Designing and deploying a FAIR-by-design data pipeline and platform for electron microscopy laboratories

Research thesis in: Data Management

Supervisor
Dott. Federica Bazzocchi

Candidate
Nicola Perin

University of Trieste

19 settembre 2025

Outline

1. Electron microscopy data: what it looks like and where the problems are
2. FAIR principles and the NeXus (NXem) standard
3. The case study: LAME and the ORFEO datacenter
4. From lab problems to a proposal
5. Designing the web application
6. Testing with VirtualOrfeo
7. Walking through the app
8. Live demo

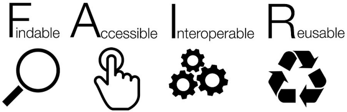
Electron microscopy data: an overview

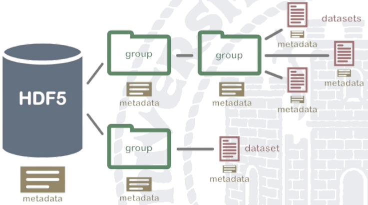
- Instruments (TEM, SEM, STEM) generate images, diffraction patterns, and spectra — often multi-terabyte datasets.
- Each vendor uses their own file formats; metadata is often incomplete or inconsistent.
- Day-to-day handling is messy: manual copies, endless zip files, unclear provenance.
- This makes collaboration and reuse difficult.

The challenge: how do we handle this data so it remains usable and shareable?

A way forward: FAIR and NeXus (NXem)

- FAIR principles: **findable, accessible, interoperable, reusable.**
- **HDF5**: efficient format for large, structured datasets.
- **NeXus**: conventions for scientific data (`NXinstrument`, `NXsample`).
- **NXem**: application definition tailored to electron microscopy.



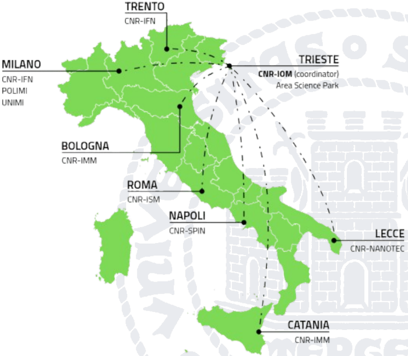


NeXus

At the national level, these FAIR practices are promoted and supported by the NFFA-DI infrastructure.

Introducing NFFA-DI

- **NFFA-DI** = Nano Foundries and Fine Analysis – Digital Infrastructure.
- Italian research initiative connecting major nanoscience centers.
- Goal: open access to advanced instrumentation, FAIR data, and computational resources.
- Acts as the national driver for FAIR data practices in nanoscience.



Source: https://nffa-di.it/en/

Introducing LAME

- LAME = Electron Microscopy Lab at Area Science Park, opened in 2022.
- Core instrument: a **double-corrected TEM/STEM** with advanced detectors.
- Enables atomic-scale imaging, spectroscopy, and in situ experiments (gas, heating, electrical bias).
- Supports nanoscience projects within NFFA-DI and European collaborations.

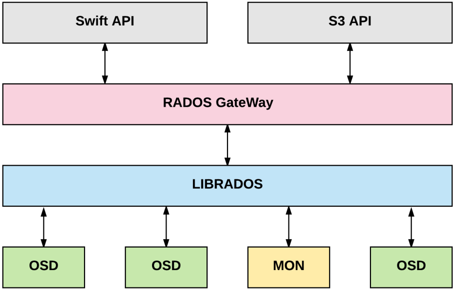
From the lab to the datacenter

- **LAME** produces multi-terabyte datasets in electron microscopy.
- As part of NFFA-DI, its work depends on sharing data with other partners.
- To support this, **ORFEO** provides the backbone: HPC resources, identity services, and S3-compatible object storage.
- The challenge: connecting LAME's lab workflows with ORFEO's infrastructure.

ORFEO's storage model

For LAME, ORFEO relies on an **object storage** system:

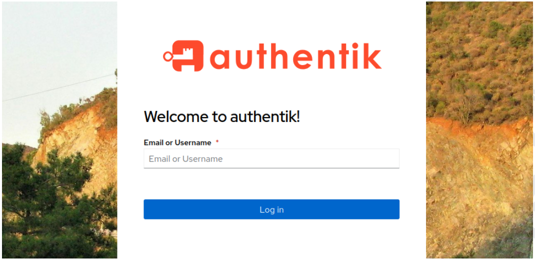
- An **object** is a file together with all the metadata that describes it.
- A **bucket** is a container that holds objects (like a project folder, but flatter).
- Access is through the **S3 protocol**, which is scalable and widely supported.



Accessing ORFEO

Access to ORFEO is managed through a **central identity system**:

- **FreeIPA** provides the underlying user and group management.
- **Authentik** builds on top of it to offer modern single sign-on (SSO).
- Users log in once and get consistent access across all services.



From problems to a proposal

What's missing for LAME

- Data often stuck on lab machines or portable drives.
- Transfers are manual, with inconsistent folder structures.
- No ingestion standard → hard to reuse or integrate with ORFEO/NFFA-DI.

Our proposal

- **Transfer**: move data directly into ORFEO using the S3 protocol.
- **Transform**: convert outputs (e.g. TIFF) into NeXus/NXem with standardized metadata.
- **Integrate**: build on ORFEO's existing services, with a simple web interface and API.

Choosing a framework

To put our proposal into practice, we need a tool that researchers can actually use. That means building a **web application** that can:

- guide researchers through projects, samples, and experiments,
- handle uploads and metadata in a consistent way,
- connect directly to ORFEO's services (S3 storage, authentication).

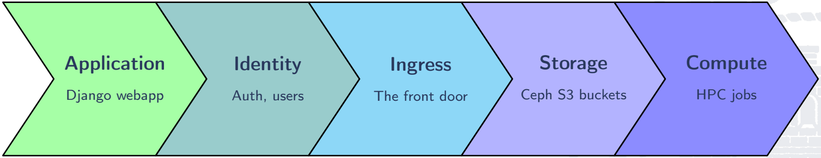
Once this was clear, the next step was to choose the right framework.

Why Django?

We chose Django because it makes it easy to translate lab workflows into software:

- **Models** let us map real-world entities (projects, samples, experiments, measurements) directly into the database.
- Comes with both a user-friendly interface and a REST API, so work can be manual or automated.
- Plays well with background workers for tasks like checksums and NeXus conversion.
- Strong authentication support, fitting smoothly into ORFEO.

Thinking about the whole pipeline



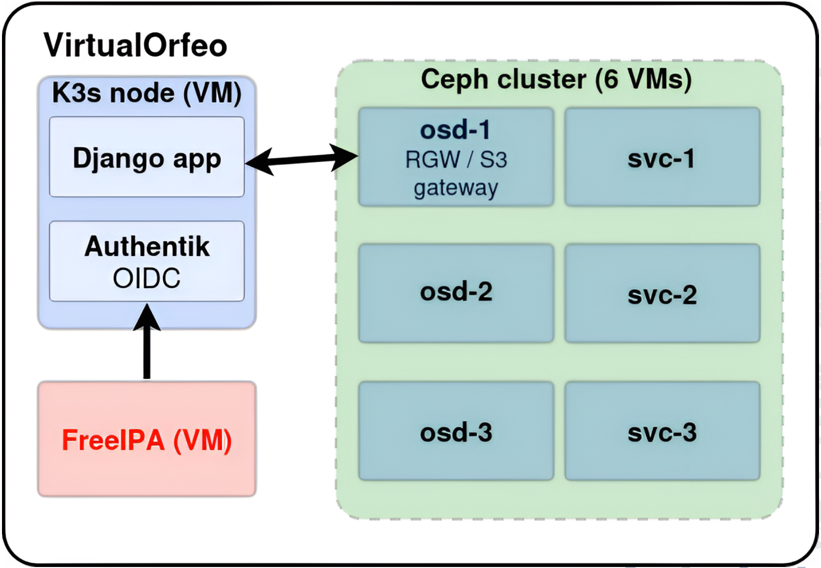
The app is one part of this chain — testing only makes sense when the whole path is reproduced. The solution: a **digital twin** of ORFEO.

VirtualOrfeo

ORFEO is a complex infrastructure: identity, ingress, storage, and compute. Testing our Django app directly on production would be risky and slow.

- **VirtualOrfeo** is a lightweight clone of ORFEO, built on K3s.
- It uses the same Helm charts and configs as production.
- This lets us deploy the **Django app** in a realistic environment: it can authenticate through Authentik, upload to S3 buckets, and be accessed through the same ingress as in ORFEO.

VirtualOrfeo topology



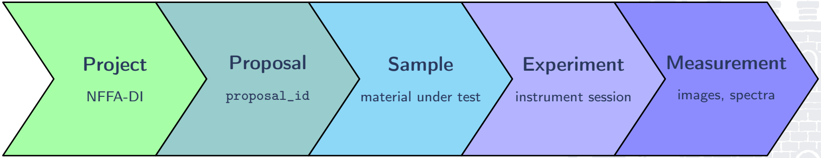
Application overview

The webapp is designed around three main pillars that keep data organized, easy to move, and ready to be reused:

- **Structure** — organize research work into a consistent model.
- **Flow** — move data reliably into storage with linked metadata.
- **Automation** — offload heavy tasks to background workers.

Research data model

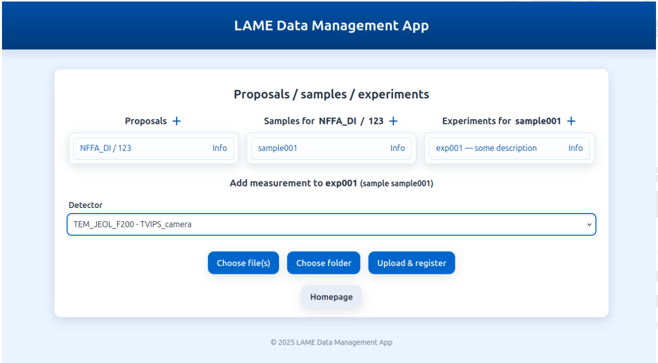
The app organizes research work in a structured chain:



This keeps context together with raw data, making results easier to track and reuse.

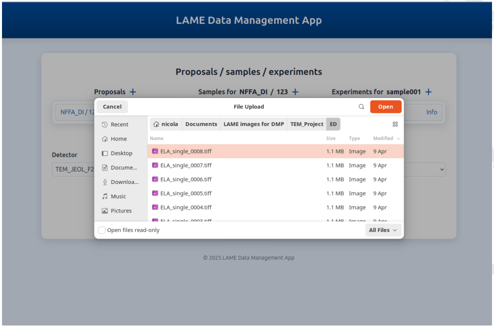
Managing research data in practice

- Three-pane board to browse and link projects, samples, and experiments.
- Metadata (context) stored alongside raw data (`README.txt`).
- Same operations available via REST API for automation.



From upload to storage

- The app issues a one-time presigned URL.
- Browser streams data **directly to S3**, bypassing the webserver.
- Uploads automatically trigger a background job: checksum $\rightarrow$ metadata extraction $\rightarrow$ NeXus build.

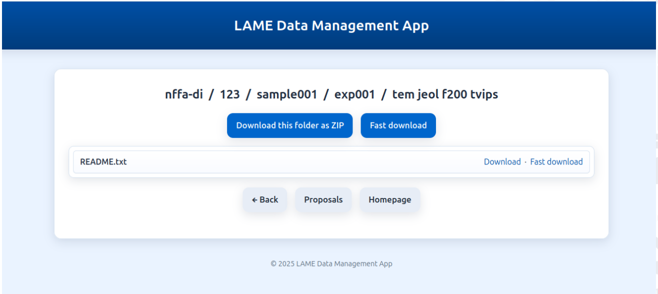


Background workers: from raw files to NeXus

- A **worker pod** runs in the cluster, always listening to Redis (a shared **to-do list**).
- When a file is uploaded, the app enqueues jobs such as:
  - Metadata extraction (parse TIFF headers or JSON).
  - Normalization into NXem fields.
  - NeXus generation: structured `.nxs` file with metadata.
- Jobs are picked up one by one, retried automatically if they fail.

Browsing and sharing data

- Browse buckets and datasets directly from the interface.
- Download with short-lived presigned links.
- Create on-the-fly ZIP archives for folders.
- Derived data stored in mirrored namespaces for clarity.



Live Demo

Let's see the workflow in action.

Thank you for your attention!