



MDMC

Master in Data Management
and Curation

MASTER IN DATA MANAGEMENT AND CURATION

A FAIR Data Ecosystem in Transmission Electron Microscopy

Supervisor(s):

FEDERICA BAZZOCCHI,
PAOLO RONCHESE

Candidate:

NICOLA PERIN

2024–2025





Finanziato
dall'Unione europea
NextGenerationEU



Ministero
dell'Università
e della Ricerca



Italiadomani
PIANO NAZIONALE
DI RIPRESA E RESILIENZA

This Pilot training activity has been funded by the European Union – NextGenerationEU within the project PNRR "PRP@CERIC" IR0000028 and «NFFA-DI" IR0000015 - Missione 4, "Istruzione e Ricerca" – Componente 2, "Dalla ricerca all'impresa" – Linea di investimento 3.1, "Fondo per la realizzazione di un sistema integrato di infrastrutture di ricerca e innovazione" – Azione 3.1.1, "Creazione di nuove IR o potenziamento di quelle esistenti che concorrono agli obiettivi di Eccellenza Scientifica di Horizon Europe e costituzione di reti".

The supporting projects:

- [NFFA-DI : Nano Foundries Fine Analysis - Digital Infrastructure](#)
- [PRP@CERIC : Pathogen Readiness Platform for Ceric - Eric Upgrade](#)



nffa-di



prp

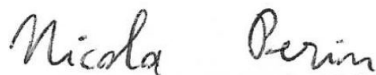
Author's Declaration

I, NICOLA PERIN, declare that this thesis entitled, 'A FAIR Data Ecosystem in Transmission Electron Microscopy' and the work presented therein are my own.

I certify that:

- This work was performed wholly or principally during MDMC internship in Laboratorio di Microscopia Elettronica (LAME) at Area Science Park.
- If any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have relied on the published work of others, this is always clearly attributed.
- Where the thesis is based on work I have done in collaboration with others, I have made clear exactly what was done by others and what I contributed.
- With respect to AI technologies:
 - I acknowledge the use of ChatGPT by OpenAI (<https://chat.openai.com>) and in particular its deep research function to provide information for background research and to assist in the drafting writing process with the creation of an outline structure for this essay.
- Where I have quoted from the work of others, the source is always cited. Except for such quotations, this thesis is entirely my own work.
- I have acknowledged all major sources of help.

Signed:



Date: April 30, 2025

“Data is like garbage. You’d better know what you are going to do with it before you collect it.”

– Mark Twain (probably not, but it sounds like him)

Acknowledgments

I would like to thank my supervisor, Dr. Federica Bazzocchi, for her guidance and support throughout this work. Her input was valuable during both the planning and development phases of the project.

My appreciation goes to Isac Pasianotto for his assistance with the Authentik integration and testing, and to Dr. Marco Prenassi for his support in developing the Django application. I am also grateful to Dr. Ruggero Lot for his help with network configuration and Kubernetes deployment, and finally to Dr. Paolo Ronchese for providing the microscopy data and explaining the instrument's operation.

Abstract

This thesis presents the design and development of a web-based data management platform tailored to the needs of the Laboratory for Electron Microscopy (LAME) at Area Science Park, with a specific focus on implementing the FAIR (Findable, Accessible, Interoperable, Reusable) data principles.

The work is situated within the broader context of Italian and European research infrastructure initiatives such as NFFA-DI and FAIRmat, which promote standardized, FAIR-by-design data workflows across nanoscience and materials science domains. The platform, built using the Django web framework, enables researchers to upload raw electron microscopy data—such as TEM images, diffraction patterns, and EDS spectra—and convert them into standardized NeXus files based on the NXem application definition, stored in an object storage backend powered by MinIO.

To support secure and scalable access, the system integrates with Authentik for identity and access management, enabling single sign-on (SSO) and role-based permissions. The final platform provides researchers with a user-friendly interface for organizing, curating, and sharing their experimental data in a reproducible and standards-compliant way.

Contents

1	Introduction	8
2	Background	10
2.1	NFFA-DI and FAIRmat initiatives	10
2.1.1	NFFA-DI (Nanoscience Foundries & Fine Analysis – Digital Infrastructure)	10
2.1.2	FAIRmat (FAIR Data Infrastructure for Materials Science)	11
2.1.3	Integrating LAME with national FAIR initiatives	11
2.2	Electron microscopy background: TEM, ED, and STEM instrumentation	11
2.2.1	Transmission Electron Microscopy (TEM)	11
2.2.2	Scanning Transmission Electron Microscopy (STEM)	12
2.3	HDF5 and NeXus Data Formats	12
2.3.1	HDF5 (Hierarchical Data Format Version 5)	12
2.3.2	NeXus: a scientific convention layer built on HDF5	12
2.3.3	NXem: the Electron Microscopy application definition	13
3	Experimental Setup	14
3.1	Instrumentation and Software	14
3.1.1	JEOL JEM-F200 “F2” Microscope and Detectors	14
3.1.2	Detectors and software	15
3.1.3	STEM-EDS data formats and integration challenges	16
3.2	Experiment scope and data organization	17
3.2.1	Example of experiment and raw data structure	17
3.2.2	Supporting software	18
3.2.3	Bucket and file-naming convention under the NFFA-DI Research Data Policy	19
3.2.4	Concrete example	21

4	A Django web application for FAIR electron-microscopy data	22
4.1	Overall user experience	24
4.2	Upload workflow	24
4.3	Inside the NeXus builder	27
4.4	NeXus file validation	29
4.5	MinIO: the object store that replaces a file server	30
4.6	Authentik: our facility's central Identity Provider	31
4.7	Production deployment with containers and Kubernetes	33
4.8	Current FAIR status	34
4.9	Next milestones	34
5	Conclusion	36
A	Example NeXus tree	38
B	JSON mappings for TIFF → NXem	40

Chapter 1

Introduction

Area Science Park, located in Trieste, Italy, is a public research organization supervised by the Italian Ministry of University and Research. Established in 1978, it operates as a bridge between research, innovation, and industry, hosting a wide range of companies and research centers [1]. Among its specialized laboratories is the *Laboratory of Electron Microscopy* (LAME), created in 2022 to provide advanced characterization capabilities for nanoscience and nanotechnology research.

LAME focuses on TEM (Transmission Electron Microscopy) and STEM (Scanning Transmission Electron Microscopy) analyses of materials such as complex oxide thin films, nanocatalysts, and energy-related devices. Its state-of-the-art instrumentation includes a JEOL F200 TEM/STEM equipped with dual EDS detectors and a hybrid-pixel detector, as well as a JEOL Grand Arm 300kV microscope and a Plasma FIB-SEM Tescan Amber X [2]. These instruments generate a wealth of complex data—images, diffraction patterns, X-ray and EELS spectra—that require careful management and long-term preservation.

To address these challenges, LAME has adopted the principles of FAIR data: data should be **F**indable, **A**ccessible, **I**nteroperable, and **R**eusable [3]. Ensuring FAIRness from the early stages of data collection is essential for maximizing scientific impact and enabling reproducibility.

The NFFA-DI (Nanoscience Foundries & Fine Analysis – Digital Infrastructure) project, supported by the PNRR (Italy’s National Recovery and Resilience Plan), exemplifies this shift. It provides shared resources, infrastructure, and expertise to help research institutions implement FAIR data workflows. This includes OFED (cf. Chapter 2), a national platform for storage and metadata management, and the training of in-lab data stewards to oversee local adoption of best practices [4, 5].

In this work, I describe how we built a FAIR-by-design infrastructure

for handling data produced at LAME. Starting from the instrument outputs and metadata capture, the system moves through standardized storage in HDF5/NeXus and culminates in a web-based portal for managing, sharing, and accessing curated datasets. These efforts are part of a broader shift toward systematic, reusable data workflows that align with the FAIR principles from the moment of data creation.

The remaining chapters describe the technical foundations and our concrete implementations:

- **Chapter 2** reviews the landscape of FAIR data infrastructures—covering the NFFA-DI and FAIRmat initiatives—and introduces the key concepts of electron microscopy (TEM, SAED, STEM-EDS). We explain why HDF5 combined with the NeXus NXem schema provides a natural fit for multi-dimensional EM datasets, and how community tools (e.g. `pynxtools`) and standards enable automated, self-describing data files.
- **Chapter 3** examines LAME’s instrumentation and raw data workflows. Beginning with the JEOL JEM-F200 TEM/STEM (and its dual-EDS and hybrid-pixel detectors), we survey the various acquisition software (TEM Center, Panta Rhei, TVIPS EM Menu) and the resulting file formats. This chapter also introduces our policy-driven bucket and file-naming convention under the NFFA-DI data policy, illustrating how structured names encode experiment identifiers, instrument details, sample tags, dates, and sequence numbers.
- **Chapter 4** presents the architecture and workflow of the web-based data management system (see Figure 4.1). We cover metadata mapping via JSON definitions, the automated conversion of TIFF images into NeXus NXem files, the web interface for metadata entry, secure storage in MinIO with pre-signed URLs, Single Sign-On with Authn-tik, validation against official NeXus definitions, and the containerized Kubernetes deployment for production.
- **Chapter 5** concludes with a summary of achievements, reflections on FAIR-by-design implementation at LAME, and an outlook on planned enhancements—such as full batch processing, unified storage backends, lightweight search indexing, and NOMAD OASIS integration.

Chapter 2

Background

2.1 NFFA-DI and FAIRmat initiatives

2.1.1 NFFA-DI (Nanoscience Foundries & Fine Analysis – Digital Infrastructure)

NFFA-DI is a national initiative funded under the PNRR (Piano Nazionale di Ripresa e Resilienza, Italy’s National Recovery and Resilience Plan) that aims to implement FAIR-by-design data management practices across nanoscience research facilities in Italy [4]. It brings together 11 Operative Units, each responsible for specific instruments and workflows, with the shared goal of building end-to-end FAIR pipelines. LAME is one of these units, contributing its electron microscopy expertise and infrastructure. Within this framework, LAME develops and maintains its own tools and conventions while aligning with the project-wide standards for data acquisition, metadata capture, and long-term preservation.

A central component of NFFA-DI is OFED, the *Overarching FAIR Ecosystem for Data*, which goes beyond a simple storage backend. OFED acts as a digital ecosystem: it provides S3-compatible buckets for archiving NeXus files, offers services for metadata indexing and visualization, and enables future integration with analysis platforms and national data portals [5]. The infrastructure is hosted at the Orfeo HPC cluster and operated using containerized microservices under Kubernetes.

To support implementation at the lab level, NFFA-DI also promotes in-house expertise through the training of data stewards—technical staff embedded in each participating laboratory. These professionals ensure that datasets meet quality requirements and metadata standards, and they help researchers shift from ad hoc practices to systematic, reusable workflows.

2.1.2 FAIRmat (FAIR Data Infrastructure for Materials Science)

FAIRmat is a group within Germany’s NFDI program that works on creating tools and rules for managing data in a FAIR way, especially in areas like condensed-matter physics and solid-state chemistry [3]. It builds on the NOMAD platform, which stores data from computer simulations of materials, and expands into experimental research as well.

One of FAIRmat’s key efforts is improving NeXus data formats for lab techniques like electron microscopy. FAIRmat’s work on schemas like `NXem` is especially useful for labs like LAME, giving them a clear and practical way to organize electron microscopy data—including images, diffraction patterns, and spectra—along with important metadata [6].

2.1.3 Integrating LAME with national FAIR initiatives

Both NFFA-DI and FAIRmat reflect a broader shift in scientific research: building infrastructures where data are automatically captured, annotated, and stored in standardized formats. This is especially relevant for electron microscopy, where modern instruments produce multi-dimensional datasets requiring careful documentation (instrument modes, scanning parameters, sample provenance, etc.). By aligning local data workflows at LAME with these larger initiatives, the lab aims to enhance the findability and interoperability of its EM data and ensure that valuable scientific results remain accessible for future analyses.

2.2 Electron microscopy background: TEM, ED, and STEM instrumentation

2.2.1 Transmission Electron Microscopy (TEM)

Transmission Electron Microscopy (TEM) is a powerful imaging method that uses a beam of high-energy electrons to pass through a very thin sample and create detailed images. It can detailed information of the specimen at the atomic level, which makes it especially useful for characterizing the structure and composition of nanomaterials [7].

TEM also allows the study of crystal structures by means of Electron Diffraction (ED). By focusing the electron beam on a small part of the sample, it’s possible to create patterns (spots or rings) that reveal how atoms are arranged and how far apart they are.

Modern TEM systems—like the JEOL instruments used at LAME—can easily switch between imaging and diffraction modes. This allows researchers to get both detailed images and structural information from the same sample.

2.2.2 Scanning Transmission Electron Microscopy (STEM)

STEM involves rastering a focused electron probe across the sample and detecting scattered electrons at each probe position [7]. The images collected are characterized by compositional contrast depending on the chemical composition of the specimen (e.g., HAADF or BF imaging). STEM is widely employed for exploiting all the capabilities of powerful analytical techniques such as:

- **STEM-EDS:** Collecting X-ray signals generated from the specimen at specific sites, and producing elemental maps, spectra and line profiles after data elaboration.
- **STEM-EELS:** Analyzing the energy loss of electrons transmitted through the sample to infer composition and valence state.

The combination of these techniques with advanced detectors (e.g., hybrid pixel detectors) and multiple signal acquisition yields multi-dimensional datasets (up to 4D STEM), which demand robust data handling and metadata tracking.

2.3 HDF5 and NeXus Data Formats

2.3.1 HDF5 (Hierarchical Data Format Version 5)

HDF5 is a widely adopted file format in scientific computing, designed to store and manage complex, high-volume data. It organizes content into a hierarchical tree of groups and datasets, much like folders and files in a file system. This structure makes it easy to manage multi-dimensional arrays, along with descriptive metadata attached as attributes. HDF5 supports compression, partial I/O, and is optimized for performance and scalability—making it an ideal foundation for storing large-scale microscopy data such as high-resolution images, 4D STEM datasets, and spectroscopy maps [8].

2.3.2 NeXus: a scientific convention layer built on HDF5

HDF5 provides a flexible, hierarchical container for arbitrary datasets and metadata, but on its own offers no standardized way to organize scientific

experiments. NeXus fills that gap by defining a set of *base classes*—such as `NXentry`, `NXinstrument`, `NXsample` and `NXdata`—each with prescribed locations and attributes within the file. These base classes form the core schema that guarantees measurement results and metadata live in well-known places, improving interoperability across instruments and software tools [9, 10].

On top of these foundational classes, NeXus uses *application definitions* to specialize the schema for particular scientific domains. An application definition names which groups and datasets are *required*, *optional*, or *recommended* for a given technique, and may introduce additional domain-specific fields. This layered approach keeps the format both extensible and self-describing.

2.3.3 NXem: the Electron Microscopy application definition

For electron microscopy, the `NXem` application definition prescribes exactly how to store imaging, diffraction, spectroscopic and sample-preparation metadata within a NeXus file. It leverages the NeXus base classes (e.g. `NXentry` as the top-level container, `NXinstrument` for microscope details, `NXdata` for image arrays) and adds EM-specific groups for beam settings, detector configurations, stage positions, and more [11, 12]. By adhering to `NXem`, laboratories can produce self-describing files that integrate seamlessly into broader FAIR data workflows, without requiring custom parsers or ad hoc conventions.

The Django-based web application that will be introduced in Chapter 4 supports automated generation of `NXem`-compliant NeXus files directly from user-uploaded TIFF images.

Chapter 3

Experimental Setup

3.1 Instrumentation and Software

3.1.1 JEOL JEM-F200 “F2” Microscope and Detectors

The JEOL JEM-F200 “F2” is a 200 kV transmission electron microscope (TEM) equipped with a cold field-emission source. It is installed at LAME as a general-purpose analytical instrument, capable of both high-resolution imaging and spectroscopy-based techniques. Its configuration supports multiple modes of operation—conventional TEM, scanning TEM (STEM) and electron diffraction—making it a flexible tool for structural and compositional analysis of materials at the nanoscale [13].

The instrument uses a cold field-emission gun (FEG) as its electron source. This emitter provides a highly coherent and bright electron beam, which is beneficial for resolving fine structural details and enhancing signal quality in spectroscopic measurements.

For chemical analysis the microscope supports energy-dispersive X-ray spectroscopy (EDS) through the installation of two large-area silicon-drift detectors (SDDs). These detectors collect X-rays generated when the electron beam interacts with atoms in the sample. With a combined collection solid angle of up to 1.7 sr, the dual-detector configuration allows for higher sensitivity and faster acquisition times—essential when analysing beam-sensitive materials or low-concentration elements [13].

In addition to EDS, the system is equipped with a CEFID EELS spectrometer by CEOS, enabling fine-grained analysis of energy losses experienced by electrons passing through the specimen. EELS is particularly useful for studying chemical bonding, oxidation states and electronic structure. The microscope mounts also two TVIPS cameras for TEM images collection, each of the two equipped with a 4K CMOS sensor.

The JEM-F200 is controlled through JEOL’s *TEM Center* software, which provides an integrated interface for switching between imaging, diffraction and analytical modes. Automation routines for beam alignment, lens calibration and spectrum-imaging workflows reduce manual tuning and help maintain reproducibility—an important factor for collaborative studies. The microscope at LAME is capable of advanced imaging techniques as well: a clear example is represented by the possibility of performing 4D-STEM studies of nanomaterials by means of a DECTRIS ELA hybrid-pixel detector.

3.1.2 Detectors and software

The JEOL JEM-F200 system at LAME is equipped with a combination of detectors and software components that support multiple imaging and analytical modes. These include:

1. **Imaging Detectors (TEM mode):** High-resolution cameras, equipped with CMOS sensors, are used to capture real-space images and diffraction patterns.
2. **STEM Detectors:** Built-in annular detectors collect scattered electrons during scanning TEM imaging, enabling bright-field and dark-field contrast modes.
3. **EDS Detectors:** A pair of Silicon Drift Detectors (SDDs) mounted on the microscope column perform elemental analysis by detecting characteristic X-rays emitted by the sample.
4. **EELS Spectrometer:** A post-column spectrometer (by CEOS) records the energy losses of transmitted electrons, offering detailed insights into bonding and elemental composition.
5. **Microscope Control and Acquisition Softwares:** JEOL’s TEM Center is used to control the microscope in its full operational routine. Anaysis Station is employed for EDX spectra and elemental maps collection. EM-menu controls TVIPS cameras for TEM images recording. Panta Rhei serves as multiple platform for EELS spectra and maps collection, as well as 4D-STEM dataset acquisition.

The current configuration at LAME includes several third-party additions beyond the standard JEOL stack. A TVIPS camera is operated via the EM Menu software, while CEFID spectrometers are managed through the Panta Rhei platform.

This heterogeneous software landscape poses challenges for data management and metadata capture. While TIFF images produced by the TVIPS and JEOL systems can be parsed and translated into NeXus-compliant HDF5 files using the workflow described in previous sections, the same is not currently feasible for EDS data. The proprietary formats used by various EDS tools make it difficult to extract meaningful metadata in a consistent, automated way. As a result, no NeXus conversion pathway for EDS outputs is in place at this stage.

3.1.3 STEM-EDS data formats and integration challenges

STEM-EDS (Scanning Transmission Electron Microscopy with Energy Dispersive X-ray Spectroscopy) generates a variety of output files, many of which are tied to proprietary data formats. Common examples include:

- **.img** – Reference STEM images, typically in bright-field (BF) or high-angle annular dark-field (HAADF) mode.
- **.map** – Elemental maps that visualize the spatial distribution of specific elements. Each map is usually saved as a separate file per element.
- **.pts** – Line profile data capturing compositional variation along a user-defined path.
- **.apb** – Spectral data files that store X-ray spectra and associated calibration. These files are software-specific and often undocumented.

Together, these files form the core of EDS analysis workflows. However, unlike the TIFF images used for conventional TEM and STEM imaging, these data types are not easily readable without the original acquisition software. Their proprietary nature complicates efforts to integrate them into standardized formats like NeXus.

At present, the EDS data workflow at LAME remains decoupled from the NeXus export system. While there is long-term interest in encapsulating full EDS datasets (maps, spectra, and line scans) into HDF5 containers using the *NXem* schema, this would require additional tooling to parse and interpret the proprietary formats reliably. In contrast, the TIFF-based workflows—for both TVIPS and ED imaging—are already supported via JSON mapping and automated NeXus generation.

Finally, the volume of data produced during STEM-EDS sessions adds further pressure to the storage infrastructure. High-resolution spectrum

imaging or video capture can generate several terabytes of data in a single session. While the lab’s policy is to retain all raw data to preserve scientific value, this introduces ongoing challenges for archiving, indexing, and network performance.

3.2 Experiment scope and data organization

In discussions with lab staff we agreed that a single *experiment* corresponds to the work performed on *one* instrument during a given session. Although it is rare to probe exactly the same region in both SEM and TEM, a *parent-sample* identifier is introduced so that derived specimens (e.g. a TEM lamella cut by FIB) can be traced back to the original preparation step.

Because unexpected phenomena sometimes emerge only when reviewing long videos or large EDS stacks, *all* raw recordings are preserved. While scientifically prudent, this produces massive collections that are slow to transfer across the present network, hence the effort to impose a compact, policy-compliant structure.

3.2.1 Example of experiment and raw data structure

Structural analysis of Ni_3Sn_2 nanoparticles was carried out on the JEOL JEM-F200. High resolution TEM images have been recorded through a TVIPS TemCam-XF416 camera equipped with a 4K CMOS sensor and controlled by EM-Menu 5 software. Single tilt diffraction patterns have been acquired with a step of 0.5deg in an angular range of 50deg and recorded with a Dectris ELA hybrid pixelated detector controlled by CEOS Panta Rhei software. The goal was to obtain information about:

- the average particle size;
- the chemical composition via STEM-EDS;
- the crystal structure.

Combined analysis of images and diffraction patterns may suggest that the particles form a homogeneous Ni_3Sn_2 alloy rather than a core-shell morphology.

Raw output file structure

```
.
|-- ED
```

```
| |-- ELA_single_0001.tiff
| |-- ELA_single_0002.tiff
| |-- ...
| '-- ELA_single_0069.tiff
|-- STEM_EDS
| |-- 2024/SnNi_NPs/Sample/00_View000/
| | |-- View000_0000000.img
| | |-- View000_0000001.map
| | |-- ...
| | '-- View000_0000009.apb
| '-- 2025/2025_02/NiSn_NPs/Sample/00_View002/
| |-- View002_0000200.img
| |-- View002_0000201.dln
| |-- ...
| '-- View002_0000208.pts
'-- TEM
   |-- SnNi_NPs_20241223_1.tif
   |-- SnNi_NPs_20241223_1_scale.tif
   |-- ...
   '-- viascript05.tif
```

3.2.2 Supporting software

In addition to the microscope control, several specialized software packages were used at different stages:

- **EM Menu** The JEOL-provided interface for controlling the TVIPS camera. It handles live view, alignment, and acquisition parameters for both imaging and diffraction, and exports raw TIFF outputs.
- **Panta Rhei** Manages the Dectris ELA hybrid-pixel detector for SAED acquisition. It interfaces directly with the camera to automate exposure, frame rate, and synchronization with the TEM beam.
- **PETS2** Performs crystal cell reconstruction from diffraction patterns, replacing the older Serial ED workflow. PETS2 refines lattice parameters and orientation based on spot positions.
- **Jana** Provides visualization and analysis of reconstructed unit cells. It renders three-dimensional crystal structures and allows comparison with known phases.

- **Microsoft Excel** Although proprietary, Excel remains a widespread, familiar tool in many laboratories. It was employed for quick statistical analysis of measured particle sizes and compositional data extracted from EDS maps. Its use here reflects a pragmatic “legacy” choice for rapid summary plots and data checks, with the expectation that future workflows will migrate to fully open-source platforms.
- **ImageJ** An open-source image-processing platform employed for tasks such as contrast adjustment, scale-bar overlay, and line-profile extraction on both TEM and STEM images.

3.2.3 Bucket and file-naming convention under the NFFA-DI Research Data Policy

NFFA-DI serves as a national research infrastructure: users gain access by submitting a proposal, which is assigned a unique *Proposal ID*. All processed NeXus files generated in this work are stored in the **Overarching FAIR Ecosystem for Data (OFED)**, NFFA-DI’s centralized “datalake” service hosted on the Orfeo HPC cluster. OFED runs on Kubernetes with a MinIO backend, exposing S3-compatible *buckets*—top-level containers where each object (here, a NeXus file) is addressed by a unique key. Raw TIFFs and other instrument outputs remain on the traditional Orfeo filesystem until conversion, while the NeXus outputs are pushed into OFED for controlled access, metadata indexing, and long-term preservation [5] (see Chapter 4 for details on MinIO buckets and access).

In accordance with Section 3 of the NFFA-DI Research Data Policy (rev. 25 March 2025) [14], our naming scheme ensures:

- §3.1.4: NeXus files reside in OFED buckets with fine-grained access controls;
- §3.1.6: any non-NeXus raw files remain interoperable on Orfeo and can be wrapped later;
- §3.1.8: every filename begins with the bucket prefix and a self-describing identifier.

1. Bucket naming

Each approved Proposal (ID [proposal_id]) is assigned an OFED bucket:

nffa-di_[proposal_id]

This also meets §3.1.7 by allocating at least 10 GB of dedicated storage per Proposal.

2. Filename composition

Within each bucket, filenames follow the pattern:

`nffa-di_[proposal_id]_LAME_[tech]_[hw]_[sample]_[YYYYMMDD]_[run]_[####].[ext]`

where:

- **Fixed prefix** `nffa-di_[proposal_id]_LAME_` ties each file unambiguously to the correct bucket and the LAME facility.
- **UO_internal_id** A sequence of six segments—measurement mode, hardware tag, sample code, acquisition date, run label, and a zero-padded counter—ensures both human readability and machine parsability.

By structuring buckets and filenames this way, we guarantee policy compliance, clear organization, and future-proofing of our EM datasets in OFED.

Raw output files (TIFFs, maps, spectra) remain on the Orfeo cluster’s shared filesystem under a separate directory structure, preserving original formats and avoiding duplication in OFED until they are wrapped as NeXus containers.

3. Segment definitions and rationale

Segment	Req’d	Example	Rationale
<code>tech</code>	yes	TEM, SAED, STEM-EDS	Indicates measurement mode (TEM imaging vs. SAED vs. elemental map).
<code>hw</code>	yes	F200, ELA, EDS	Disambiguates instrument or detector used.
<code>sample</code>	yes	SnNiNPs, NiSnNPs	Short, no-spaces tag to identify the sample system.
<code>YYYYMMDD</code>	yes	20241223	UTC acquisition date; ensures chronological sorting.
<code>run</code>	opt.	v001, lp002	Grouping label (view number, line-profile ID, tilt-series index).
<code>####</code>	yes	0001, 0104	Four-digit, zero-padded counter for uniqueness.
<code>[ext]</code>	yes	tiff, map, pts	Original file extension, preserving native format.

Table 3.1: Breakdown of the *UO_internal_id* segments with their purpose.

By structuring filenames this way, we achieve:

- **Policy compliance:** direct mapping to §3.1.8.

- **Human readability:** anyone can see technique, date, sample and sequence at a glance.
- **Machine parsability:** fixed-width and hyphen/underscore separators allow automated cataloguing scripts.
- **Future-proofing:** the same identifier can become the NeXus entry name when data are wrapped.

3.2.4 Concrete example

Resulting bucket tree for Proposal #1234

```
nffa-di_1234/
|-- RAW/
|   |-- SAED/
|       |-- nffa-di_1234_LAME_SAED_ELA_SnNiNPs_20241223_v000_0001.tiff
|       |-- nffa-di_1234_LAME_SAED_ELA_SnNiNPs_20241223_v000_0002.tiff
|       '-- ...
|   |-- STEM-EDS/
|       |-- nffa-di_1234_LAME_STEM-EDS_EDS_SnNiNPs_20250214_v001_0104.map
|       |-- nffa-di_1234_LAME_STEM-EDS_EDS_SnNiNPs_20250214_v001_0105.pts
|       '-- ...
|   '-- TEM/
|       |-- nffa-di_1234_LAME_TEM_F200_SnNiNPs_20241223_0001.tif
|       |-- nffa-di_1234_LAME_TEM_F200_SnNiNPs_20241223_0002.tif
|       '-- ...
```

This layout satisfies:

- §3.1.4 – everything hosted on OFED;
- §3.1.6 – non-NeXus raw files remain interoperable yet may be wrapped later in NeXus containers using the same *UO_internal_id*;
- §3.1.8 – file names begin with the bucket prefix and follow the prescribed convention.

Chapter 4

A Django web application for FAIR electron-microscopy data

As anticipated in Chapter 1, which introduced the FAIR-by-design principles and overall data management workflow, this web application transforms what used to be a loose collection of scripts into a unified portal for both metadata capture and file handling. It serves two complementary roles: a lightweight Electronic Laboratory Notebook (ELN) for inputting sample details and experiment metadata, and a streamlined interface for data upload, NeXus conversion, and long-term storage.

Rather than adopting a full-featured electronic lab notebook such as *elabFTW*—an open-source ELN providing experiment versioning and collaboration but deemed too elaborate by LAME staff—this Django portal fulfills both ELN and data pipeline functions in one. First, it allows researchers to record sample metadata, experiment descriptions, and instrument settings via intuitive web forms, ensuring that essential context accompanies each dataset. Second, it replaces a loose collection of standalone scripts with a single entry point for TIFF upload, guided metadata mapping, automated NeXus NXem conversion, and seamless storage in MinIO’s S3-compatible buckets [15], all secured through facility-wide Single Sign-On with Authn-tik.

Because each dataset is self-contained in its NeXus file, no relational database is required: the HDF5 containers themselves serve as the canonical records. A lightweight catalogue (e.g. SQLite or JSON) can be added later without altering the user interface. By focusing on these core capabilities and integrating directly with OFED’s Kubernetes-hosted services, the application remains straightforward to deploy, maintain, and extend as LAME’s FAIR data ecosystem continues to evolve. **The application is still in active development and may evolve as new requirements and features**

emerge.

The FAIR-by-design data flow supported by the application is summarized below:

- Instrument (Lab) produces the raw experimental data (e.g. TIFF images, spectra, diffraction patterns).
- Data is temporarily saved to local storage on the instrument's computer.
- The web application, running in a container on the ORFEO cluster, is accessed from the lab to browse this local storage and select files for processing.
- Selected files are converted into NeXus format and saved into OFED MinIO buckets, ensuring structured, FAIR-compliant storage.
- Raw files (non-NeXus) are optionally copied to ORFEO outside OFED, allowing separate archival of native instrument formats.

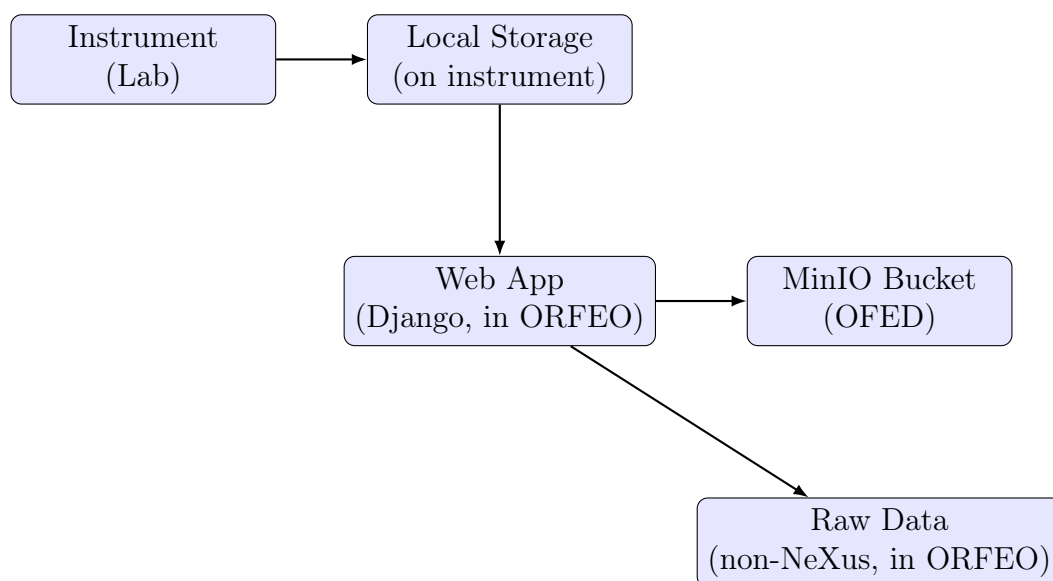


Figure 4.1: Overview of the FAIR-by-design data pipeline implemented at LAME.

4.1 Overall user experience

Figure 4.3 shows the page that researchers use to submit a new experiment. Besides selecting one or more TIFF files, the user enters free-text fields such as operator name and sample identifier, and chooses a JSON mapping that tells the system how to translate TIFF tags into NeXus fields.

After uploading, the user is redirected to a listing of processed datasets (Figure 4.4); clicking on a filename opens a preview page that renders the first image stored inside the NeXus file (Figure 4.5). A *Download* button delivers either the raw `.nxs` or a PNG extracted on-the-fly, depending on the query parameter.

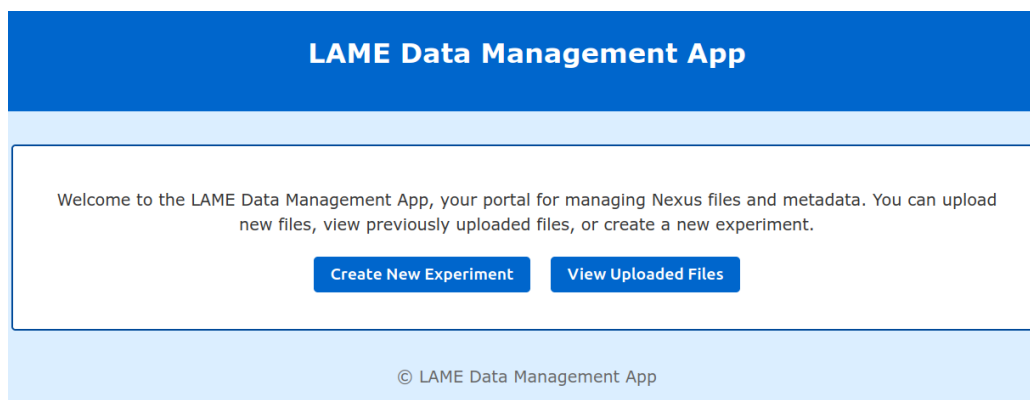


Figure 4.2: Homepage of the LAME Data Management App, showing the main navigation buttons for creating a new experiment and viewing uploaded files.

4.2 Upload workflow

1. A researcher opens the “New Experiment” page served by `new_experiment.html`. This form asks for operator name, instrument, sample material, and lets the user choose one or more TIFF files.
2. When the form is submitted, Django executes `new_experiment_view()` in `views.py`. That function
 - (a) writes the first TIFF to `/tmp`;
 - (b) loads a user-selected mapping (`ED_mapping.json` or `TVIPS_mapping.json`);
 - (c) calls the NeXus builder to create an `.nxs` file;

Create New Experiment

Fill out the metadata and upload your TIFF image(s).
Required NeXus fields coming from the form are marked with *.

Experiment Type*

Select experiment type

Operator Name*

Description*

Sample Material*

Select material

Sample identifier (required):

Preparation date:

04 / 26 / 2025

Comma-separated atom symbols (e.g. Fe,Ni,Al):

Initial Composition

Final Composition

Hypothetical Composition

Instrument Name*

Instrument Location

Trieste

Select TIFF images*

Metadata Mapping*

Select mapping

Create NeXus File

Back to Homepage

© 2025 LAME Data Management App

Figure 4.3: The *New Experiment* page. Mandatory fields are enforced by standard HTML validation.

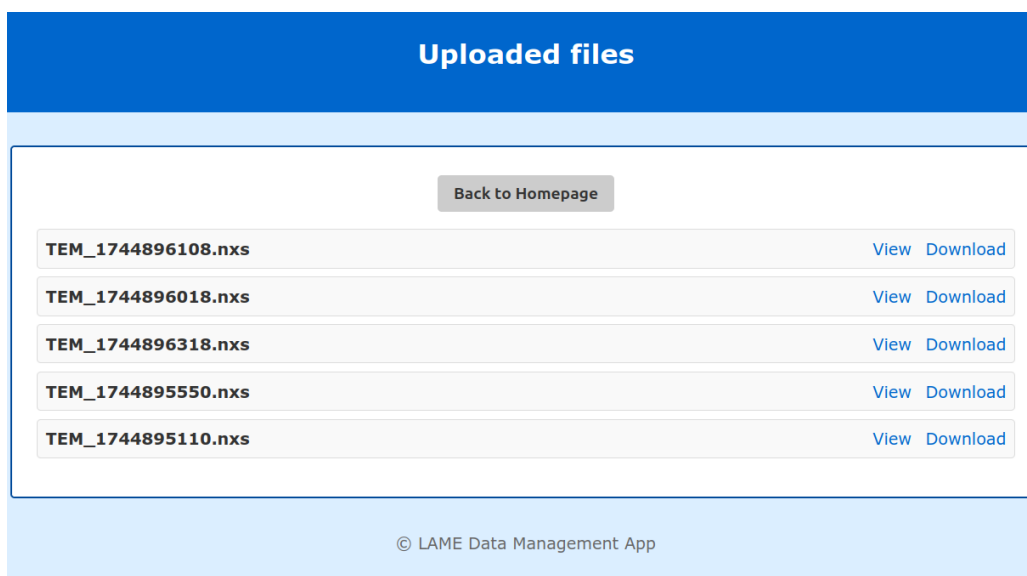


Figure 4.4: Browsing previously uploaded datasets. The list is generated directly from the object keys found in the MinIO bucket.

- (d) either moves the file to `local_storage/` (dev branch) or streams it to MinIO (production branch).

A condensed version of that code path is shown below:

```
image_file = request.FILES['image_files']
tiff_path = f"/tmp/{image_file.name}"
with open(tiff_path, 'wb') as out:
    for chunk in image_file.chunks():
        out.write(chunk)

build_nexus_from_tiff_TEM_ED(
    tiff_path,
    mapping_json_path,
    out_nxs=f"/tmp/TEM_%d.nxs" % time.time(),
    extra_fields=extra_fields, # values from the HTML form
)

# switchable storage backend
if settings.USE_MINIO:
    s3.upload_fileobj(open("/tmp/TEM_%d.nxs" % ts, "rb"),
                      bucket="lame-bucket",
                      Key=unique_key)
else:
    shutil.move("/tmp/TEM_%d.nxs" % ts,
                Path(settings.BASE_DIR) / "local_storage")
```

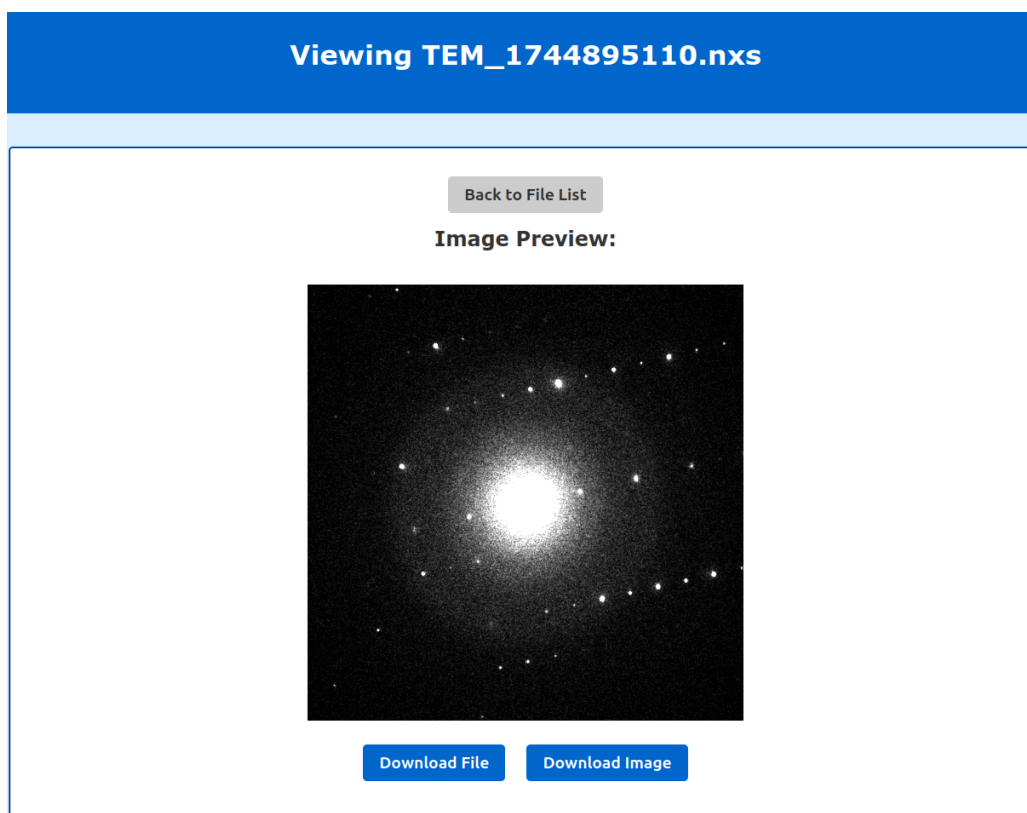


Figure 4.5: Preview page for a single NeXus file. The image is extracted with `h5py` and normalised for display.

4.3 Inside the NeXus builder

The core of our conversion is in `nexus_integration.py`, which performs two complementary tasks:

1. **Automatic metadata extraction** All TIFF header tags are flattened into a simple dictionary and then mapped—via the user-selected JSON file—into the corresponding NXem fields. This covers any metadata that the camera or microscope wrote directly into the TIFF.
2. **Manual metadata injection** Required NeXus fields that are *not* present in the TIFF header (sample identifier, operator name, instrument settings, etc.) are supplied by the user through the web form and passed as `extra_fields` to the builder.

First, the helper function `flatten_dict()` turns nested TIFF tags or JSON values into a flat map:

```
def flatten_dict(d, parent_key="", sep="."):
    items = []
    if isinstance(d, dict):
        for k, v in d.items():
            new_key = f"{parent_key}{sep}{k}" if parent_key else k
            items.extend(flatten_dict(v, new_key, sep=sep).items())
    elif isinstance(d, list):
        for i, v in enumerate(d):
            new_key = f"{parent_key}[{i}]"
            items.extend(flatten_dict(v, new_key, sep=sep).items())
    else:
        items.append((parent_key, d))
    return dict(items)
```

Next, the mapping JSON (e.g. `ED_mapping.json`) tells the code:

“When you see the TIFF tag `ImageDescription.camera.start_time`, write it into `/NXentry/measurement/events/NXevent_data_em/start_time`.”

Any TIFF-derived tags not in the mapping are skipped, while the builder then embeds the image array itself and applies instrument-specific groups.

Simultaneously, fields that the TIFF cannot supply are collected in the view and handed to the builder as `extra_fields`. For example, in `views.py`:

```
# Collect form values that must be filled by the user:
extra_fields = {
    "operator_name": request.POST["operator_name"],
    "description": request.POST["description"],
    "material": material,
    "sample_identifier": request.POST["sample_identifier"],
    "preparation_date": request.POST["preparation_date"],
    "atom_types": request.POST["atom_types"],
    "instrument_name": request.POST["instrument_name"],
    "instrument_location": request.POST["instrument_location"],
    # any other manual entries
}

# Build the NeXus file, merging TIFF metadata and extra_fields:
build_nexus_from_tiff_TEM_ED(
    tiff_path,
    mapping_json_path,
    out_nxs=nexus_file_tmp,
    extra_fields=extra_fields,
)
```

Inside `_build_scaffold()`, the code loops over the flattened TIFF map, applies the JSON mapping, and writes those fields into the HDF5. It then iterates over `extra_fields`:

```
# After inserting image and mapped metadata...
for key, value in (extra_fields or {}).items():
    nxentry[key] = value      # write manual fields into NXentry
```

This dual strategy—automatic extraction using the JSON mappings (see Appendix B) combined with manual supplementation via the web form—ensures that all metadata emitted by the instrument and any additional context required by the NXem schema are included in the final NeXus file.

4.4 NeXus file validation

To ensure that every NeXus file conforms to the NXem application definition and catches missing or mis-typed fields, we validate each generated file with the command-line tool `nxvalidate`[10]. This tool requires a local clone of the FAIRmat-NFDI NeXus definitions repository[16]. A typical workflow is:

```
# 1. Clone the definitions repo once:
```

```
git clone https://github.com/FAIRmat-NFDI/nexus_definitions.git \
    ~/nexusvalidation/nexus_definitions
```

```
# 2. Validate your .nxs file against those definitions:
```

```
nxvalidate -l /path/to/nexus_definitions/definitions my_experiment.nxs
```

Here:

- `-l /nexusvalidation/nexus_definitions/definitions` points to the folder containing the official `*.nxd1.xml` files from the cloned repository.
- `my_experiment.nxs` is the file you wish to check.

Any structural or metadata errors are reported with full context, for example:

```
definition=NXem.nxd1.xml
message="Required attribute URL missing"
nxd1Path=/NXentry/definition
sev=error
dataPath=/entry/definition
dataFile=my_experiment.nxs
```

This output tells you exactly which NeXus definition failed, where in your file the problem occurs, and the severity of the error. By incorporating `nxvalidate` into the upload pipeline, we guarantee that every dataset meets the official NXem specification before it is made available (see Listing A.1 in Appendix A).

4.5 MinIO: the object store that replaces a file server

In a conventional setup, files live on a shared filesystem arranged into folders and subfolders. Users navigate a POSIX hierarchy like `/data/experiments/2025/` and request downloads via SSH or CIFS. While familiar, this approach can struggle when datasets grow into the tens or hundreds of gigabytes:

- Adding more storage often means downtime to expand or migrate disks.
- If a drive fails, recovering data can be slow or incomplete.
- Serving large files through the web server itself ties up network and CPU resources.

Object storage takes a different approach. Instead of folders and files, every piece of data is stored as an *object* identified by a unique key. Objects live in *buckets*, which act like top-level containers (somewhat like drive letters or volume names, rather than nested directories). This design simplifies scaling, replication, and access control, and is the foundation of modern “cloud-native” storage systems.

MinIO is an open-source, self-hosted object storage server that speaks the same HTTP API as Amazon S3. By deploying MinIO in our lab, we gain:

1. **Flat namespace** — Every file (object) lives in a bucket and is fetched by its key. There is no need to manage deep directory trees—keys can themselves contain “/” characters if you want pseudo-folders.
2. **Horizontal scaling** — You can add new disks or even whole servers to the MinIO cluster while it is running. Data is automatically balanced across all nodes without manual reconfiguration.
3. **Fault tolerance** — MinIO supports erasure coding and replication. This means that even if one or more drives or servers fail, the data remains intact and accessible.
4. **Pre-signed URLs** — Instead of routing every download through Django (which would consume CPU and memory), the application issues a short-lived, cryptographically signed URL. The user’s browser then downloads the object directly from MinIO, offloading the traffic from the web server.

5. **API compatibility** — Because MinIO uses the S3 API, we can use standard libraries (such as `boto3` in Python) without change. This also makes any future migration to a cloud provider (AWS, Google Cloud, or others) trivial.

To connect Django to MinIO, we use the `django-storages` package. A small set of settings in `settings.py` is all that is required:

```
# Use the S3Boto3Storage backend for all FileFields
DEFAULT_FILE_STORAGE = 'storages.backends.s3boto3.S3Boto3Storage'

# MinIO endpoint (S3-compatible URL) and bucket name
AWS_S3_ENDPOINT_URL = "https://s3.k3s.virtualorfeo.it"
AWS_STORAGE_BUCKET_NAME = "lame-bucket"

# Credentials for a MinIO user with write/read access
AWS_ACCESS_KEY_ID = "lame-app-user"
AWS_SECRET_ACCESS_KEY = os.getenv("MINIO_SECRET_KEY")
```

With these settings, any Django model or manual upload that writes to a `FileField` will transparently push the data into MinIO. Likewise, downloading a file uses the same storage handler to generate pre-signed URLs for efficient, direct access.

4.6 Authentik: our facility's central Identity Provider

At Area Science Park, all web applications—including this LAME portal—use **Authentik**[\[17\]](#) as the single source of truth for user accounts. Rather than creating and maintaining separate accounts in Django, every researcher and staff member already has credentials managed by Authentik. Our application for the moment delegates login to that existing system.

How login works

1. **User navigates to the portal.** They arrive at a protected page and must authenticate before proceeding.
2. **Redirect to Authentik.** Django (via the `mozilla_django_oidc` library) sends the browser to Authentik's login URL. Since Authentik is our institution's Identity Provider (IdP), users enter the same username/password they use for email, HPC, and other services.

3. **Multi-factor and policy enforcement.** Authentik applies our facility’s rules—password strength, two-factor prompts, group-based approvals—before issuing an authentication token.
4. **Token exchange.** On success, Authentik redirects back to the application’s callback endpoint (`/oidc/callback/`) with a signed ID token (a JSON Web Token).
5. **Session creation.** Django verifies the token’s signature, expiry, and issuer, then creates a local session cookie so the user remains logged in for subsequent requests.

Because all account lifecycle events (password resets, deactivations, group assignments) happen in Authentik, our portal never stores or processes raw passwords. This means:

- No separate “user admin” interface is needed inside Django.
- Account suspension or role changes in Authentik immediately apply to the portal.
- Single Sign-On (SSO) experience is seamless across all services in the park.

Logging out from both ends

To log the user out of both the portal and Authentik in one step, we use an “end session” redirect:

```
def my_logout_view(request):
    # 1. End the local Django session
    logout(request)

    # 2. Build the Authentik logout URL,
    #     including the ID token as a hint
    id_token = request.session.get('oidc_id_token', '')
    end_url = (
        f"{settings.OIDC_OP_LOGOUT_ENDPOINT}"
        f"?id_token_hint={id_token}"
        f"&post_logout_redirect_uri={settings.OIDC_LOGOUT_REDIRECT_URL}"
    )

    # 3. Redirect so Authentik also terminates its session
    return redirect(end_url)
```

This ensures that when a user logs out, they are signed out of the LAME portal *and* their Authentik SSO session, protecting both local and central access in a single action.

4.7 Production deployment with containers and Kubernetes

To move from a development prototype to a robust production service, we package the application as a Docker container and deploy it on Kubernetes. Containers ensure that the same software image—with all its dependencies—runs identically on any host, improving reproducibility and simplifying updates. Kubernetes then manages scaling, rolling upgrades, and high-availability across multiple instances.

Containerizing the application

We begin by building a Docker image from the project's `Dockerfile`, which installs Python, Django, and the required libraries, copies the source code, and configures the web server. The image is tagged and pushed to a container registry:

```
podman build \
  -t registry.gitlab.com/area7/nffa-di/lame_fair_by_design:v1.0 \
  .
podman push \
  registry.gitlab.com/area7/nffa-di/lame_fair_by_design:v1.0
```

Kubernetes manifests and secrets

All sensitive values (Django secret key, Authentik client secret, MinIO credentials) are stored in a Kubernetes Secret (`lame-fair-secrets.yaml`). Configuration parameters (OIDC endpoints, bucket names) live in a ConfigMap (`lame-fair-config.yaml`). The Deployment manifest describes the container image, environment variables, and the desired number of replicas, while Service and Ingress manifests expose the application via HTTPS.

Applying the configuration

Once the image and manifest files are ready, deployment proceeds with a few `kubectl` commands:

```
# 1. Create a dedicated namespace
kubectl create namespace lame-fair

# 2. Create secrets and config
kubectl apply -f lame-fair-secrets.yaml
kubectl apply -f lame-fair-config.yaml
```

```
# 3. Deploy the app, service & ingress
```

```
kubectl apply -f service.yaml
kubectl apply -f nginx-config.yaml
kubectl apply -f ingress.yaml
kubectl apply -f deployment.yaml
```

```
# 4. Roll out updates when the image changes
```

```
kubectl rollout restart deployment/lame-fair-app -n lame-fair
```

With this setup, Kubernetes automatically schedules pods onto available nodes, restarts failed containers, and can scale the number of replicas based on load. Logs and metrics can be aggregated centrally, and rolling updates ensure zero-downtime deployments. This container-based approach lays a solid foundation for a resilient, maintainable production service.

4.8 Current FAIR status

- **Findable** — datasets can be discovered within the NFFA-DI infrastructure via bucket listings and structured filenames (see Chapter 3), ensuring internal FAIR compliance (cf. Section 2.1.1).
- **Accessible** — every object can be fetched through a signed URL; Authentik guarantees that only authorised users can request such URLs.
- **Interoperable** — data are wrapped in NeXus NXem, a self-describing HDF5 layout widely understood by open-source tools.
- **Reusable** — instrument settings, sample descriptors and even the original TIFF header are all embedded in the NeXus file, so the measurement context is preserved.

4.9 Next milestones

Looking ahead, several enhancements will expand the platform’s functionality and integration:

1. **Full batch-upload support.** Convert every TIFF in a multi-file submission into its own NeXus file (or group related TIFFs into multi-entry NeXus containers), rather than processing only the first file.
2. **Unified storage backend.** Merge the “local storage” and “MinIO” code paths behind a single configuration flag in `settings.py`, simplifying deployment and maintenance.

3. **Lightweight search index.** Introduce a simple on-disk catalogue (SQLite or JSON) that periodically scans MinIO object metadata and supports fast keyword searches as the dataset count grows.
4. **Refactoring and modularization.** Clean up and reorganize the codebase: extract shared logic into reusable modules, consolidate view branches for MinIO vs. local storage, and improve naming and documentation for easier future development.
5. **End-to-end testing.** Build automated test suites covering the full workflow: Authentik login, file upload, NeXus conversion, MinIO upload, signed-URL download, and validation via `nxvalidate`.
6. **Local NOMAD OASIS integration.** Deploy an on-premises NOMAD Archive OASIS instance[18], connect it directly to our MinIO bucket for seamless data ingestion, and (in the future) leverage Authentik as the federated identity provider so that users authenticate once for both the LAME portal and NOMAD.

Even in its model-free state, the application already delivers a fully traceable pipeline:

**Upload → NXem conversion → Object storage → Secure
download**

All heavy lifting is off-loaded to MinIO and Authentik, keeping the web codebase slim and easy to maintain.

Chapter 5

Conclusion

A complete FAIR-by-design data management workflow for electron microscopy has been implemented at the LAME facility, streamlining the journey from raw data acquisition to long-term, accessible storage. From the moment TIFF images and diffraction patterns are produced by the JEOL JEM-F200, each dataset is systematically handled following the principles of findability, accessibility, interoperability, and reusability.

The core elements of the system are:

- **Policy-compliant file organization:** Each dataset is placed into an OFED bucket using a structured, self-descriptive naming convention that encodes key metadata such as proposal identifiers, instrument and detector types, sample tags, acquisition dates, and sequence counters.
- **Metadata extraction and standardization:** TIFF headers are automatically parsed, flattened into a uniform dictionary, and selectively mapped into NeXus NXem fields via JSON definitions. Any essential metadata absent from the instrument outputs—such as sample identifiers, operator names, preparation dates, and laboratory conditions—is collected through a web form and merged into the NeXus file at build time.
- **Centralized and secure data handling:** A Django web application guides users through metadata entry and file uploads. Completed NeXus files are stored in MinIO object storage, using pre-signed URLs to offload heavy file transfers and ensure efficient, authenticated access.
- **Identity management and validation:** Integration with Authentik allows Single Sign-On (SSO) based on the facility’s existing identity infrastructure, eliminating the need for local account management. Every

NeXus file is rigorously validated against the official NXem schema using `nxvalidate`, ensuring conformance before the datasets are made available to users.

- **Scalable and reproducible deployment:** The full application stack is containerized and deployed on Kubernetes, supporting reliable scaling, rolling updates, and simplified maintenance over time.

These elements come together to form a seamless, traceable pipeline:

Acquisition → Conversion → Validation → Storage → Access

The results achieved so far represent a substantial improvement in the management and preservation of electron microscopy data at LAME. Looking ahead, several enhancements are planned:

- Extending the application to handle full multi-file uploads in a single session.
- Consolidating local filesystem and MinIO storage modes behind a unified backend configuration.
- Adding a lightweight search and indexing service to support metadata-based queries.
- Deploying a local NOMAD OASIS archive instance and integrating it with MinIO and Authentik.
- Refactoring and optimizing the codebase to facilitate future extensions and maintenance.

By embedding FAIR principles into every step—from the moment of data acquisition to long-term archival—this infrastructure lays a strong foundation for sustainable and reusable scientific data at LAME. Moreover, it offers a replicable model that can be adapted by other microscopy laboratories and research facilities seeking to modernize their data workflows without sacrificing usability or scientific integrity.

Appendix A

Example NeXus tree

Listing A.1: Contents of /NXentry in a NeXus file

```
NXentry/  
  SAMPLE/  
    atom_types  
    identifier_sample  
    is_simulation  
    name  
    physical_form  
    preparation_date  
  atom_types  
  coordinate_system_1/  
    handedness  
    origin  
    type  
  definition  
  description  
  experiment_identifier  
  final_composition  
  hypothetical_composition  
  image_2d/  
    data  
    signal  
  initial_composition  
  instrument_location  
  instrument_name  
  material  
  measurement/
```

```
events/  
  event_data/  
    instrument/  
      detector/  
        count_time  
        mode  
        x_pixel_size  
      ebeam_column/  
        electron_source/  
          voltage  
        filter/  
          mode  
        operation_mode  
      stage/  
        position  
        rotation  
        tilt1  
        tilt2  
        x_position  
        y_position  
        z_position  
  instrument/  
    ebeam_column/  
      electron_source/  
        location  
      name  
      type  
operator_name  
preparation_date  
sample_identifier  
start_time  
start_time_date  
start_time_time
```

Appendix B

JSON mappings for TIFF → NXem

In order to automate the transfer of metadata from raw TIFF headers into the standardized NeXus NXem structure, we maintain two JSON “mapping” files. Each mapping is a simple dictionary: the keys are the TIFF tag names as they appear in the file header, and the values are the dot- or slash-delimited paths of the corresponding fields in the NeXus hierarchy. During conversion, our code reads the TIFF tags, looks them up in these mappings, and writes the values into the correct group or dataset in the output `.nxs` file.

ED_mapping.json

Listing B.1: ED_mapping.json (with longer lines split into segments)

```
{
  "ImageDescription.camera.start_time":
    "NXentry.measurement.events.NXevent_data_em.start_time",

  "ImageDescription.camera.exposure_time":
    "NXentry.measurement.events."
    "NXevent_data_em.instrument.NXdetector.exposure_time",

  "ImageDescription.camera.pixel_size":
    "NXentry.measurement.events."
    "NXevent_data_em.instrument.NXdetector.x_pixel_size",

  "ImageDescription.stage.x":
    "NXentry.measurement.events."
```

```

    "NXevent_data_em.instrument.stage.x_position",

    "ImageDescription.stage.y":
      "NXentry.measurement.events."
      "NXevent_data_em.instrument.stage.y_position",

    "ImageDescription.stage.tilt_x":
      "NXentry.measurement.events."
      "NXevent_data_em.instrument.stage.tilt1",

    "ImageDescription.stage.tilt_y":
      "NXentry.measurement.events."
      "NXevent_data_em.instrument.stage.tilt2",

    "ImageDescription.filter.mode":
      "NXentry.measurement.events."
      "NXevent_data_em.instrument.ebeam_column.filter.mode",

    "ImageDescription.electron_gun.voltage":
      "NXentry.measurement.events."
      "NXevent_data_em.instrument.ebeam_column.electron_source.voltage",

    "ImageDescription.electron_gun.voltage_offset":
      "NXentry.measurement.events."
      "NXevent_data_em.instrument.ebeam_column."
      "electron_source.voltage_offset",

    "ImageDescription.camera.simplon_parameters.trigger_mode":
      "NXentry.measurement.events."
      "NXevent_data_em.instrument.NXdetector."
    "simplon_parameters.trigger_mode",

    "ImageDescription.repo_id:NXentry.experiment_identifier"
  }

```

TVIPS_mapping.json

Listing B.2: TVIPS_mapping.json (with longer lines split into segments)

```

{
  "TVIPS.Date:NXentry/start_time_date",

```

```
"TVIPS.Time:NXentry/start_time_time",

"TVIPS.HighTension":
  "NXentry/measurement/events/"
  "event_data/instrument/ebeam_column/electron_source/voltage",

"TVIPS.ExposureTimeV1":
  "NXentry/measurement/events/"
  "event_data/instrument/detector/count_time",

"TVIPS.PhysicalPixelSize":
  "NXentry/measurement/events/"
  "event_data/instrument/detector/x_pixel_size",

"TVIPS.TemStagePosition[0]":
  "NXentry/measurement/events/event_data/instrument/stage/x_position",

"TVIPS.TemStagePosition[1]":
  "NXentry/measurement/events/event_data/instrument/stage/y_position",

"TVIPS.TemStagePosition[2]":
  "NXentry/measurement/events/event_data/instrument/stage/z_position",

"TVIPS.SpecimenTiltAngle":
  "NXentry/measurement/events/event_data/instrument/stage/tilt1",

"TVIPS.SpecimenTiltDirection":
  "NXentry/measurement/events/event_data/instrument/stage/tilt2",

"TVIPS.ImageMode":
  "NXentry/measurement/events/"
  "event_data/instrument/ebeam_column/filter/mode",

"TVIPS.ImageName:NXentry/experiment_identifier"
}
```

Bibliography

- [1] Area Science Park, “Area science park: A world in progress,” 2025.
<https://www.areasciencepark.it/en/>.
- [2] Area Science Park, “Lame – laboratory of electron microscopy,” 2025.
<https://www.areasciencepark.it/en/research-infrastructures/innovative-materials/lame-electron-microscopy-laboratory/>.
- [3] FAIRmat Consortium, “Fairmat – fair data infrastructure for condensed-matter physics and the chemical physics of solids,” 2024.
<https://www.fairmat-nfdi.eu/fairmat/>.
- [4] NFFA-DI, “New master in data management and curation kicks off,” Mar., 2024. <https://nffa-di.it/en/news/new-master-in-data-management-and-curation-kicks-off/>.
Accessed: 2025-03-31.
- [5] F. Bazzocchi, “Data management in nffa-di : Ofed (overarching fair ecosystem for data),” Sept., 2024.
<https://doi.org/10.5281/zenodo.13773924>.
- [6] The HDF Group, “Fairmat and hdf5: Experiments – calculations – integrate the experimental world,” July, 2021.
https://www.hdfgroup.org/wp-content/uploads/2021/07/20210707_FAIRmat_hdf5.pdf. Presentation slides.
- [7] L. Metilli, M. Francis, M. Povey, A. Lazidis, S. Marty-Terrade, J. Ray, and E. Simone, “Latest advances in imaging techniques for characterizing soft, multiphasic food materials,” *Advances in Colloid and Interface Science* **279** (2020) 102154. <https://www.sciencedirect.com/science/article/pii/S0001868619304567>.
- [8] NeXus International Advisory Committee, “Nexus file format specification manual,” 2024.
<https://manual.nexusformat.org/fileformat.html>.

- [9] NXcanSAS Working Group, “Introduction to the nexus data format,” 2012. https://www.cansas.org/NXcanSAS/_downloads/nexus.pdf.
- [10] FAIRmat-NFDI, “Validate a nexus file with nxvalidate,” 2025. <https://fairmat-nfdi.github.io/pynxtools/how-tos/validate-nexus-file.html>.
- [11] F. Bazzocchi, “Nexus and electron microscopy.” Presentation, nffa-di, 2024. https://nffa-di.it/media/n2nn0qo1/8_nexus-and-electron-microscopy_bazzocchi.pdf.
- [12] NeXus International Advisory Committee, *Electron Microscopy Structure*, 2024. https://manual.nexusformat.org/classes/contributed_definitions/em-structure.html. Accessed: 31 March 2025.
- [13] JEOL USA, Inc., “JEM-F200 "F2" Multipurpose Analytical S/TEM,” 2025. <https://www.jeolusa.com/PRODUCTS/Transmission-Electron-Microscopes-TEM/200-kV/JEM-F200-F2>. Accessed: 1 April 2025.
- [14] NFFA-DI, “Research data policy,” 2025. <https://nffa-di.it/en/research-data-policy/>.
- [15] P. Relekar, “Minio: A high performance, distributed object storage system,” 2025. <https://dsrs.illinois.edu/blog/minio>. Accessed: 1 April 2025.
- [16] FAIRmat-NFDI, “Nexus definitions repository,” 2025. https://github.com/FAIRmat-NFDI/nexus_definitions/.
- [17] authentik Security Inc., *authentik Documentation*, 2025. <https://docs.goauthentik.io/docs/>.
- [18] NOMAD Laboratory, “Nomad oasis — archival service for materials science data,” 2025. <https://nomad-lab.eu/nomad-lab/nomad-oasis.html>.