

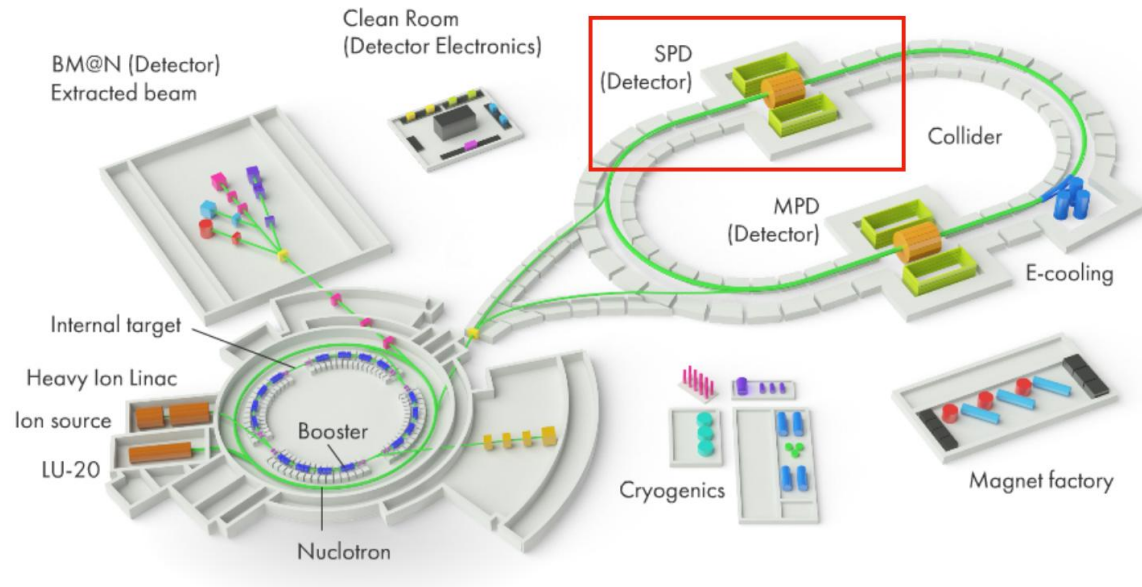
Data Management System for SPD Online Filter

Korshunova Polina
JINR MLIT

2025 AYSS

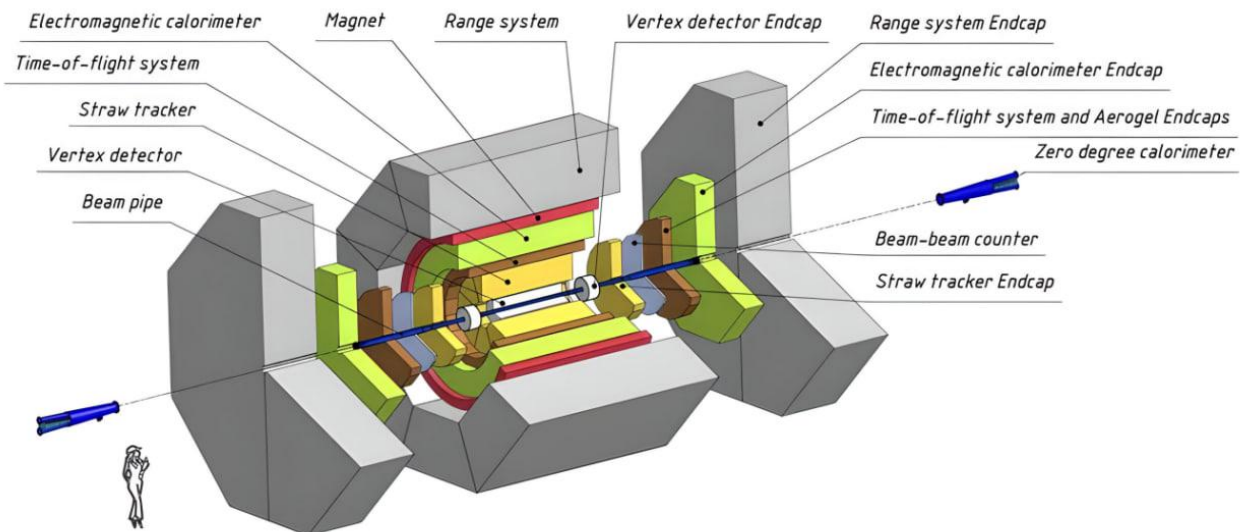


Spin Physics Detector

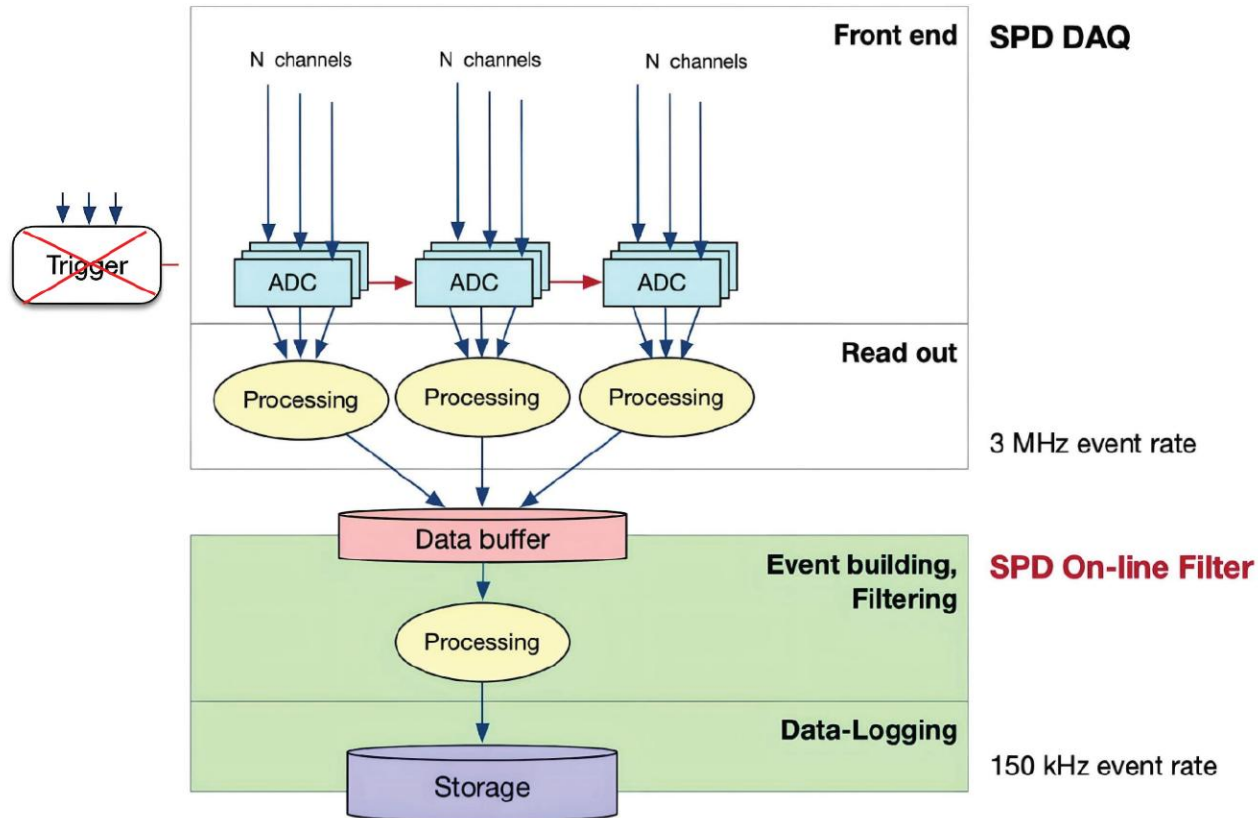


The SPD detector will be used to study the spin structure of the proton and deuteron and other spin-related phenomena

The main purpose of the SPD experiment is a comprehensive study of the unpolarized and polarized gluon component of the nucleon



Data collection from the detector

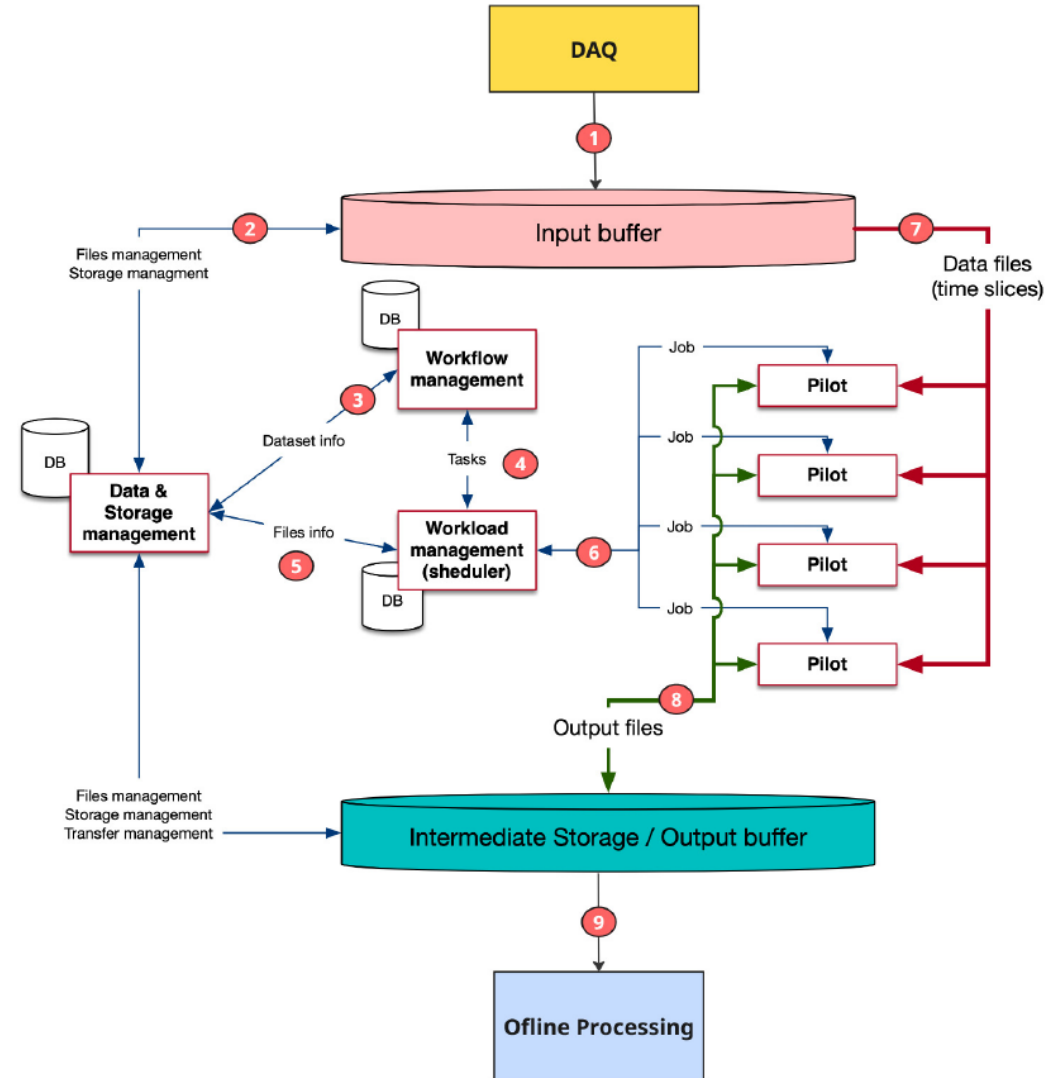


- A **simple selection** of physical events at the hardware level is **not possible**
- The need to **collect the entire set of generated signals** from subsystems combined in time blocks
- Large data flow up to **20 GB/s** (~200 PB/year)
- The need to **reduce the amount of data** for subsequent analysis
- Software Trigger Development – **SPD Online filter**

Online Filter

SPD Online Filter is a high-performance computing system for high-throughput data processing

A special feature of **high-throughput data processing** is the large amount of data, both primary that needs to be processed and intermediate that occurs during processing



Middleware software

Data management system

- data lifecycle support (data catalog, consistency check, cleanup, storage)

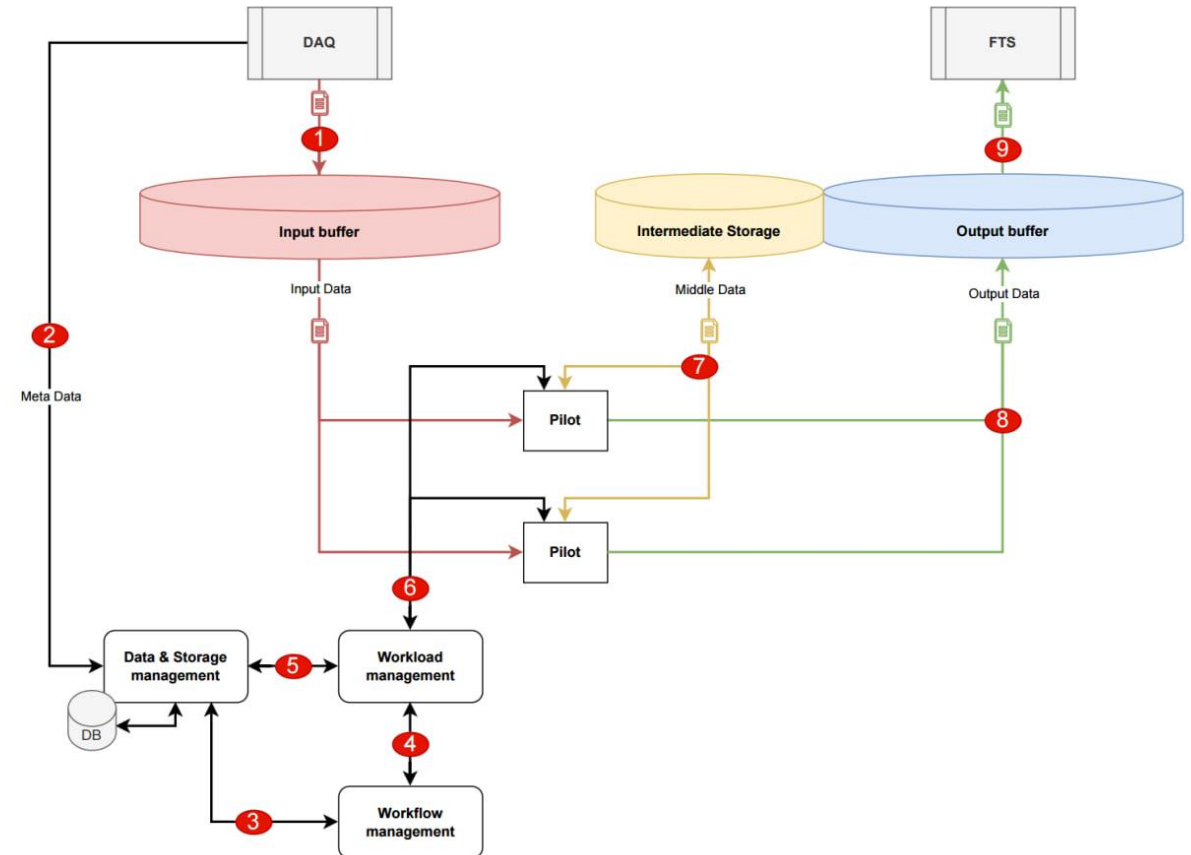
Workflow Management System

- define and execute processing chains by generating the required number of computational tasks

Workload management system

- implementation of processing stages (task generation, sending tasks to pilots)

Pilot – an application running on a computing node and performing tasks



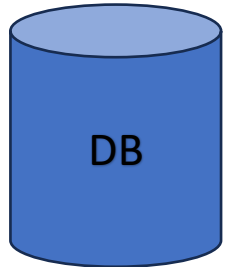
Data management system: tasks

- ✓ Registration of new data
- ✓ Cataloging
- ✓ Ensuring control over data storage and integrity

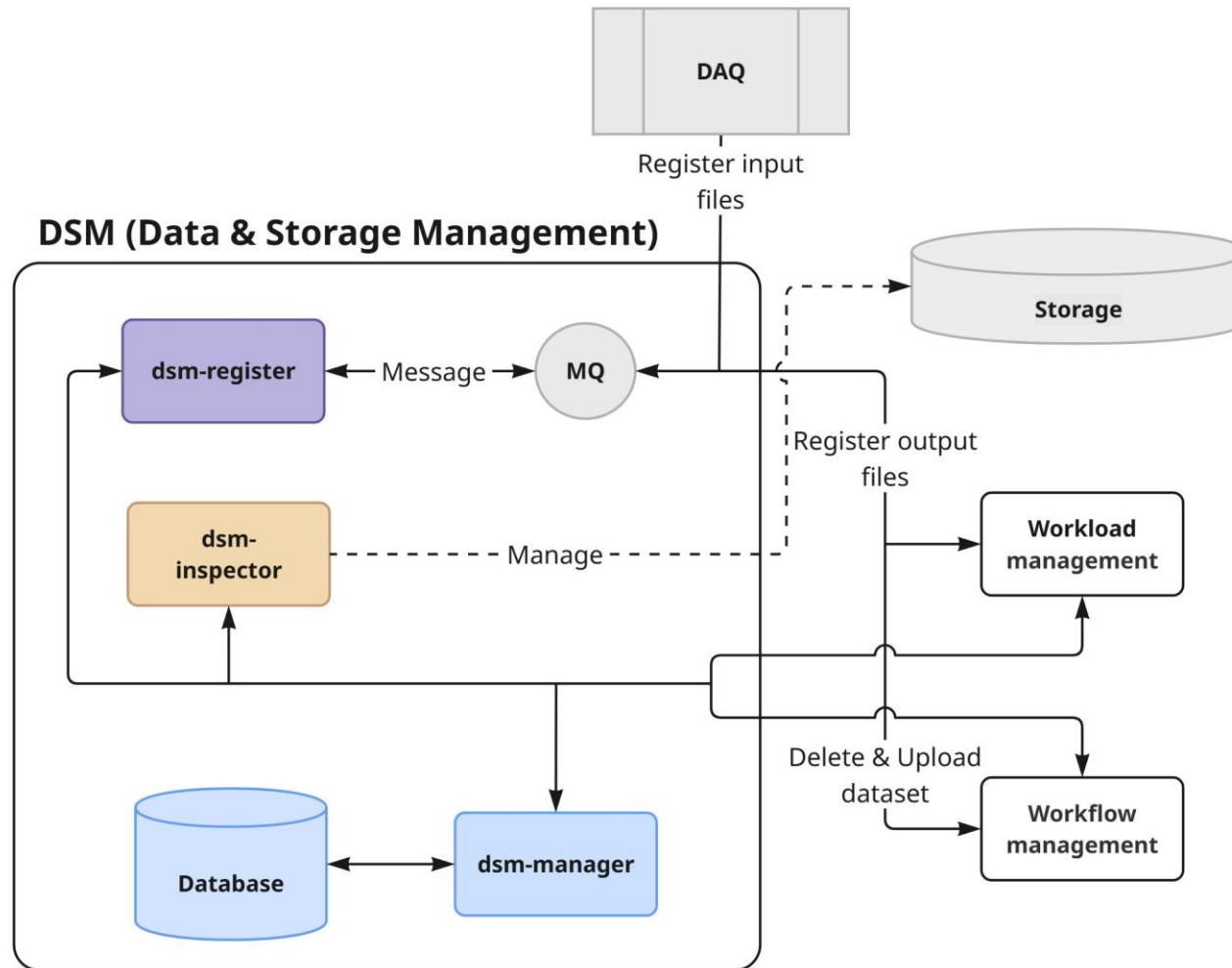
Interface for accepting requests for adding/deleting data in the system

Interface to the data catalog

A set of background tasks to ensure consistency between storage and database



Data management system



dsm-register (data registration)

- a service that receive requests for adding/deleting data in the system asynchronously (via MQ)

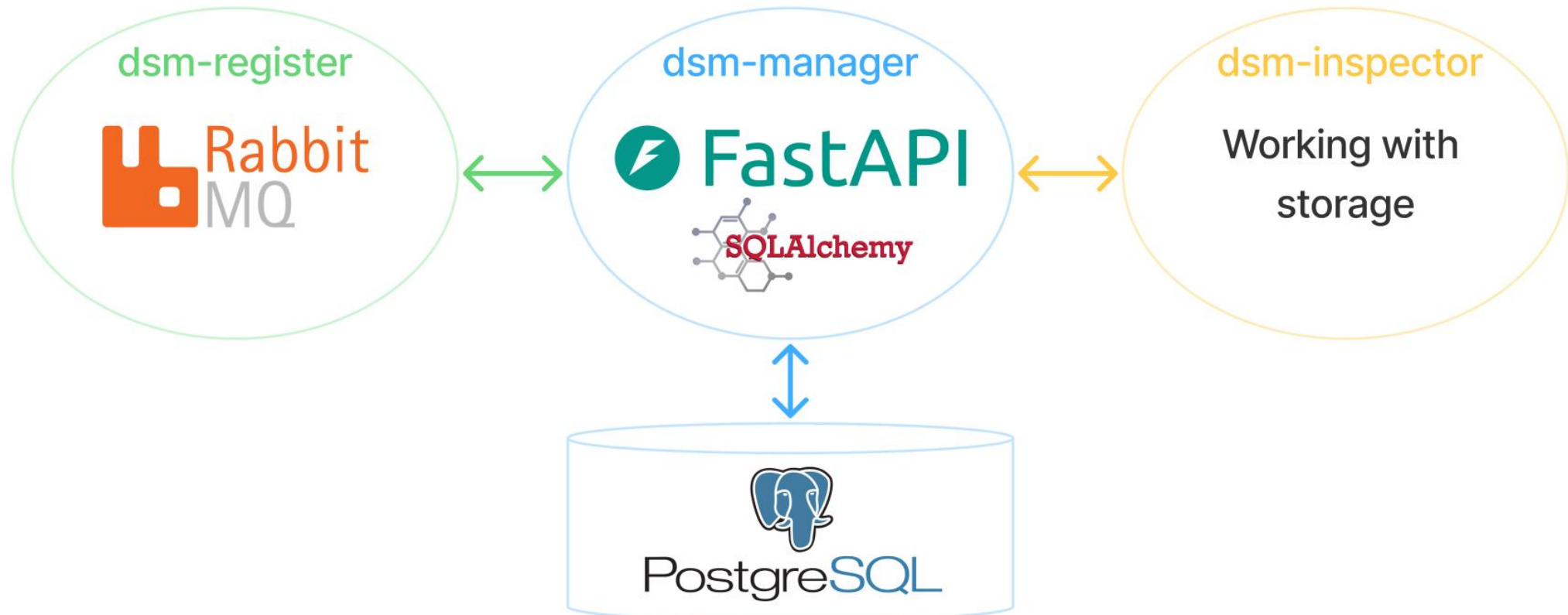
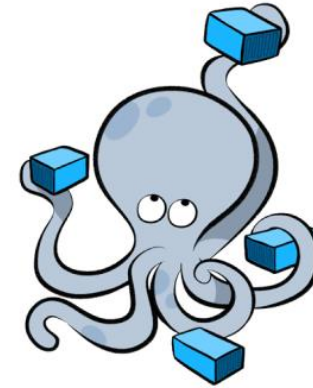
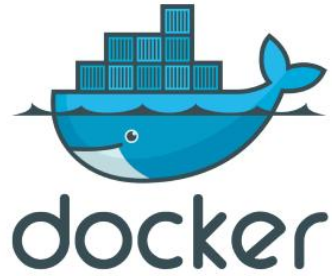
dsm-manager (REST API of data catalog)

- file and dataset management (adding data to a database, changing data, deleting data)

dsm-inspector (daemon tasks)

- delete files on storage, check consistency of files, monitoring the use of storage (for example, "dark" data)

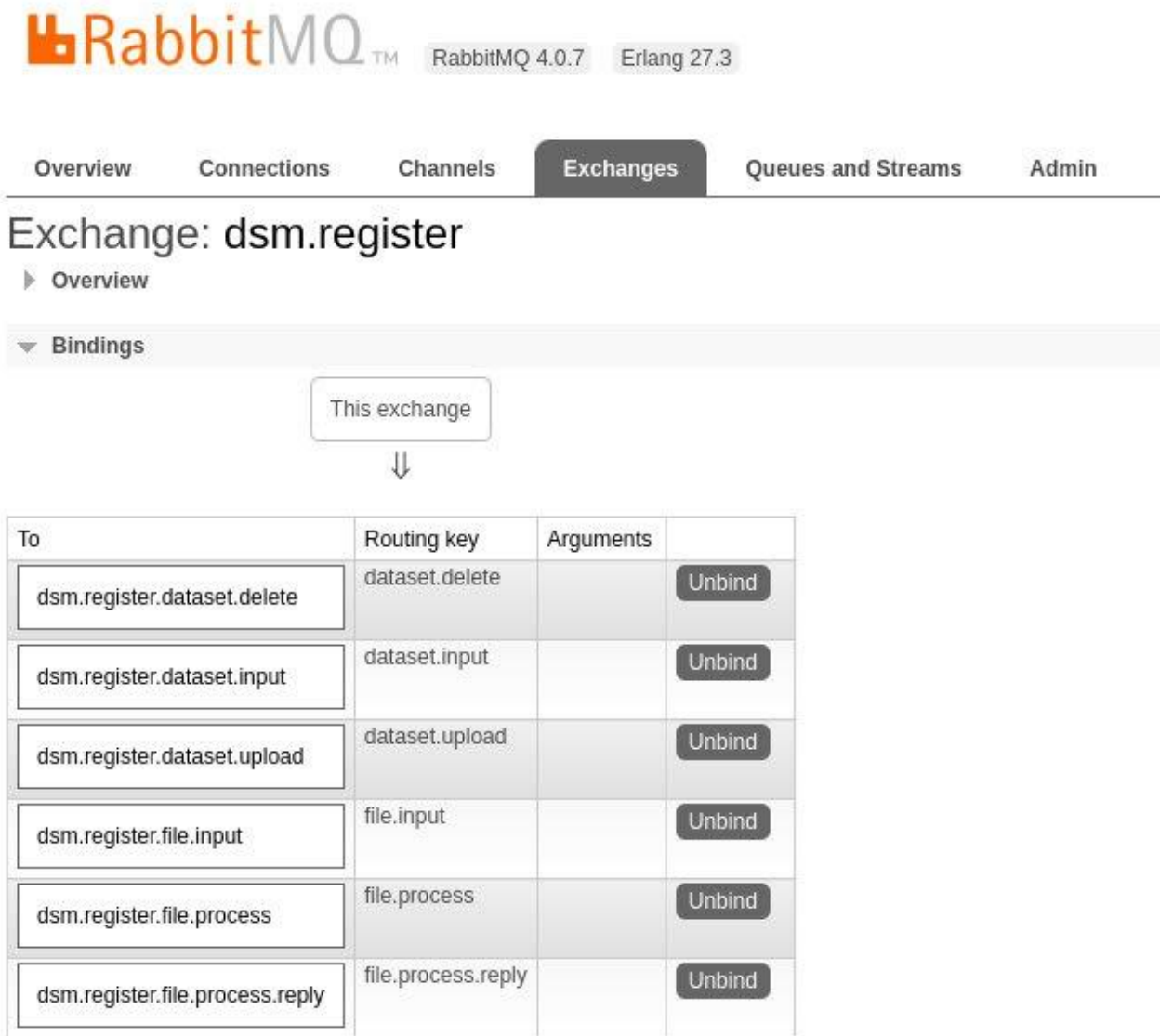
Technology stack



dsm-register: configured queues

The service should check the message queue and process requests for adding/deleting data in the system

[RabbitMQ](#) is used as an AMQP broker that performs routing and subscribing to the necessary queues



The screenshot shows the RabbitMQ web interface. At the top, the RabbitMQ logo is displayed along with the version information: RabbitMQ 4.0.7 and Erlang 27.3. The navigation bar includes links for Overview, Connections, Channels, Exchanges (which is currently selected), Queues and Streams, and Admin.

The main section is titled "Exchange: dsm.register" and has a sub-link for "Overview". Below this, the "Bindings" section is expanded, showing a diagram where "This exchange" points down to a table of bindings.

To	Routing key	Arguments	
dsm.register.dataset.delete	dataset.delete		Unbind
dsm.register.dataset.input	dataset.input		Unbind
dsm.register.dataset.upload	dataset.upload		Unbind
dsm.register.file.input	file.input		Unbind
dsm.register.file.process	file.process		Unbind
dsm.register.file.process.reply	file.process.reply		Unbind

dsm-manager: API to the DB

file		^
GET	/api/v1/file/ Get List	✓
POST	/api/v1/file/ Add	✓
GET	/api/v1/file/{file_id} Get By Id	✓
PUT	/api/v1/file/{file_id} Update	✓
DELETE	/api/v1/file/{file_id} Remove	✓
GET	/api/v1/file/file_name/{file_name} Get By Name	✓
dataset		^
GET	/api/v1/dataset/ Get List	✓
POST	/api/v1/dataset/ Add	✓
GET	/api/v1/dataset/{dataset_id} Get By Id	✓
PUT	/api/v1/dataset/{dataset_id} Update	✓
DELETE	/api/v1/dataset/{dataset_id} Remove	✓
PATCH	/api/v1/dataset/{dataset_id} Update Status	✓
GET	/api/v1/dataset/name/{dataset_name} Get By Name	✓

- ✓ Getting a list of files in a dataset
- ✓ Getting a list of files with a specific status



The service must provide a REST API to the database

The asynchronous **FastAPI** framework is used as a web framework

- ✓ Getting a list of datasets that contain a specific file
- ✓ Getting a list of datasets with a specific status

dsm-inspector

The service consists of a set of **background** tasks:

- Deleting files on storages
 - Deleting datasets and files
 - Deleting dark files
- File integrity check
- Storage monitoring
 - Dark files monitoring
 - Monitoring storage usage
- File upload control
 - Setting tasks for uploading datasets
 - Uploading monitoring

SPD DAQ Emulator

⚙️ config.ini

[General]

number_of_generators = 5
number_of_runs = 10
sleep = 7200

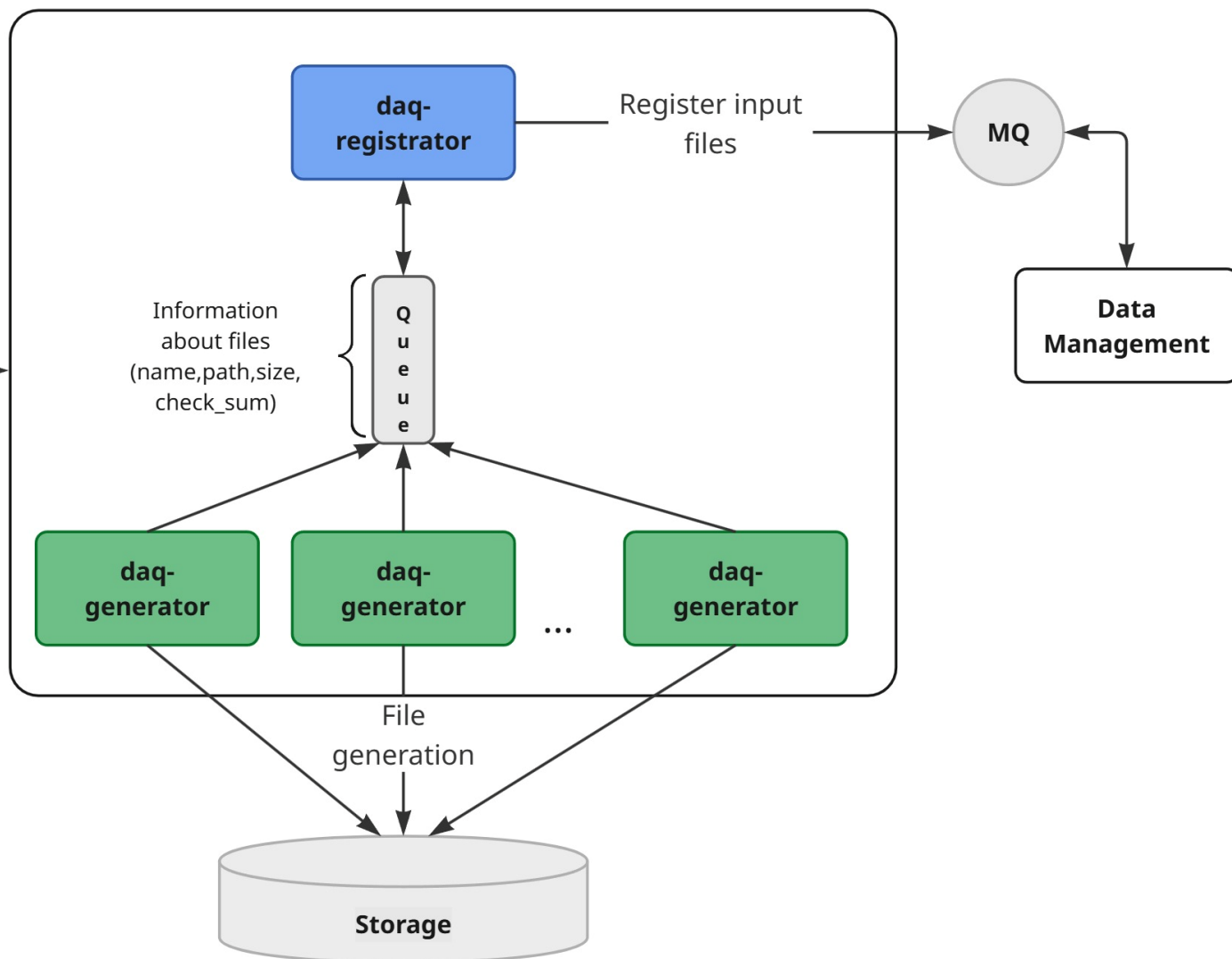
[Files]

file_size = 50
number_of_files = 100

[Datasets]

name_template = "_run_"
dataset_size = 10

DAQ EMULATOR



Testing

Queue dsm.register.file.input

▼ Overview

Queued messages last minute ?



Ready 0
Unacked 0
Total 0

Message rates last minute ?



Publish 0.00/s
Deliver (manual ack) 0.00/s
Deliver (auto ack) 0.00/s

Consumer ack 0.00/s
Redelivered 0.00/s
Get (manual ack) 0.00/s

Get (auto ack) 0.00/s
Get (empty) 0.00/s

- Produce 10 datasets of 10 files (5 MB each file)
- Send messages to dsm-register
- dsm-register creates a new dataset through dsm-manager and registers files
- Registration time for one dataset is ~ 0.5 seconds

```
app-1 | 2025-10-21 14:03:38 INFO: Registration time = 0.51 sec
app-1 | 2025-10-21 14:03:44 INFO: Registration time = 0.46 sec
app-1 | 2025-10-21 14:03:50 INFO: Registration time = 0.46 sec
app-1 | 2025-10-21 14:03:56 INFO: Registration time = 0.47 sec
app-1 | 2025-10-21 14:04:03 INFO: Registration time = 0.5 sec [
app-1 | 2025-10-21 14:04:11 INFO: Registration time = 0.47 sec
app-1 | 2025-10-21 14:04:17 INFO: Registration time = 0.45 sec
app-1 | 2025-10-21 14:04:22 INFO: Registration time = 0.48 sec
app-1 | 2025-10-21 14:04:28 INFO: Registration time = 0.53 sec
app-1 | 2025-10-21 14:04:33 INFO: Registration time = 0.51 sec
```

Conclusion

Current results:

- ✓ **dsm-manager** fully functional for this stage of implementation
- ✓ **dsm-register** implemented for this stage
- ✓ **dsm-inspector** is implemented by 75%
- ✓ Requirements for **SPD DAQ Emulator** are developed

Further plans:

- Implement file upload control for dsm-inspector
- Implement DAQ Emulator for continuous file generation

