From Desktop to Web:
The Evolution of Our GUIs from PyQt to React

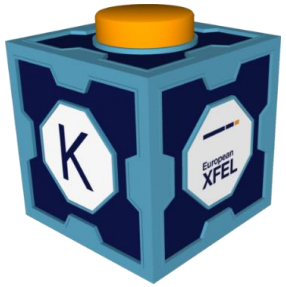
Cammille Carinan
Software Developer – Data Scientist
Data Analysis Group

Grenoble, France
September 23, 2024

European XFEL currently uses PyQt for several of our critical applications

Controls and Data Analysis groups are now transitioning to web technologies

- Advantages:
  - Cross-platform accessibility
  - Modern and responsive interfaces
  - Centralized deployment
- Frontend: Consistent across applications
  - React (JavaScript library)
  - Redux (state management)
- Backend: Implementation varies by application
  - FastAPI (Python web framework)
  - Tailored solutions to meet specific requirements

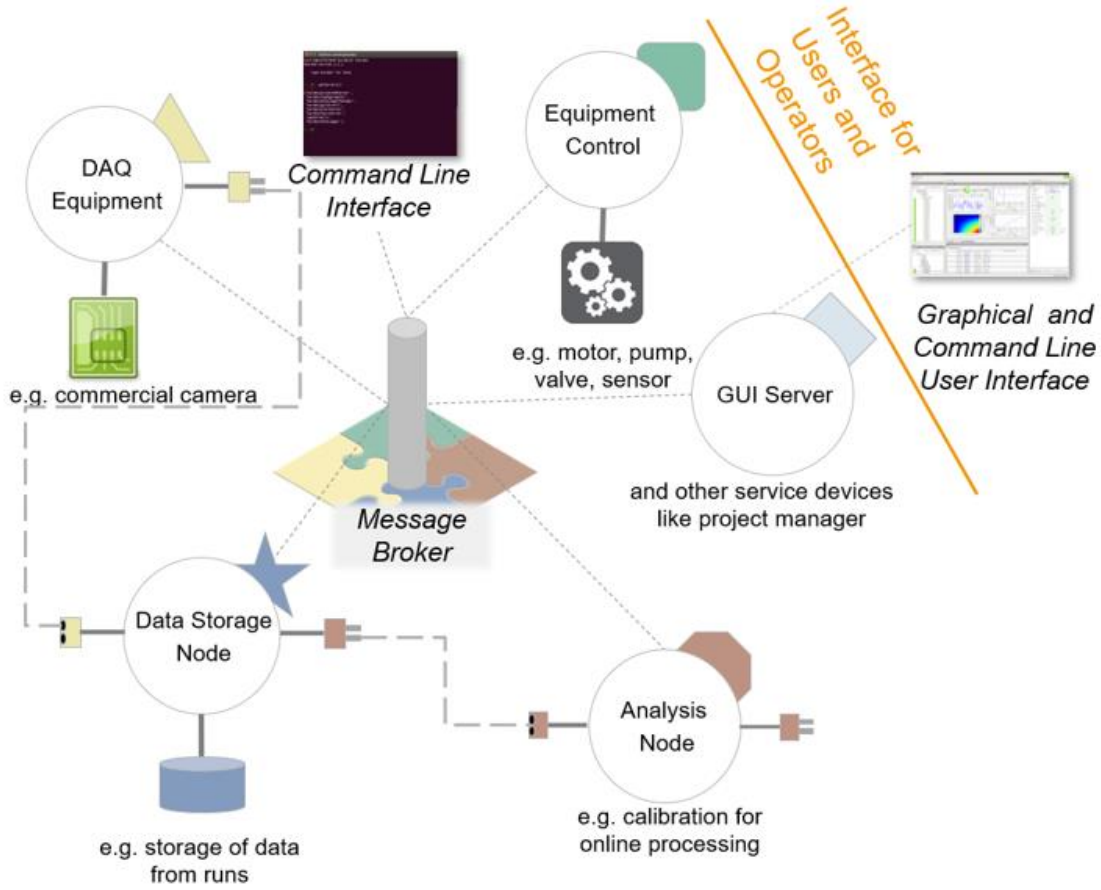


Karabo
(distributed controls system)
http://karabo.eu/

DAMNIT
(experiment data and metadata management)
https://damnit.rtfd.io/

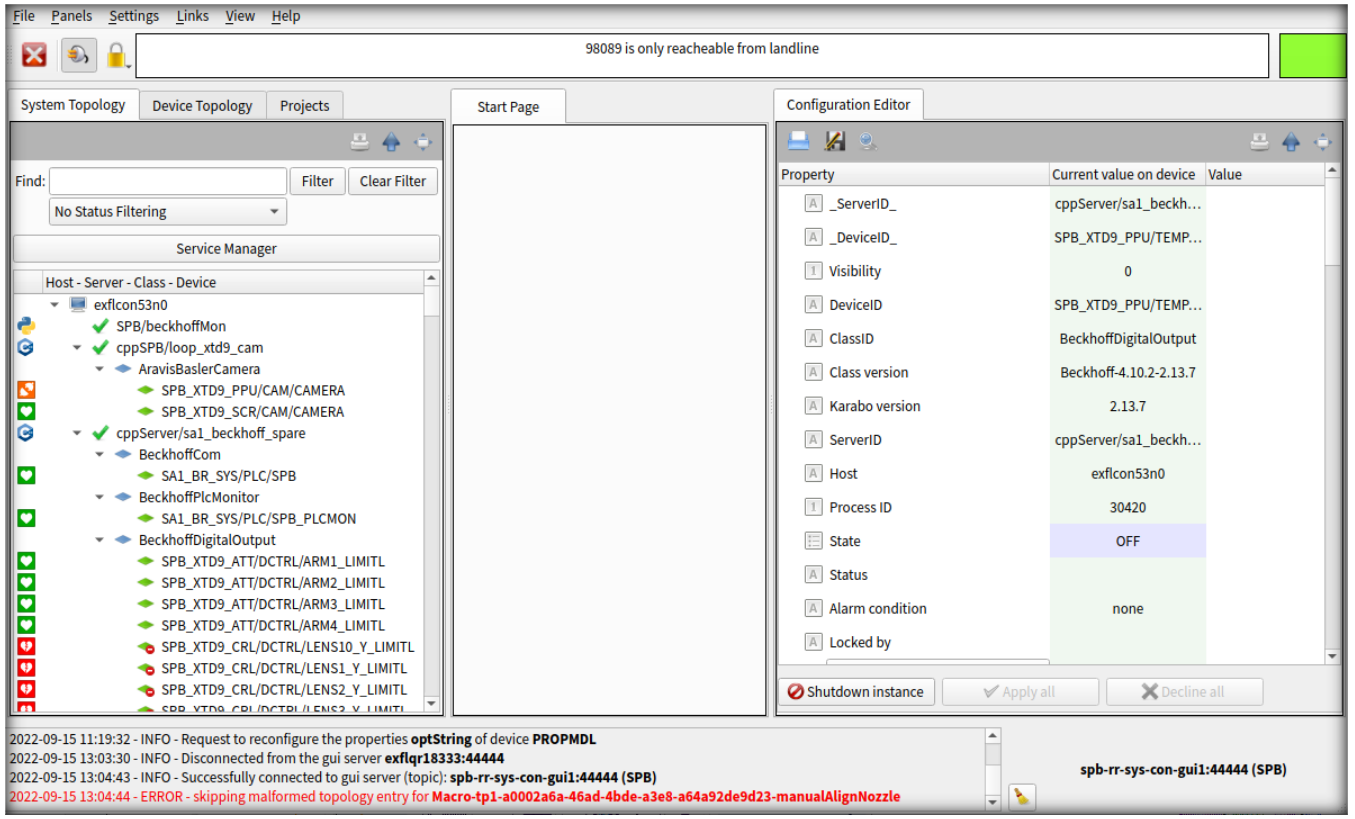
Karabo in a nut shell: Architecture

- Central Message Broker: RabbitMQ (Control data)
- Event and message driven:
  - Data propagates through the system on value change
  - Signal – Slot paradigm: publish and subscribe
- Peer-to-peer TCP connections for fast data
- GuiServer Device:
  - **Gateway** to the control system
  - TCP connection to application clients



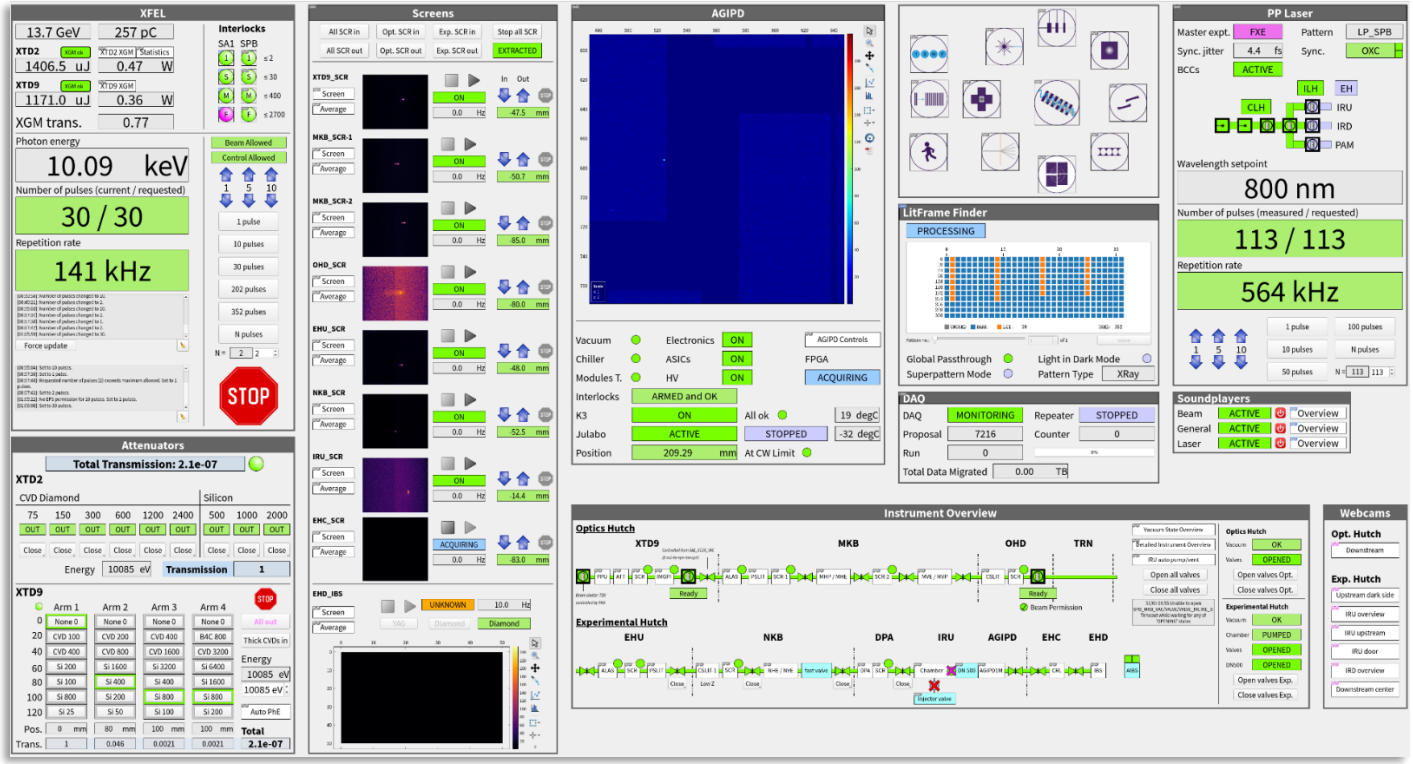
Karabo GUI – The Cockpit

- **Python** software contained in Karabo framework
  - Separate package installation
- Technology: **Qt** Library and **Traits**
- **Core feature:**
  Scene (Panel) Designer
  Scene Model Interpreter
  - Extensible via guiextensions:
    a plugin updater for more widgets and controllers



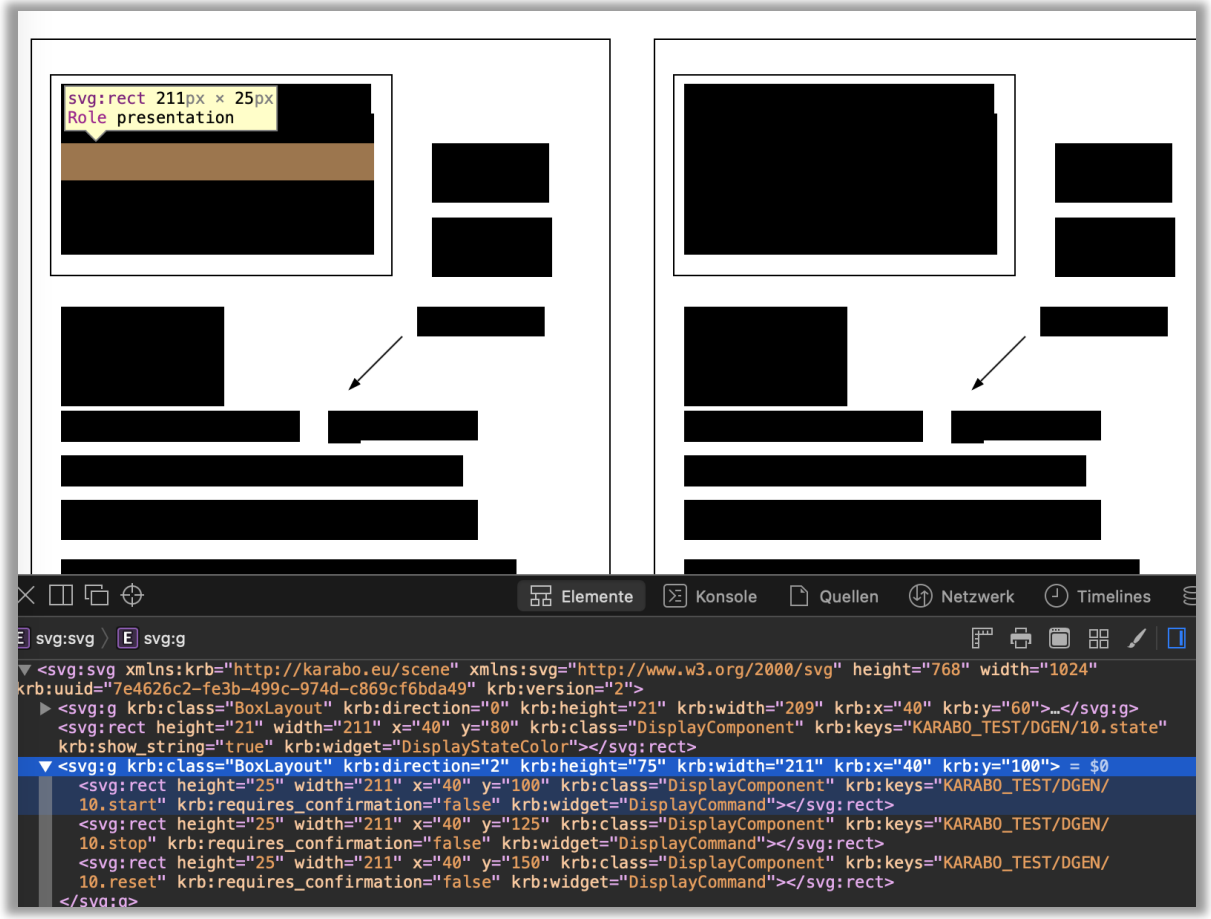
Karabo GUI – The Cockpit

- **Python** software contained in Karabo framework
  - Separate package installation
- Technology: **Qt** Library and **Traits**
- **Core feature:**
  Scene (Panel) Designer
  Scene Model Interpreter
  - Extensible via guiextensions:
    a plugin updater for more widgets and controllers



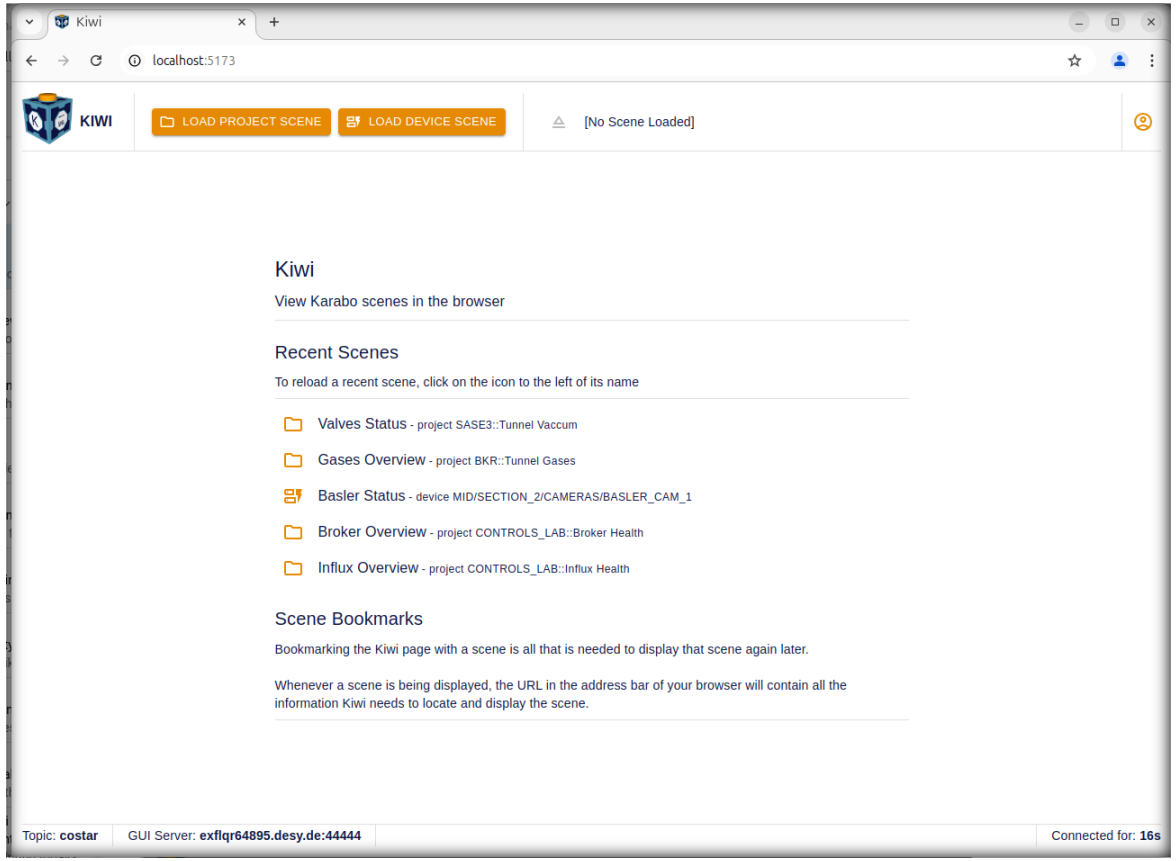
The Core Feature: The Scene

- Linked to every widget controller is a widget model
  - Separation of model and UI code facilitates external tools and applications
  - Scenes are stored as *.svg
  - The scene model is UI technology independent
- Scenes are stored in central database (ExistDB)



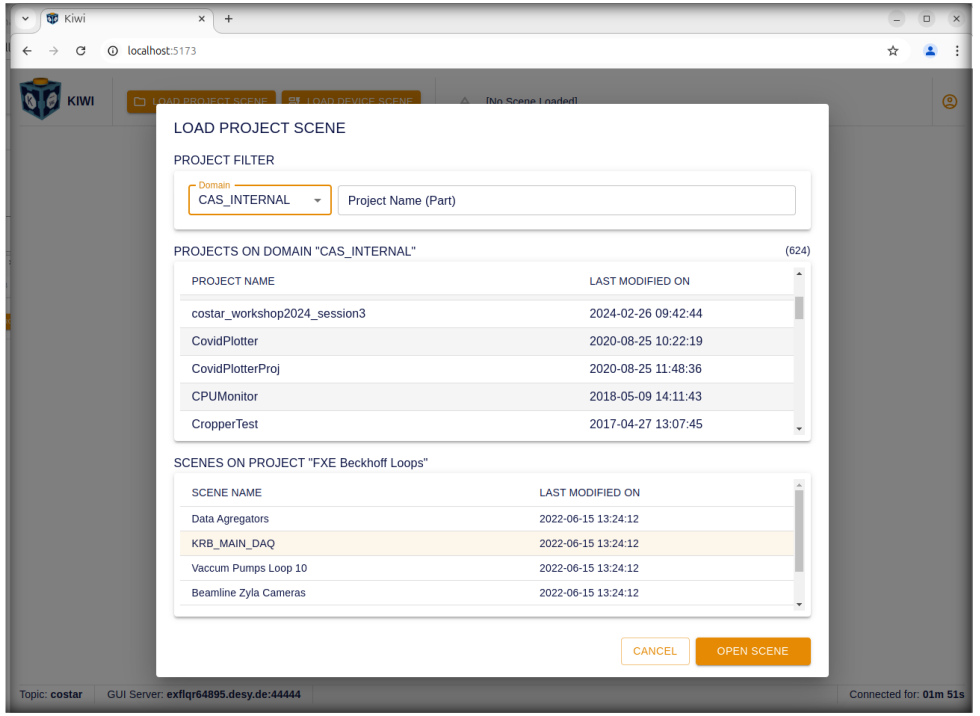
Motivation

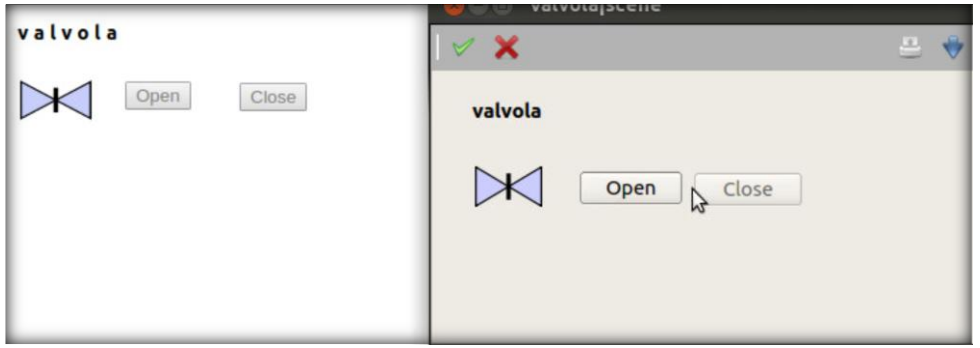
- Encapsulate scenes in web form
  - To preserve integrity
  - To combine with other web services and control systems **(Synoptic View)**
- Hybrid between Qt and Web
  - Build Karabo scenes in Qt GUI Editor
  - Direct Web display
- Use of strong authentication features on the web (LDAP, KeyCloak)
- Server-side deployment instead of application deployment



Motivation

- Encapsulate scenes in web form
  - To preserve integrity
  - To combine with other web services and control systems **(Synoptic View)**
- Hybrid between Qt and Web
  - Build Karabo scenes in Qt GUI Editor
  - Direct Web display
- Use of strong authentication features on the web (LDAP, KeyCloak)
- Server-side deployment instead of application deployment





Karabo GUI - Strategy

- Backend: GuiServer device used for both Web and Qt Application (share same protocol)
  - WebSocket to TcpSocket mediator
- Technology Stack:
  - Frontend: React, Redux
  - Backend: FastAPI
- Translation of Karabo scene model to web is ongoing
  - Derive synoptic web view (read only) and concept of fast image streaming by beginning of **2026**
  - Provide plugin for widget extensions for other applications (Scantool Karabacon) by **2026**

Concept: Synoptic web view

Full user-facing experiment in a web view

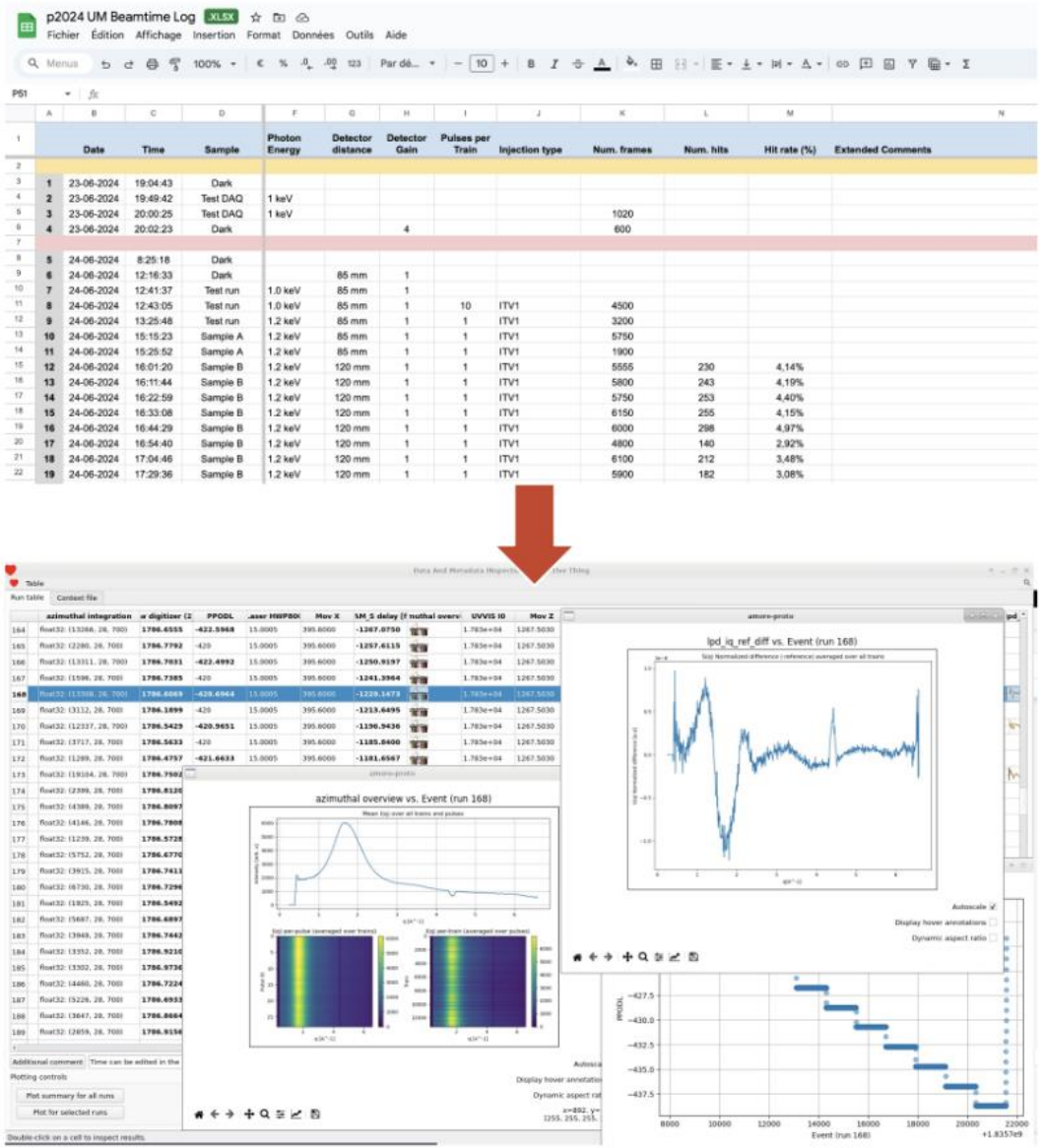
2025

Work Group (~3 personel)

2026

DAMNIT in a nutshell

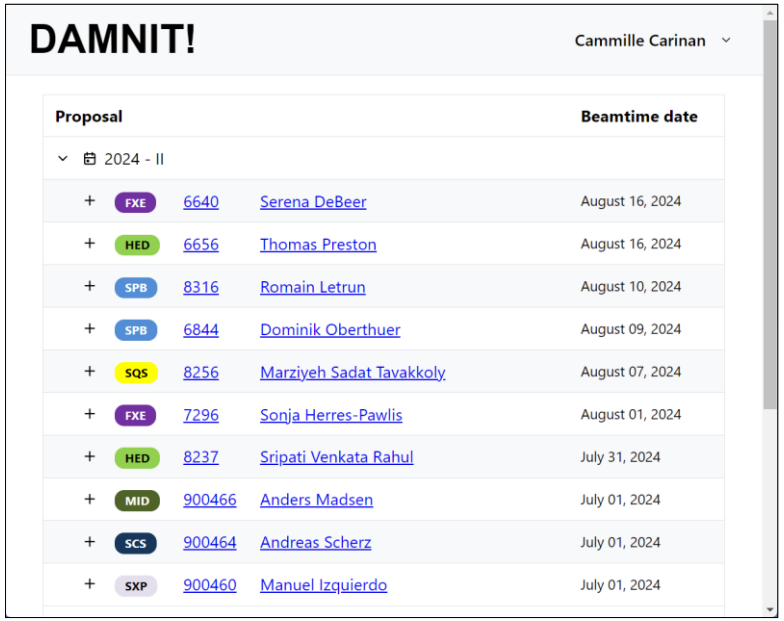
- Advanced alternative to traditional spreadsheets, specifically designed for **run tables**
- Interactive **PyQt**-based desktop application for data visualization and exploration
- Tabular format allows intuitive management of experimental data and metadata
- Automation through Python scripts for backend data processing and storage, integrated within the interface

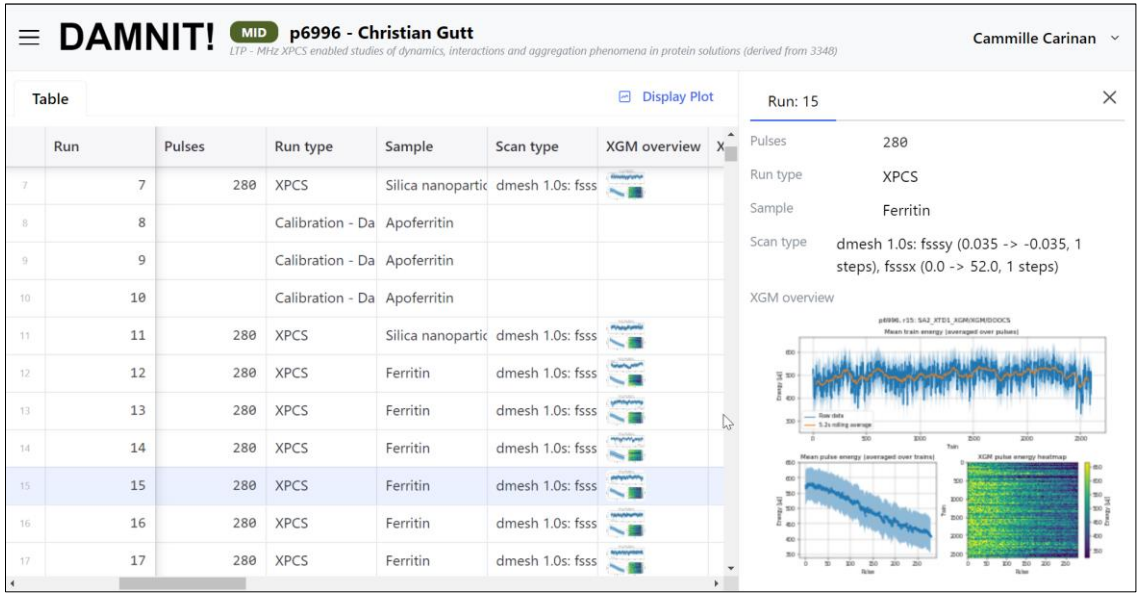


DAMNIT talk on the *data reduction session* (Thomas M.)
Wednesday, 24. September 2024, 5:10 PM

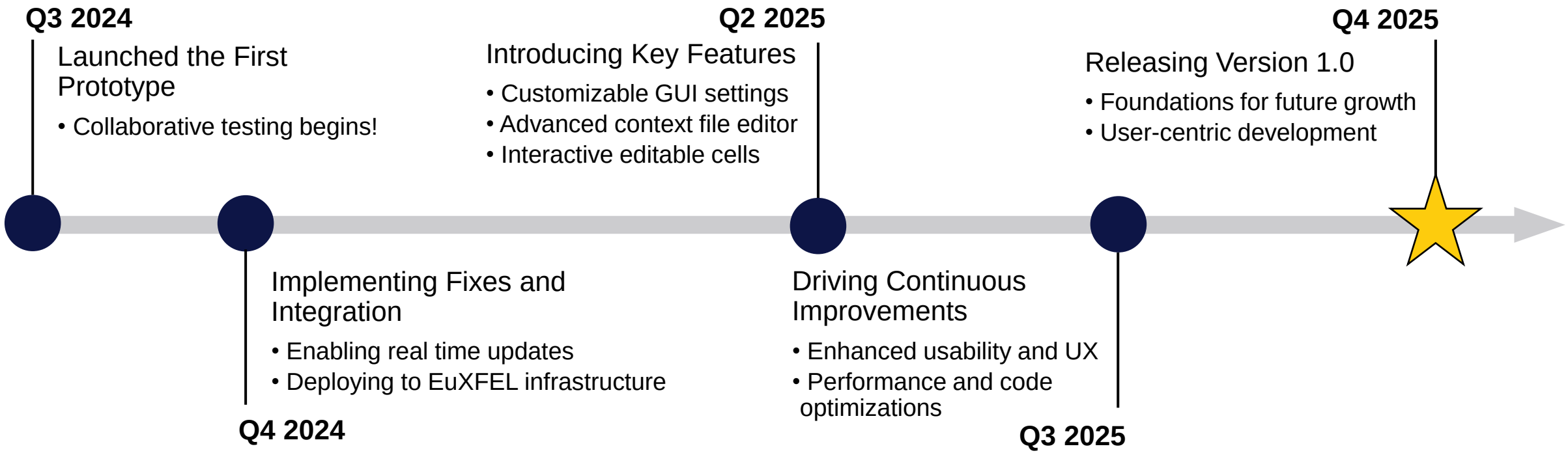
DAMNIT-web: First Prototype

- Frontend: React, Redux, Apollo GraphQL
- Backend: FastAPI, Strawberry GraphQL
  - OAUTH authentication
  - LDAP authorization
- Benefits of transitioning to web:
  - Enhanced usability with improved interface
    - ► Table pagination with infinite scrolling
    - ► Faster overview and interaction
  - Accessibility outside of the compute cluster
    - ► Web-based access
    - ► Secure external access
  - Real time updates via GraphQL subscriptions (in development)





Roadmap: From Prototype to Release



Part of the work presented here is supported by the DAPHNE4NFDI consortium, which is funded by the DFG (project number 460248799).

Questions for the Community

- Transition Experiences
  - Have you undertaken a similar transition from desktop to web-based applications? If so, what were the key challenges you faced?
- Maintaining Consistency
  - What best practices have you implemented to maintain consistency across applications during a technology transition?
- Enhancing User Experience
  - What strategies have you found effective in improving user experience when redesigning application interfaces?
- Utilizing Metrics and Insights
  - Do you use metrics and other methods to gather insights that inform your design and development decisions? If so, which metrics and techniques have been most beneficial?