HELMHOLTZ ZENTRUM
DRESDEN ROSSENDORF

Erstellung einer wiederverwendbaren Messinfrastruktur an der HAMSTER-Forschungsanlage

HZDR · Forschungstechnik · Jonas Gorgis · www.hzdr.de

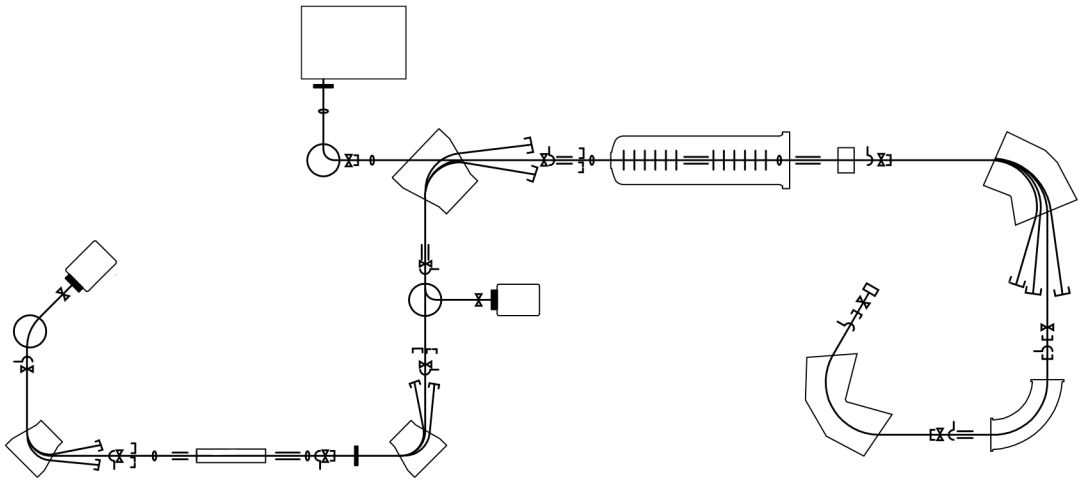
Motivation/Gliederung

1) HAMSTER-Forschungsanlage
2) Kontroll-/Messsystemsoftware
3) Benutzeroberfläche
4) Deployment / Wiederverwendbarkeit

HAMSTER Beamline

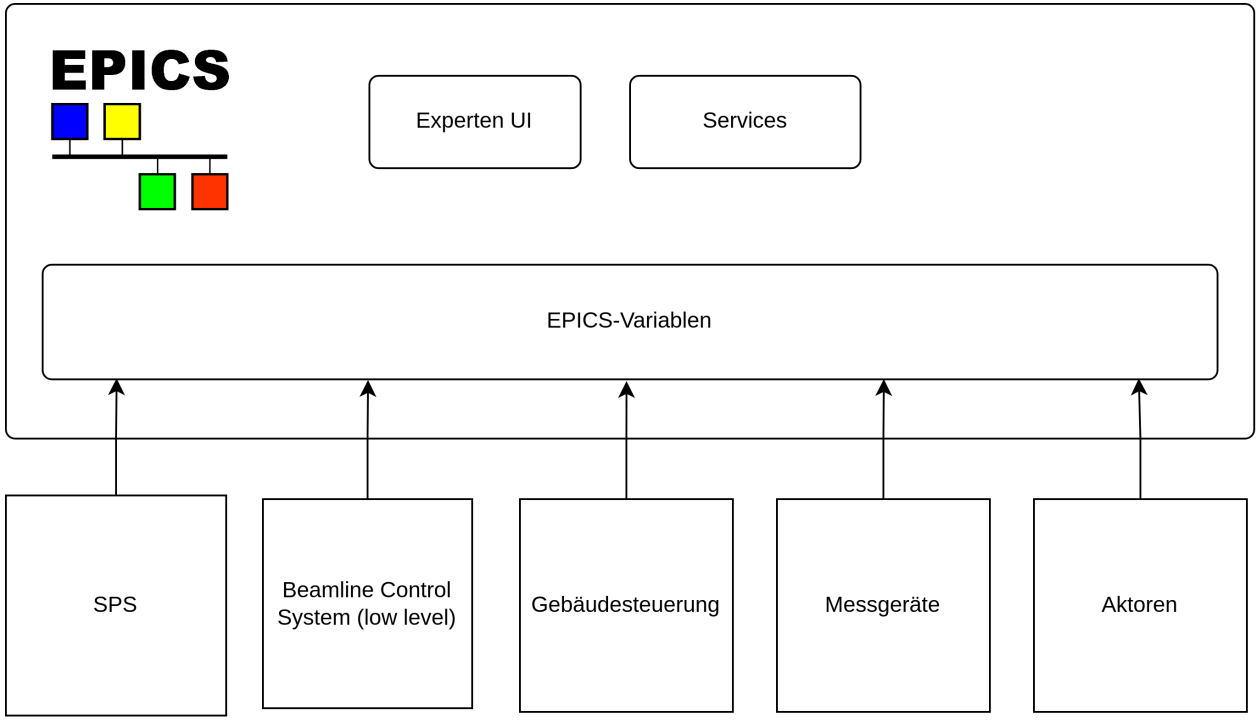
Helmholtz **A**ccelerator **M**ass **S**pectrometer **T**racing **E**nvironmental **R**adionuclides

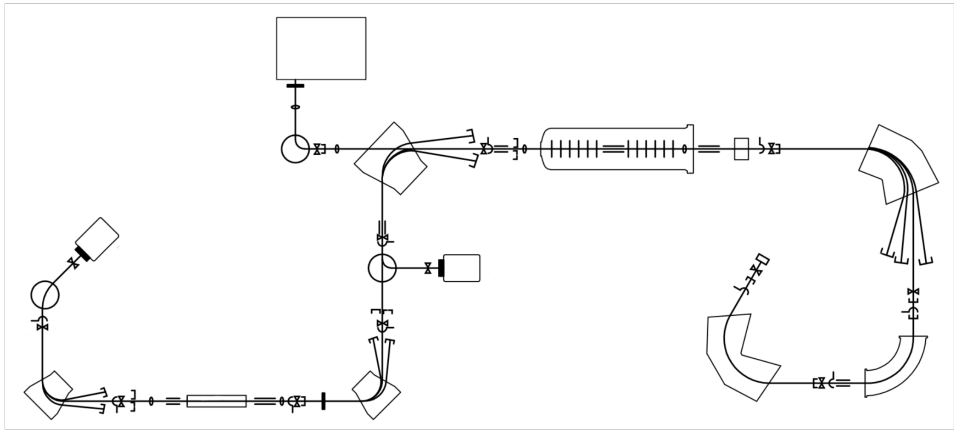
- Beschleuniger-Massenspektrometer
- Hauptbestandteile:
  - 2 Ionenquellen
  - SIMS (Quelle oder unabhängige Messungen)
  - Ionenkühler
  - Beschleuniger (1-MV Pelletron)
  - Detektor
- Erkennung von Radionukliden in geologischen Proben z.B. Al-26, Be-10, Ca-41, Cl-36



HAMSTER EPICS

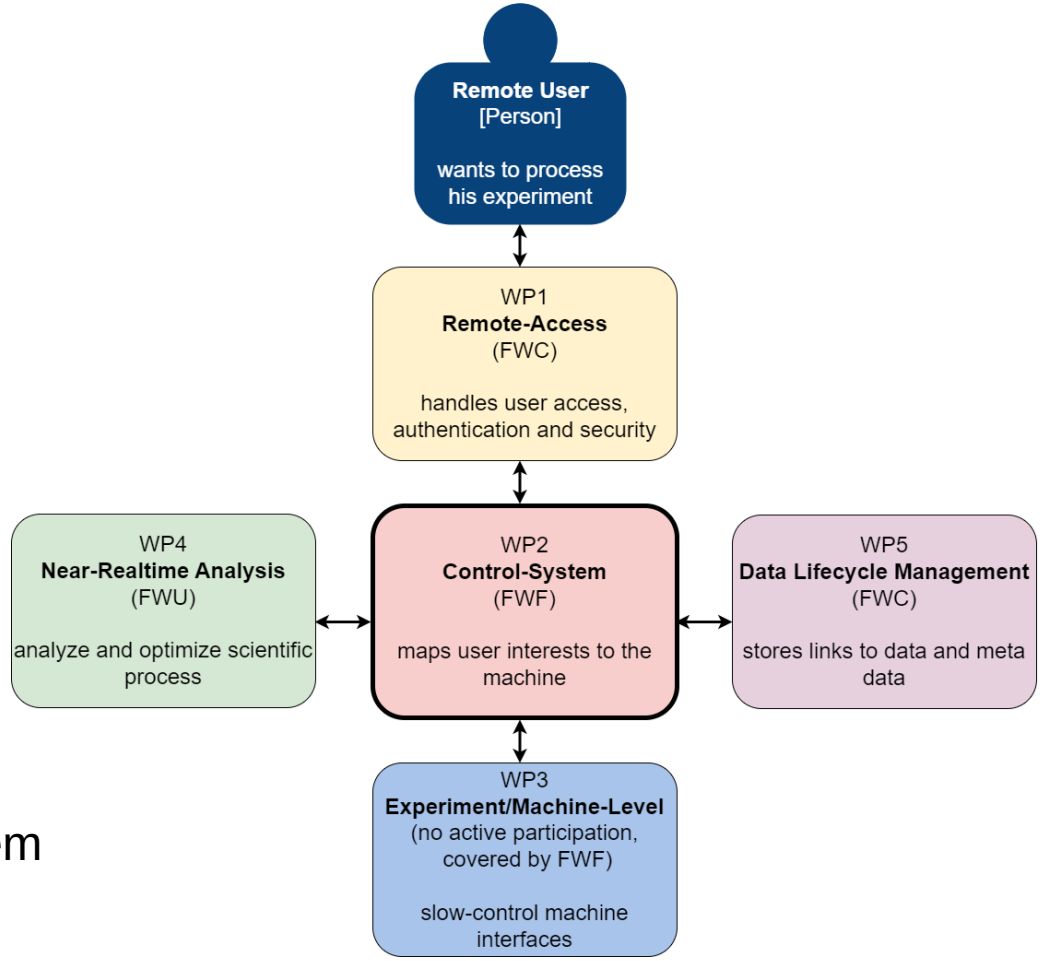
- Einheitliche Schnittstelle zur Hardware
- Parameter pro Variable z.B. Limits, Einheiten
- Flache Hierarchie der Variablen
- Variablenstruktur durch Namenskonvention
- Diverse Services z.B. Archivierung des Anlagenzustands, Alarmserver
- Experten UI mit EPICS Qt





Projekt Rock-IT

- Softwareprojekt
- **Ziele**:
  - Hochautomatisierte Messungen
  - Remote bedienbar
  - Live Analyse und Optimierung mit KI
  - Management des Datenlebenszyklus
- Beteiligte Zentren: KIT, HZB, DESY und HZDR
- HZDR Forschungstechnik: Fokus auf Kontrollsystem



ROCK-IT

DESY. KIT Karlsruher Institut für Technologie HZB Helmholtz Zentrum Berlin

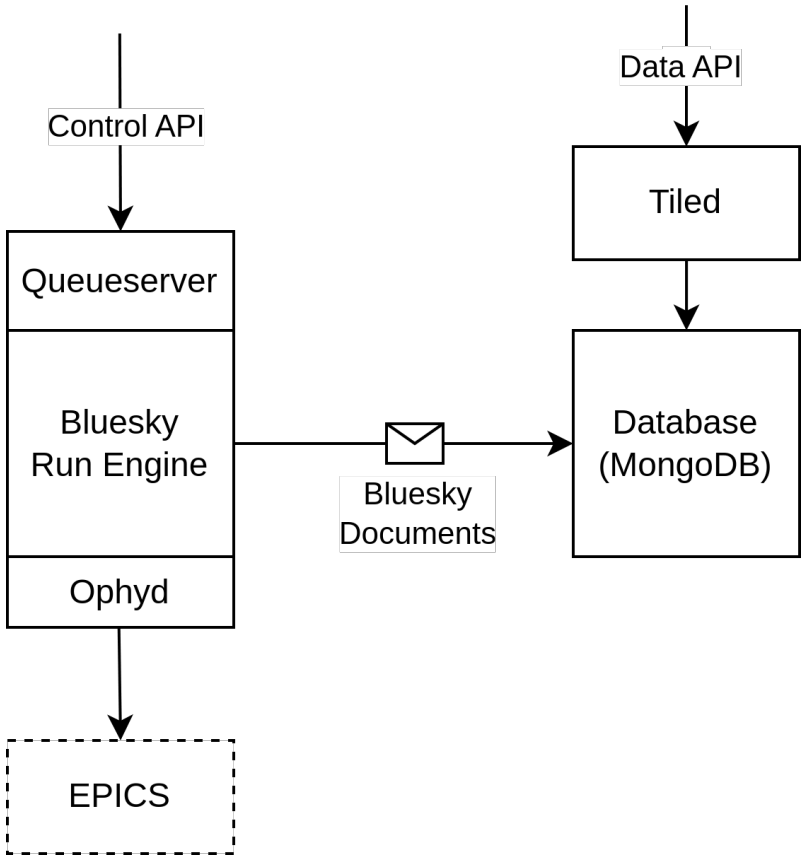
Bluesky

Software zur Steuerung von wissenschaftlichen Experimenten und Aufnahme von Mess- und Metadaten

- Open-Source
- Python-basiert
- Aktive Community (z.B. Mattermost)
- Strukturierung der Daten in Bluesky Documents
- Implementierungen für typische Abläufe und Geräte vorhanden
- Gute Erweiterbarkeit/Anpassbarkeit

Bluesky Services

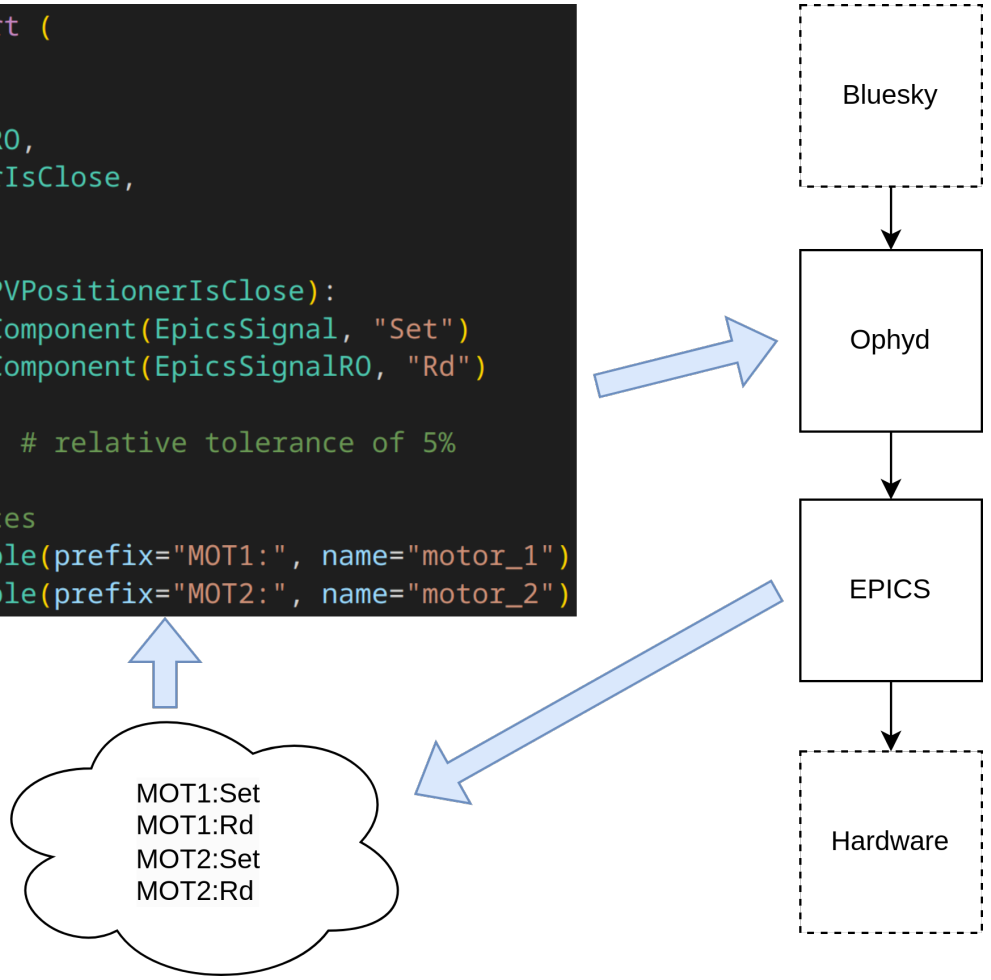
- Ökosystem von Services und Bibliotheken
  - **Run Engine**: Durchführung von Messungen
  - **Ophyd**: Hardwareabstraktion
  - **Tiled**: API zum Zugriff auf Messdaten
  - **Queueserver**: API, Plan-Warteschlange, Nutzer und Gruppen Berechtigungen
- Grafische Benutzeroberfläche nur zum Teil vorhanden



Hardwareabstraktion mit Ophyd

- Anbindung der Hardware an Bluesky
- Anbindung anderer Kontrollsysteme (z.B. Tango) möglich
- Gruppieren von EPICS Variablen in Geräten
- Zusätzliche Funktionen z.B.:
  - Warten bis Sollwert erreicht
  - Erstellung einer Gerätehierarchie
- Einheitliches EPICS Namensschema hilfreich

```python
from ophyd import (
    Component,
    EpicsSignal,
    EpicsSignalRO,
    PVPositionerIsClose,
)

class Writable(PVPositionerIsClose):
    setpoint = Component(EpicsSignal, "Set")
    readback = Component(EpicsSignalRO, "Rd")

    rtol = 0.05  # relative tolerance of 5%

# Device instances
motor_1 = Writable(prefix="MOT1:", name="motor_1")
motor_2 = Writable(prefix="MOT2:", name="motor_2")
```



Pläne in Bluesky

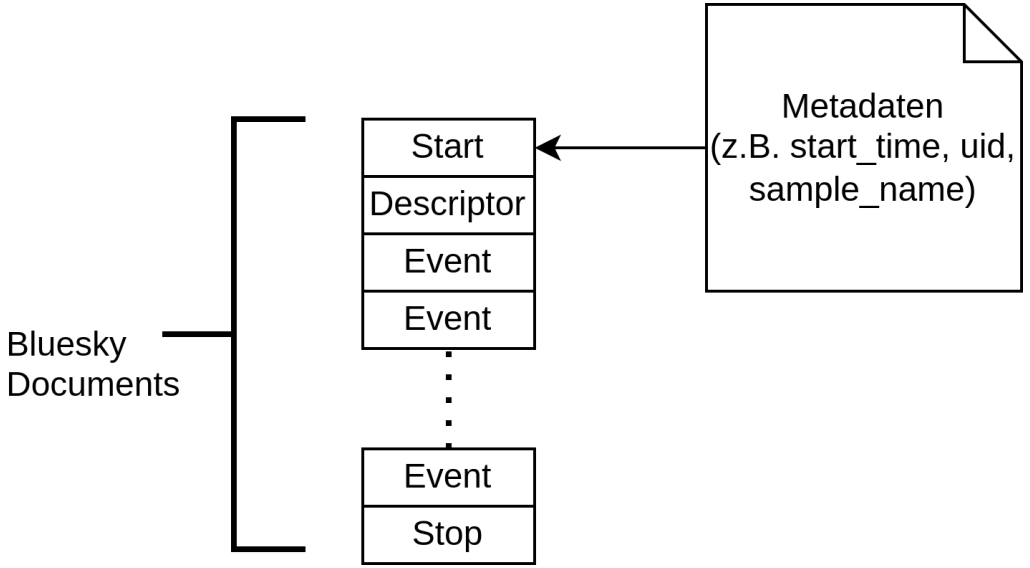
- Abläufe und Messungen
- Generisch oder Spezifisch
- „Python Skript“

```python
from typing import Sequence
import bluesky.plan_stubs as bps
from bluesky.protocols import Readable

def one_run_one_event(detectors: Sequence[Readable]):
    # Declare the beginning of a new run.
    yield from bps.open_run()

    # Trigger each detector and wait for triggering to complete.
    # Then read the detectors and bundle these readings into an Event
    yield from bps.trigger_and_read(detectors)

    # Declare the end of the run.
    yield from bps.close_run()
```

- Pläne am HAMSTER:
  - Hoch- und Herunterfahren der Ionenquelle
  - Optimierung
  - Messungen

Messdaten und Metadaten in Bluesky

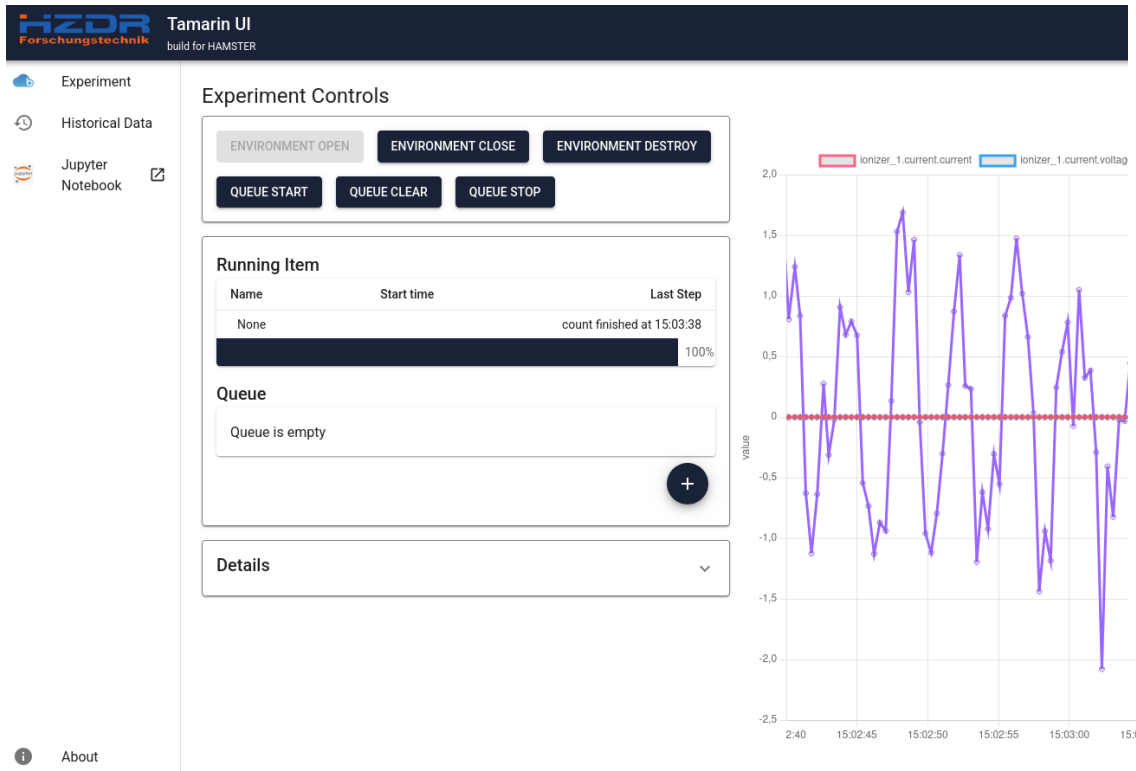
- Verknüpfung von Mess- und Metadaten
- Strukturierte Ablage in Bluesky Documents
- Große Datenmengen als Referenz in Documents
- **Typen**:
  - Start
  - Descriptor
  - Event
  - Stop

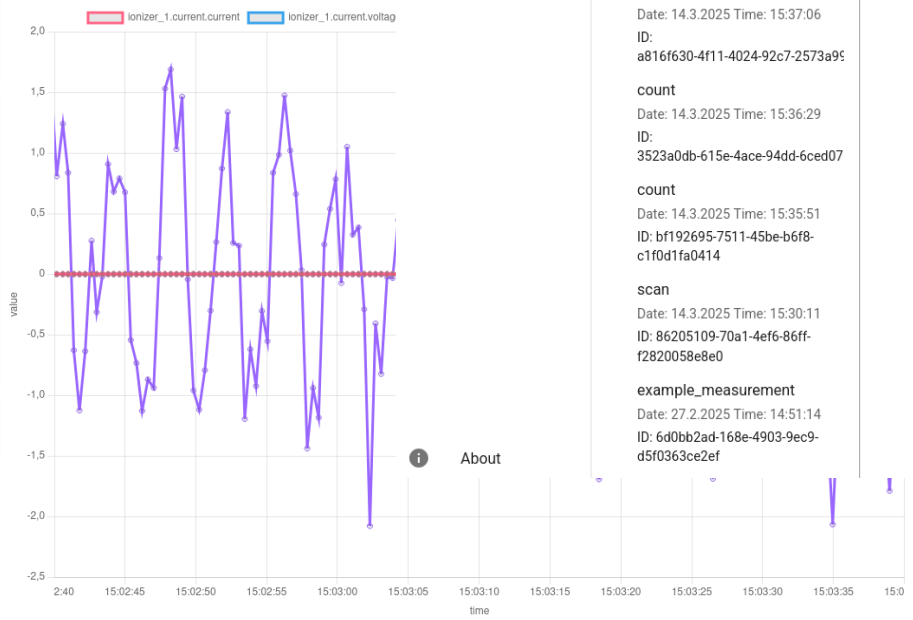


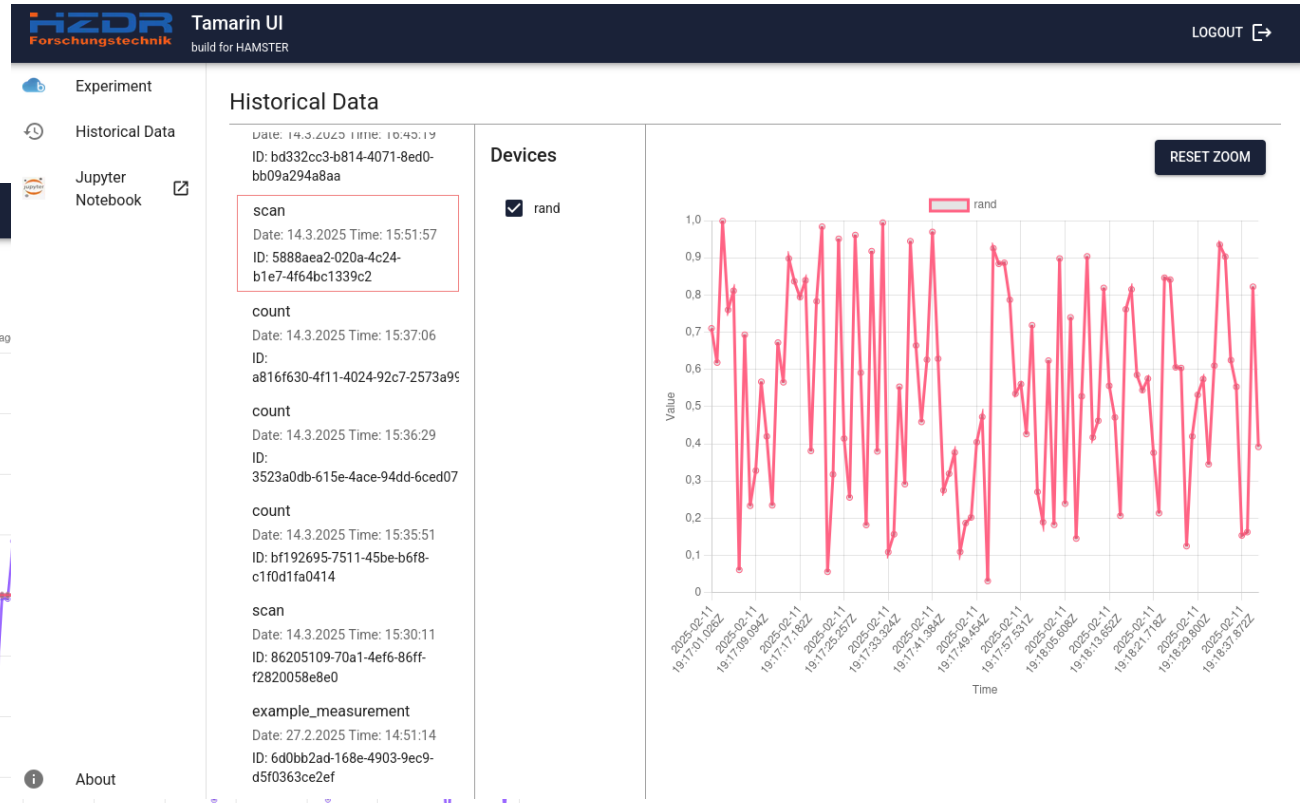
Abwägung Eigenentwicklung Benutzeroberfläche

	Eigenentwicklung Webbasiert	**Integration in bestehendes Webbasiertes UI**	**Qt-Anwendung**
Initialer Entwicklungsaufwand	0	-	+
Benutzerdefinierte Features	++	-	+
Erforderlich Kenntnisse der Entwickler	Webtechnologie (React/Typescript)	Spezialwissen, Webtechnologie, Python	Qt, Python, (C++)
Remotezugriff	++	++	-
Wartungsaufwand	-	0	0

Tamarin UI


HZDR Forschungstechnik
Tamarin UI
build for HAMSTER
Experiment
Historical Data
Jupyter Notebook
About
Experiment Controls
ENVIRONMENT OPEN
ENVIRONMENT CLOSE
ENVIRONMENT DESTROY
QUEUE START
QUEUE CLEAR
QUEUE STOP
Running Item
Name
Start time
Last Step
None
count finished at 15:03:38
100%
Queue
Queue is empty
Details


Ionizer_1.current.current
Ionizer_1.current.voltag
value
time


HZDR Forschungstechnik
Tamarin UI
build for HAMSTER
LOGOUT
Experiment
Historical Data
Jupyter Notebook
About
Historical Data
ID: bd332cc3-b814-4071-8ed0-bb09a294a8aa
scan
Date: 14.3.2025 Time: 15:51:57
ID: 5888aea2-020a-4c24-b1e7-4f64bc1339c2
count
Date: 14.3.2025 Time: 15:37:06
ID: a816f630-4f11-4024-92c7-2573a9
count
Date: 14.3.2025 Time: 15:36:29
ID: 3523a0db-615e-4ace-94dd-6ced07
count
Date: 14.3.2025 Time: 15:35:51
ID: bf192695-7511-45be-b6f8-c1f0d1fa0414
scan
Date: 14.3.2025 Time: 15:30:11
ID: 86205109-70a1-4ef6-86ff-f2820058e8e0
example_measurement
Date: 27.2.2025 Time: 14:51:14
ID: 6d0bb2ad-168e-4903-9ec9-d5f0363ce2ef
Devices
rand
RESET ZOOM
rand
Value
Time

Tamarin UI

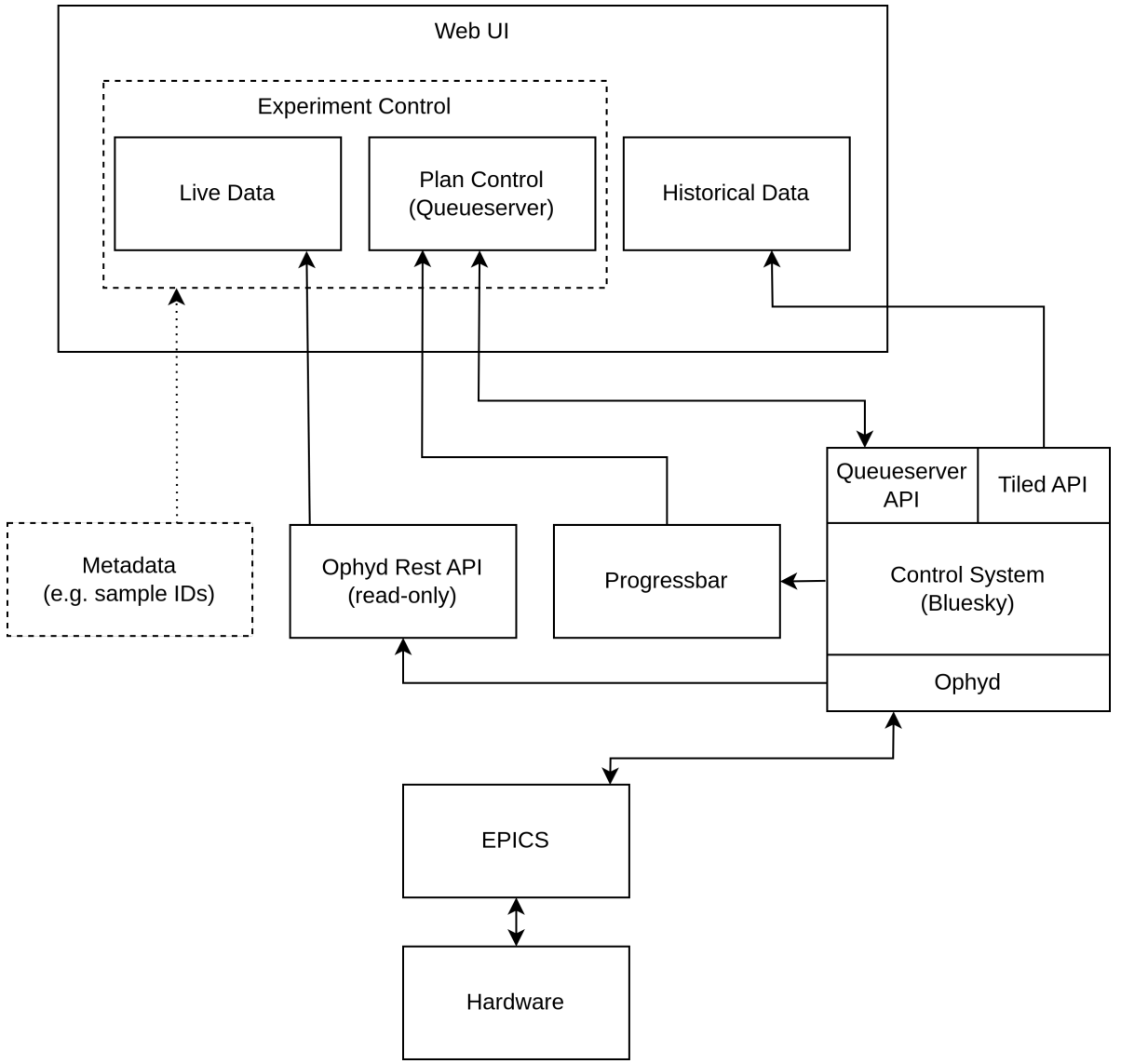
- Flexibler Ansatz
- Viel Arbeit in den Details

Technologien

- TypeScript (Statisch Typisierung für JavaScript)
- React / Nextjs (Framework für Webanwendungen)
- MUI (Komponenten Bibliothek für Google Material Design)

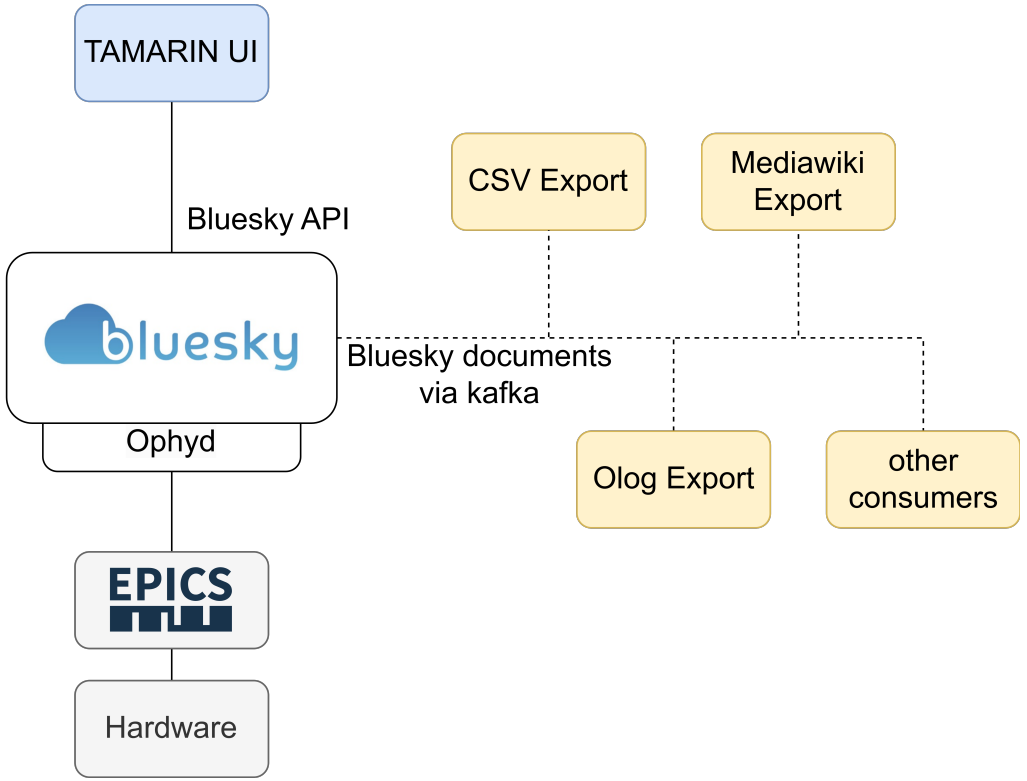
APIs

- Ophyd Rest API
- Queueserver HTTP API
- Tiled

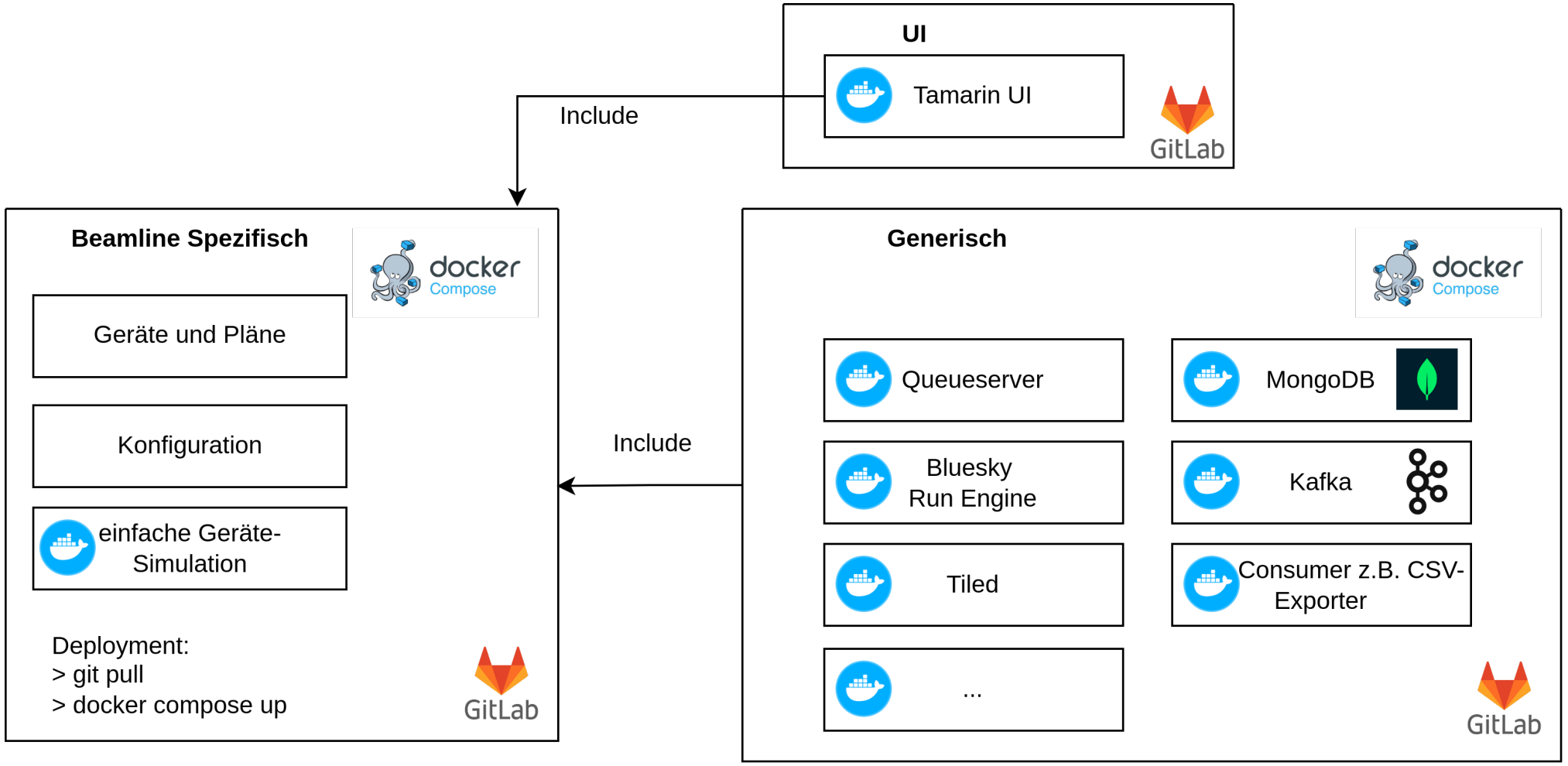


Erweiterbarkeit - Apache Kafka Interface

- Native Unterstützung in Bluesky
- Python Paket zur Erstellung von Consumern
- Verbreitete Event Streaming Plattform
- Lose Kopplung
- Erweiterbarkeit durch Tools zu Live-Daten Verarbeitung



Struktur Container - Deployment



Quellen und Links

Bluesky: https://blueskyproject.io/