

# ENHANCING DATA MANAGEMENT WITH SciCat: A COMPREHENSIVE OVERVIEW OF A METADATA CATALOGUE FOR RESEARCH INFRASTRUCTURES

C. Minotti<sup>†</sup>, A. Ashton, S. Bliven, Paul Scherrer Institute, Villigen, Switzerland  
F. Bolmsten, M. Novelli, T. Richter, European Spallation Source, Lund, Sweden  
D. McReynolds, Lawrence Berkeley National Laboratory, Berkeley, CA, USA  
M. Leorato, D. Van Dijken, MAX IV Laboratory, Lund, Sweden  
L. Schemilt, Rosalind Franklin Institute, Didcot, UK

## Abstract

In today's data-driven scientific research landscape, the management of vast amounts of data, together with low-effort search capabilities, easy access and retrieval of acquired data is crucial for enabling successful collaborations, empowering all stakeholders to contribute to data enrichment and fostering scientific advancement. Metadata catalogues, where fields are extracted from datasets and inserted into a searchable database, are pivotal in accessing scientific data in such a landscape. Data FAIRness qualities have become more and more important as an increasing number of publishing entities require transparency in published results and their provenance. Metadata catalogues help facilitate FAIR principles by enabling findability, and accessibility. With careful curation of metadata fields, they can play a vital role in interoperability.

We present SciCat a metadata catalogue designed to meet the needs of the community of scientists carrying out experiments and measurements. SciCat offers a scalable and flexible solution that empowers researchers to effectively manage, share, publish, and discover scientific datasets, thereby fostering collaboration, increasing data visibility and accelerating scientific progress.

## INTRODUCTION

Metadata is defined as the data providing information about one or more aspects of the data; it is used to summarize basic information about data that can make tracking and working with specific data easier. [1] It includes, among many, information about the source of the data, its acquisition process, responsible people and the location on a computer network where the data was created and collected.

Some metadata is auto-extracted from experimental data, while others may include unique quantitative and qualitative information produced or inferred post-experiment. This information is essential for future data utilization and may not be stored elsewhere. Metadata varies in format and standards across research fields and must capture data source dependencies. Additionally, the metadata storage process is adapted to each facility's existing infrastructure.

SciCat [2] has been developed with all these challenges in mind and the aim of being the central metadata storage

<sup>†</sup> carlo.minotti@psi.ch

solution, especially for Photon and Neutron facilities. It started as an in-kind contribution and as an open-source project between the European Spallation Source [3] and the Paul Scherrer Institut [4] within the European Photon and Neutron community. The goal was to develop a versatile metadata catalogue supporting researchers across their entire data lifecycle, as depicted in Fig. 1.

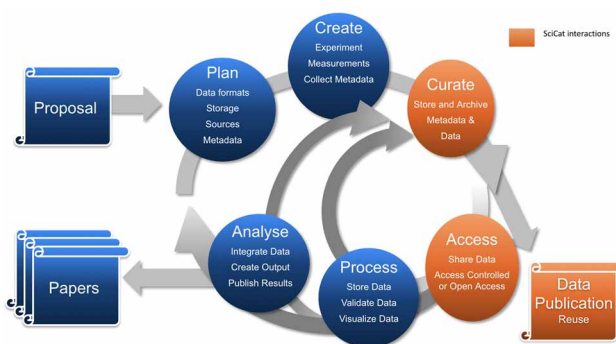


Figure 1: A typical researcher's data journey. The orange shapes are the interactions with SciCat.

Since its release, SciCat has been adopted by other facilities, including MAX IV Laboratory [5], Rosalind Franklin Institute [6], Advanced Light Source [7], the German DAPHNE project [8], Bundesanstalt für Materialforschung und prüfung [9] and the Shanghai Synchrotron Radiation Facility [10]. Additional facilities are rolling it out, for example, Synchrotron SOLEIL [11] and Deutsches Elektronen-Synchrotron [12]. The project has grown over the years with features and the support of dedicated developers and users. It now includes contributions from most of the adopting facilities. The project has recently released the latest version (4.x) which featured a complete rewrite of the backend and includes the corresponding upgrades of the frontend.

We start by introducing SciCat's core components and scalability for handling large volumes of metadata. Then, we cover search functions, metadata record creation (aka metadata ingestion), post-experiment metadata enrichment (aka data curation), and automated data management through *jobs*. The article will also discuss data publication, DOI minting, collaboration, and integration with generic and field-specific web search engines, such as the EOSC data portals [13-15], Google Dataset Search [16] and, for the latter, the PaNOSC data portal [17], developed during the PaNOSC [18] and ExPaNDS [19] EU projects.

To enhance data dissemination, we present Python [20] libraries for SciCat. Finally, we touch on future directions for SciCat as envisioned by the community.

## DESIGN OVERVIEW

The SciCat stack is organised following a microservice architecture, where each service can be containerised and configured to interact with the others and the pre-existing facility infrastructure, following standard TCP [21] communication protocols, such as HTTP(s) [22], AMQP/MQTT [23, 24] and Web-Sockets [25]. All SciCat services communicate with each other through HTTP(s).

The backbone of the ecosystem is the *backend* which relies on a Mongo Database [26], the connection to which must be configured as part of the setup. The *backend* is also responsible for defining the data model which formalises the scaffolding of the metadata fields, setting the required ones and leaving great flexibility for customisation.

### Data Model

Each SciCat entity is represented in a MongoDB data model [27], with a subset of fields controlled by *backend*-imposed validation rules (*high-level* fields). These fields are predetermined during design, managed by the system, and agreed upon by the SciCat community. They include general information like record name, creation date, or the associated experiment group. *High-level* fields can be mandatory or optional and have consistent meanings across SciCat adopters.

*Scientific metadata*, conversely, is a flexible structure where users can save the relevant scientific metadata that is important, for example, for the scientific process that the dataset has been acquired for. This design gives the required flexibility for science-specific data cataloguing and is of great importance when it comes, for example, to finding data or needing to check or replicate the experimental conditions. User-defined validation of the *scientific metadata* is being discussed in the core developer group.

The main SciCat entities, and consequently MongoDB collections [28] are *Datasets*, *Proposals*, *Samples* and *Instruments*.

**Datasets** a dataset is a collection of data organized in one or multiple files that have been acquired during the same data acquisition within a scientific experiment with a well-defined set of metadata that uniquely identifies the acquired data, experimental settings and conditions. They store the reference to the data files and additional information like the title and descriptions, ownership, samples, proposal, etc.

There are two dataset types: raw and derived. Raw datasets originate from experiments, such as measurements. Derived datasets result from post-processing raw data and their metadata tracks dependencies between the two.

**Samples** a sample is the physical piece of material that has been used during the experiment. A sample can be used

to acquire one or more raw datasets. Raw datasets can reference a Sample.

**Instruments** an instrument is the physical instrumentation that has been used and where the sample has been placed to acquire the data. As before, an instrument is usually used to acquire many datasets. Raw datasets can reference an Instrument.

**Proposals** in most research facilities, experimenters need to submit the type of experiment(s) that they want to run, which instruments they would like to use, which sample they will use during the experiments and further additional information. The comprehensive set of information is called a proposal and is managed by a dedicated platform. To simplify the management process, SciCat allows to store the relevant information of the proposals internally. Datasets can reference a Proposal.

### Technologies

At the time of writing, the SciCat stack consists, including the clients, of eleven services [29]. Of these eleven services, the core ones, and most widely used are the *backend*, encapsulating the core logic of the data catalogue, and the *frontend*, presenting the information to the users. In this section, we will only cover these two.

The *backend*, providing the RESTful API [30], user authentication and authorization, data management and the interface to the database, is the portion of SciCat running server-side which operates on and stores the information in the underlying database. It manages the user login and enforces access permissions. It is developed in Typescript [31] using the framework NestJS [32]. It uses MongoDB as a database and the Mongoose ORM [33]. The *backend* implements the classical REST API with a CASL [34] model for authorization. It supports local administrative accounts and OIDC authentication [35]. It supports all CRUD [36] operations to operate on metadata records. Records manipulation is controlled by the Data Transfer Object, one per SciCat entity, which maps them to the underlying data model.

The *backend* provides a Swagger UI [37] which exposes the endpoints, detailing the expected input and output formats and types. It also complies with the OpenAPI initiative [38].

It can be configured using environmental variables. Administrators can deploy it on bare OS or within a Docker [39] container, often using Kubernetes [40]. The basic setup involves configuring the database connection and the site acronym. Additional customization is possible but may require custom deployment and existing infrastructure, such as integration with an OIDC-supported Identity Access Management system [41].

The *frontend*, running in the client's browser, is a single-page [42] application created using the Angular [43] framework with TypeScript. It offers a user-friendly interface for searching *datasets* and presenting information based on user permissions. Users can also edit metadata and create datasets. If configured, it can trigger external actions, such as interactions with the backend *jobs* REST

API. It primarily displays information and relies on the backend REST API for computations.

### Scalability

SciCat harnesses the power of Docker containers and Kubernetes orchestration to achieve seamless scalability and efficiency. By embracing the containerization paradigm, SciCat empowers research institutions and laboratories with a sophisticated solution, enabling them to effortlessly store, organize, and retrieve metadata while adapting to the ever-evolving demands of data volumes.

In this ecosystem, Docker containers provide flexibility and consistency. These self-contained units encapsulate the entire metadata catalogue along with its dependencies, communicate through TCP, ensuring that each environment remains isolated and coherent. This approach not only simplifies the deployment process but also guarantees that the application functions uniformly across various setups.

SciCat publishes builds of its code as containers that can be deployed in a variety of container deployment solutions. The utilization of Kubernetes as the orchestration framework amplifies SciCat's capabilities. Kubernetes adapts the number of container instances depending on load, thereby optimizing performance and resource utilization. This elasticity becomes particularly crucial in the realm of scientific data, where demand fluctuations are common.

Message brokers serve as the gateway through which data flows seamlessly into the catalogue, ensuring that no information is lost or compromised, even when the input rates surge dramatically. Interactions with the experimental data or external applications can be managed through message brokers, allowing for easy configuration of connections between SciCat and external systems. *Jobs*, for example, leverage this technology, as we will cover later.

MongoDB as the underlying database ensures fast querying together with powerful search capabilities. The *backend* is configured to create indexes [44] for frequently accessed fields, through which MongoDB can lower the query response time. Replication [45] provides high availability, by maintaining two or more copies of the data. Sharding ensures the database scales horizontally, by distributing the data according to ranges in user-defined shard keys [46]. MongoDB can then load balance incoming requests to the correct shard according to the range of the requested key.

## FEATURES AND FUNCTIONALITIES

### Flexible Search and Discovery

SciCat employs powerful search capabilities, allowing users to explore datasets using a range of search parameters, including keywords, authors, data types, time ranges, and locations.

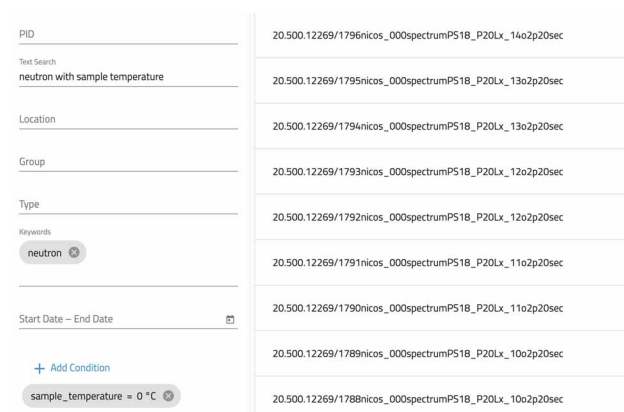
There are two types of searches the user can perform: text search, aka soft filter, and hard filters, namely *key=value* conditions, for example, *Keyword="Neutron"*.

Software

Data Management

Here is an example, as shown in Fig. 2, of a search with both soft and hard filters:

*Find all the datasets that are relevant to the query "neutron with sample temperature", contain the keyword "Neutron" and have a scientific metadata sample temperature equal to 0 Celsius.*



The screenshot shows a search interface with the following filters and results:

Filter	Value
Text Search	neutron with sample temperature
Location	
Group	
Type	
Keywords	neutron
Start Date - End Date	
Hard Filter	sample_temperature = 0 °C

PID	Value
20.500.12269/1796nicos_000spectrumP51B_P20Lx_14o2p20sec	
20.500.12269/1795nicos_000spectrumP51B_P20Lx_13o2p20sec	
20.500.12269/1794nicos_000spectrumP51B_P20Lx_13o2p20sec	
20.500.12269/1793nicos_000spectrumP51B_P20Lx_12o2p20sec	
20.500.12269/1792nicos_000spectrumP51B_P20Lx_12o2p20sec	
20.500.12269/1791nicos_000spectrumP51B_P20Lx_11o2p20sec	
20.500.12269/1790nicos_000spectrumP51B_P20Lx_11o2p20sec	
20.500.12269/1789nicos_000spectrumP51B_P20Lx_10o2p20sec	
20.500.12269/1788nicos_000spectrumP51B_P20Lx_10o2p20sec	

Figure 2: Example of *frontend* search capabilities, showing *soft* and *hard* filters applied on the datasets view.

Search queries are sent to the backend via REST endpoints, each corresponding to a specific entity. There's also a more robust "*fullquery*" route with enhanced metadata querying capabilities, utilizing MongoDB aggregation pipelines [47]. The OpenAPI specification helps in selecting the appropriate endpoint parameters.

For backward compatibility, the query parameters and headers of the endpoints of the SciCat entities follow the syntax defined by the LoopBack 3 [48] framework for querying data [49]. This allows specifying filtering conditions, including related entities, sorting, pagination and field extraction.

### Ingestion

Usually, the process of creating a dataset, aka ingesting, takes place when the data has been acquired and the files are stored in the facility storage, which SciCat can then reference.

Datasets can be ingested by submitting the information as required by the dataset DTO [50] to the *backend* endpoint, or through the *frontend* form. Users can create datasets for their groups, while administrators can create them for any group. This behaviour can be changed to allow all users to create datasets or restrict this functionality to only administrators.

Providing REST APIs for SciCat entities allows facilities to create custom ingestion clients in their preferred language, enhancing adoption due to varied metadata ingestion needs.

### Curation

SciCat is designed to enable users to easily annotate and enrich their data with standardized and domain-specific metadata fields, ensuring consistency and interoperability.

THMBCM002

1197



When selecting a *dataset*, its *scientific metadata* can be edited directly from the *frontend* if the user is one of the owners or a platform admin.

Researchers, data curators and interested stakeholders can update values, add new ones and remove obsolete ones by clicking the “edit” tab, as shown in Fig. 3.

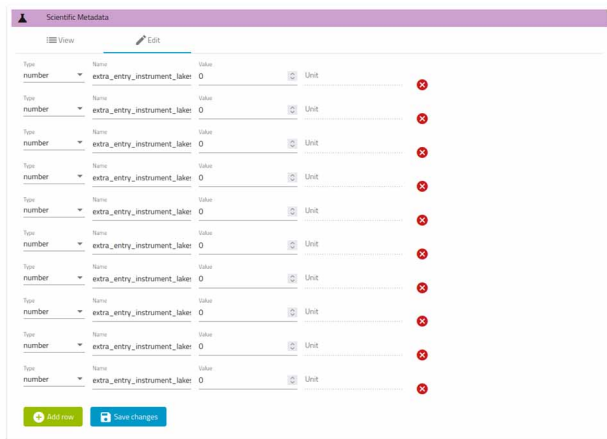


Figure 3: Example of editing *dataset's scientific metadata* from the UI. It can be modified, removed and added.

After the editing, SciCat keeps track of the changes, by storing the versions of the metadata.

The *frontend* allows users to update single datasets easily. However, it is advisable to use the backend for bulk updates, more control, or changes involving multiple datasets or entities.

### Data Sharing and Collaboration

Researchers can share datasets and associated metadata with collaborators or make them publicly available through SciCat. With the advent of FAIR [51], publishing metadata records has become an increasing practice in research institutes, where data policies impose the publishing after a fixed period of time. Many funding agencies impose publishing as part of the eligibility requirements.

Each dataset has the authorisation properties: owner group and access groups. The owner group is the group that owns the dataset and can update it. The access groups are all the groups that have access to this dataset, can view it and download its data files.

A vanilla deployment allows two levels of dataset access: ownership and access. If a more fine-grained custom authorization model is needed, it can be achieved by editing directly the CASL configuration.

To make a dataset public, the user can set "*isPublished*" to true in the database record. This can be achieved using the "*Public*" toggle in the *frontend*.

Datasets can be grouped as "Published Data" records. The user can simply select the datasets, and click "Publish", which opens a form. On completion, it registers them with a DOI authority, becoming publicly accessible. This can also be done through backend endpoints like other cases.

### Workflows

To enable operations on experimental data, *jobs* were developed, the user can select a number of datasets and select a job that should be executed.

Two common use cases are persisting data in a tape drive storage system [52], and triggering the data retrieval to the user's laptop or a dedicated server for data analysis. The *frontend* can be configured to support these actions, which the user can trigger after having selected the datasets. Figure 4 shows an example of retrieval action.

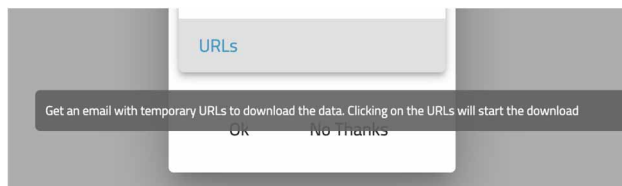


Figure 4: Example of a retrieval action from the UI. The data is moved from the tape drive to an object store URL.

This sends a request to the *backend's jobs* endpoint with the essential information in the payload, including the action and specific arguments like where to retrieve the data. The source location was previously saved in SciCat as part of the cataloguing process. Finally, it posts a message to a message broker. The archiver/retriever clients subscribe to the message broker, trigger the data movement and notify SciCat of updates, by doing a PATCH to the *jobs/{id}* endpoint. Notable examples of such clients are based on AREMA [53] or Globus [54]. As this is an asynchronous process, SciCat notifies users upon job completion. Email notifications can be configured as part of the backend installation.

Additional actions can be initiated by directly engaging with the *backend's jobs* endpoint. Facilities can configure the triggered process by subscribing an appropriate client to the message broker. This decoupling enhances customization, and integration into the IT landscape, and supports asynchronous processes, reducing user interaction.

### Integration with External Data Repositories

SciCat easily integrates with existing data federation platforms which provides a unified interface to search and access data across multiple platforms, including B2Find, EOSC, openAIRE and Google Dataset Search.

Under the SciCat project on Github, multiple data integration services can be found and installed to expose the public data to the platform of choice.

New integration services can be developed and deployed leveraging the extensive REST API and the accessibility of the code.

During the PaNOSC & ExPaNDS projects, a service was created to align SciCat and facility-specific metadata with the project's naming schema. They also developed a web portal and a distributed scoring system, enabling search capabilities across multiple independent facilities with distinct information sets.

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

## Python Interfaces

In order to lower the barrier to data usage, the SciCat collaborators' group have developed two different libraries to interface SciCat with Python, Pysicat [55] and Scitacean [56].

Pysicat is a Python library that aims to expose all the SciCat backend endpoints, mapping each API endpoint to Python functions.

Scitacean is a user-friendly high-level library. It is designed to mask all the underlying SciCat API details from the users and streamline the process to retrieve datasets (including metadata and data files) and also create new datasets (including data file upload) for the users.

## FUTURE DEVELOPMENTS

We envision expanding the REST API and providing ready-to-use aggregation specifically tailored for AI and ML-established techniques. We would aim to implement automated keyword assignment and metadata quality evaluation.

We also aim to improve the search capabilities by applying the latest Information Retrieval techniques and relevance scoring. To address further customization, the core collaborators feel the need to design and develop a plugin framework, so site administrators can quickly develop custom functionalities to meet their facility's unique needs.

SciCat has proven its maturity with multiple instances deployed in production in different facilities and labs. The recent successful migration to the new implementation of the backend V4.x has been beneficial in streamlining CI/CD, improving automated testing, and increasing the case tested considerably. It also helped us to discover areas where both frontend and backend need to be improved and allowed us to better plan the work ahead of us.

## CONCLUSIONS

At each facility, the deployment of SciCat with its versatility allowed it to quickly adjust each instance to the ever-changing needs of the community served. The deployments range widely in size, number of users, and scientific domain proving that it is a scalable and flexible tool.

The development of the two Python libraries and the REST API that the backend implements have proven a successful choice as they have fostered the development of a number of third-party tools and enabled each facility to write custom ingestors for the data produced on-premises and in remote locations.

The flexibility, the worldwide adoption and the established community, all are signals of a bright future ahead of SciCat.

The upcoming features and the new members will allow the product to strive even further, always answering the community's needs, always at the service of scientific progress, and always devoting itself to improvements.

## ACKNOWLEDGEMENTS

We would like to warmly thank all current and past contributors, without whom the project would have progressed significantly slower.

In particular, we convey our heartfelt gratitude to Dr. Stephan Egli and wish him a happy retirement.

## REFERENCES

- [1] A Guardian Guide to your Metadata, [theguardian.com](https://theguardian.com)
- [2] SciCat, <https://scicatproject.github.io>
- [3] ESS, <https://europeanspallationsource.se>
- [4] PSI, <https://www.psi.ch/en>
- [5] MAXIV, <https://www.maxiv.lu.se>
- [6] Rosalind Franklin Institute, <https://www.rfi.ac.uk>
- [7] ALS, <https://als.lbl.gov>
- [8] DAPHNE, <https://www.daphne4nfdi.de/english>
- [9] BAM, <https://www.bam.de/Navigation/DE/Home/home.html>
- [10] SSRF, <https://lssf.cas.cn/en/facilities-view.jsp?id=ff8080814ff56599014ff599b8550033>
- [11] SOLEIL, <https://www.synchrotron-soleil.fr/en>
- [12] DESY, <https://www.desy.de>
- [13] B2FIND, <https://b2find.eudat.eu>
- [14] openAIRE, <https://explore.openaire.eu>
- [15] EOSC Portal, <https://eosc-portal.eu>
- [16] Google Dataset Search, <https://datasetsearch.research.google.com>
- [17] PaNOSC, data portal, <https://data.panosc.eu>
- [18] PaNOSC, European Union's Horizon 2020 research and innovation programme under grant agreement No. 823852, <https://www.panosc.eu>
- [19] ExPaNDS, European Union's Horizon 2020 research and innovation programme under grant agreement No 857641, <https://expands.eu>
- [20] Python, <https://www.python.org>
- [21] Transmission Control Protocol, in Wikipedia, [https://en.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](https://en.wikipedia.org/wiki/Transmission_Control_Protocol)
- [22] HTTP, in Wikipedia, <https://en.wikipedia.org/wiki/HTTP>
- [23] Advanced Message Queuing Protocol, in Wikipedia, [https://en.wikipedia.org/wiki/Advanced\\_Message\\_Queueing\\_Protocol](https://en.wikipedia.org/wiki/Advanced_Message_Queueing_Protocol)
- [24] MQTT, in Wikipedia, <https://en.wikipedia.org/wiki/MQTT>
- [25] WebSocket, in Wikipedia, <https://en.wikipedia.org/wiki/WebSocket>
- [26] MongoDB, <https://www.mongodb.com>
- [27] SciCat, Data Model, [https://scicatproject.github.io/documentation/Development/v4.x/Data\\_Model.html](https://scicatproject.github.io/documentation/Development/v4.x/Data_Model.html)
- [28] MongoDB collections, <https://www.mongodb.com/docs/manual/core/databases-and-collections>

Content from this work may be used under the terms of the CC BY 4.0 licence (© 2023). Any distribution of this work must maintain attribution to the author(s), title of the work, publisher, and DOI

- [29] SciCat Github Organisation, <https://github.com/SciCatProject>
- [30] REST, in Wikipedia, <https://en.wikipedia.org/wiki/REST>
- [31] Typescript, <https://www.typescriptlang.org>
- [32] NestJS, <https://nestjs.com>
- [33] Mongoose, <https://mongoosejs.com>
- [34] CASL, <https://casl.js.org/v6/en>
- [35] OpenID Connect, in Wikipedia, [https://de.wikipedia.org/wiki/OpenID\\_Connect](https://de.wikipedia.org/wiki/OpenID_Connect)
- [36] Create, read, update and delete, in Wikipedia, [https://en.wikipedia.org/wiki/Create,\\_read,\\_update\\_and\\_delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete)
- [37] Swagger UI, <https://swagger.io/tools/swagger-ui>
- [38] OpenAPI initiative, <https://www.openapis.org>
- [39] Docker, <https://www.docker.com>
- [40] Kubernetes, <https://kubernetes.io>
- [41] Identity management. (2023, August 30). In Wikipedia, [https://en.wikipedia.org/wiki/Identity\\_management](https://en.wikipedia.org/wiki/Identity_management)
- [42] Single-page application. (2023, September 27). In Wikipedia, [https://en.wikipedia.org/wiki/Single-page\\_application](https://en.wikipedia.org/wiki/Single-page_application)
- [43] Angular, <https://angular.io>
- [44] MongoDB, Indexes, <https://www.mongodb.com/basics#indexes>
- [45] MongoDB, Replication, <https://www.mongodb.com/basics#replica-sets>
- [46] MongoDB, Sharding, <https://www.mongodb.com/basics/sharding>
- [47] MongoDB, Aggregation Pipelines, <https://www.mongodb.com/basics#aggregation-pipelines>
- [48] LoopBack 3, <https://loopback.io/lb3>
- [49] LoopBack 3, Query Data, <https://loopback.io/doc/en/lb3/Querying-data.html>
- [50] Data transfer object. (2023, July 20). In Wikipedia, [https://en.wikipedia.org/wiki/Data\\_transfer\\_object](https://en.wikipedia.org/wiki/Data_transfer_object)
- [51] FAIR, <https://www.go-fair.org/fair-principles>
- [52] Tape drive, In Wikipedia, [https://en.wikipedia.org/wiki/Tape\\_drive](https://en.wikipedia.org/wiki/Tape_drive)
- [53] AREMA, <https://www.ibm.com/products/arema-archive-and-essence-manager/details>
- [54] Globus, <https://www.globus.org>
- [55] Pyscicat, <https://scicatproject.github.io/pyscicat>
- [56] Scitacean, <https://scicatproject.github.io/scitacean>