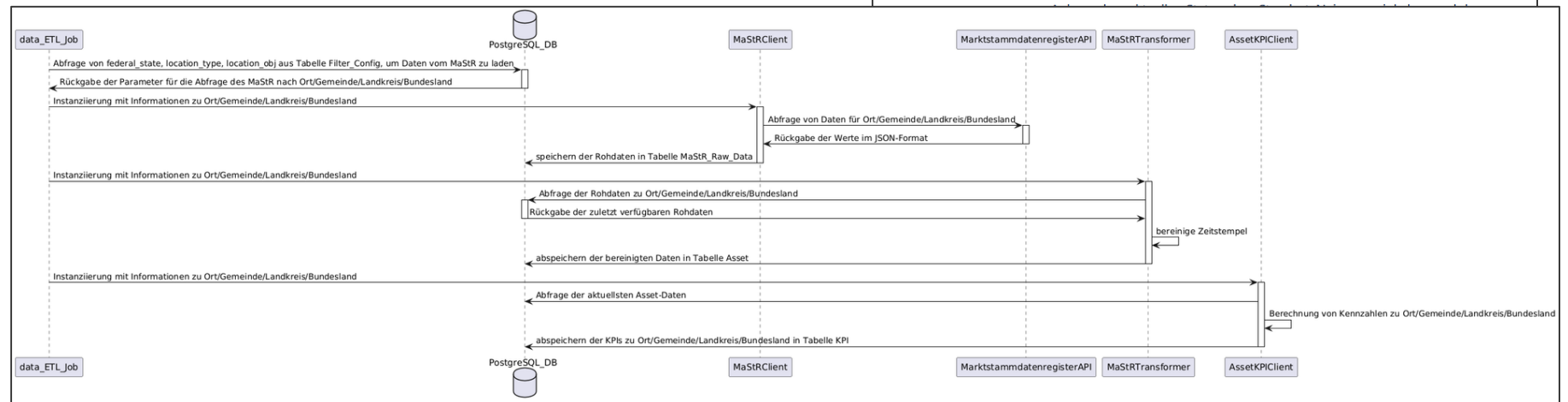
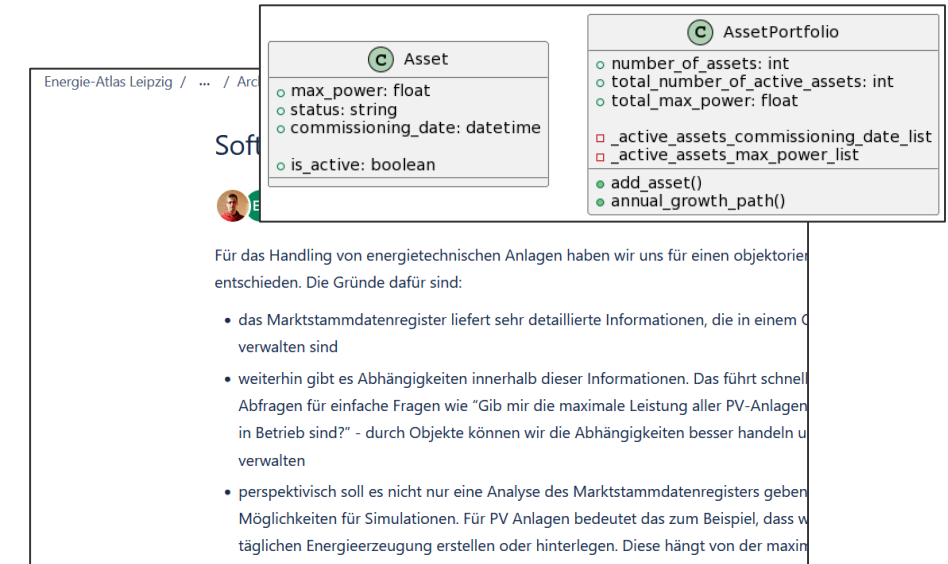
Ein fokussierter Arbeitsprozess

- Wir arbeiten agil in 4 Wochen Sprints
- Das Planning findet gemeinsam statt
- Wir nutzen die üblichen Kommunikationstools
- Die Arbeit ist weitestgehend asynchron organisiert (guter Guide: <https://handbook.gitlab.com/handbook/company/culture/all-remote/asynchronous/>)



Eine schlanke Doku erleichtert das Leben

- Wir wollen unser Wissen teilen, um langfristig unseren Code weiterentwickeln zu können
- Wir dokumentieren unsere Entwicklungsschritte direkt am Ticket
- Architektur-Entscheidungen werden als ADR (Architectural Decision Record) festgehalten



Verfügbar auf GitLab und OpenCoDE



Energiewende Dashboard

main energiewende-dashboard /

Find file Edit Code

ci: add pipeline Jan Beckert authored 1 month ago

5e253a86 History

Name	Last commit	Last update
frontend	inital commit	1 month ago
migrations	inital commit	1 month ago
src	inital commit	1 month ago
tests	inital commit	1 month ago
.dockerignore	inital commit	1 month ago
.gitignore	inital commit	1 month ago
.gitlab-ci.yml	ci: add pipeline	1 month ago
.pre-commit-config.yaml	inital commit	1 month ago
Contributing.md	inital commit	1 month ago

Project information

- 1 Commit
- 2 Branches
- 0 Tags
- 393 KIB Project Storage
- README
- MIT License
- CONTRIBUTING
- CI/CD configuration
- + Add CHANGELOG
- + Add Kubernetes cluster
- + Add Wiki
- + Configure Integrations

Created on January 27, 2025

<https://gitlab.com/entwicklungscommunity-energiedaten/energiewende-dashboard>

cut-energiewende-dashboard

main cut-energiewende-dashboard

Datei finden Code

ci: add pipeline Jan Beckert erstellt vor 1 Monat

5e253a86 Verlauf

Name	Letzter Commit	Letzte Aktualisierung
frontend	inital commit	vor 1 Monat
migrations	inital commit	vor 1 Monat
src	inital commit	vor 1 Monat
tests	inital commit	vor 1 Monat
.dockerignore	inital commit	vor 1 Monat
.gitignore	inital commit	vor 1 Monat
.gitlab-ci.yml	ci: add pipeline	vor 1 Monat
.pre-commit-config.yaml	inital commit	vor 1 Monat
Contributing.md	inital commit	vor 1 Monat
Dockerfile	inital commit	vor 1 Monat

Projektinformation

- 2 Commits
- 2 Branches
- 0 Tags
- README
- MIT License
- CONTRIBUTING

Erstellt am February 24, 2025

<https://gitlab.opencode.de/connected-urban-twins/cut-energiewende-dashboard>

- Fördermittel enden, aber gute Produkte nicht
- Die Entwicklungscommunity lebt weiter und arbeitet an ihrer Vision die passenden Werkzeuge für die Energiewende und eine nachhaltige Zukunft zu gestalten
- Du willst mitmachen?
Dann sprich uns einfach an 😊



STADT REGENSBURG

Digitaler Energie-Zwilling

Das Energiewende-Dashboard:
Dein Code für unsere Zukunft

Open Source. Agile Community. Klimaneutrale Städte

Werde Teil der Community und entwickle das Energiewende-Dashboard gemeinsam mit uns weiter.

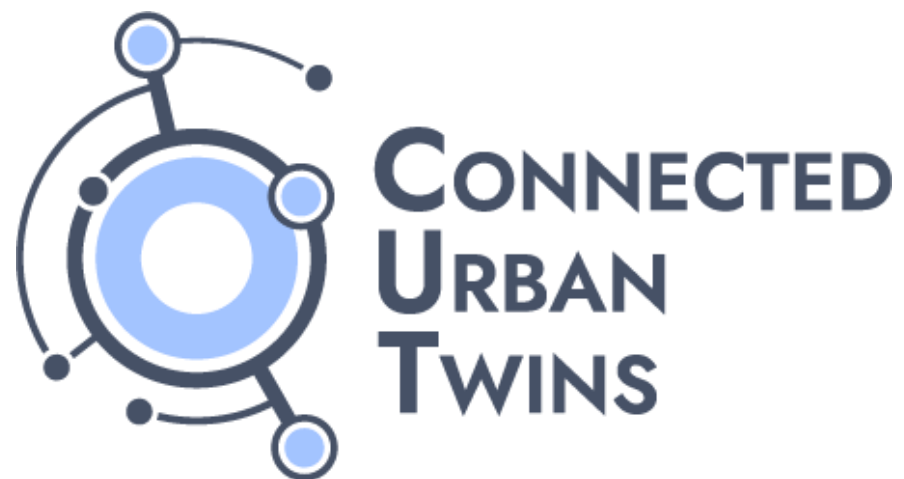
- 🔧 Entwickle mit uns in der Open-Source-Community
- 🌍 Realer Impact für Klimaschutz und Stadtplanung
- 🚀 4-Wochen-Sprints. Keine Bürokratie. Viel Code.

👉 Jetzt einsteigen: **Stand R_Next Regensburg**



R_next Digitaler Energiezwilling

Gefördert durch:
 Bundesministerium für Klimaschutz, Umwelt, Energie und Bauwesen
 KfW
mitgliedschaft im Bundesministerium für Klimaschutz, Umwelt, Energie und Bauwesen



Partnerstädte:



Gefördert durch:



Vielen Dank!