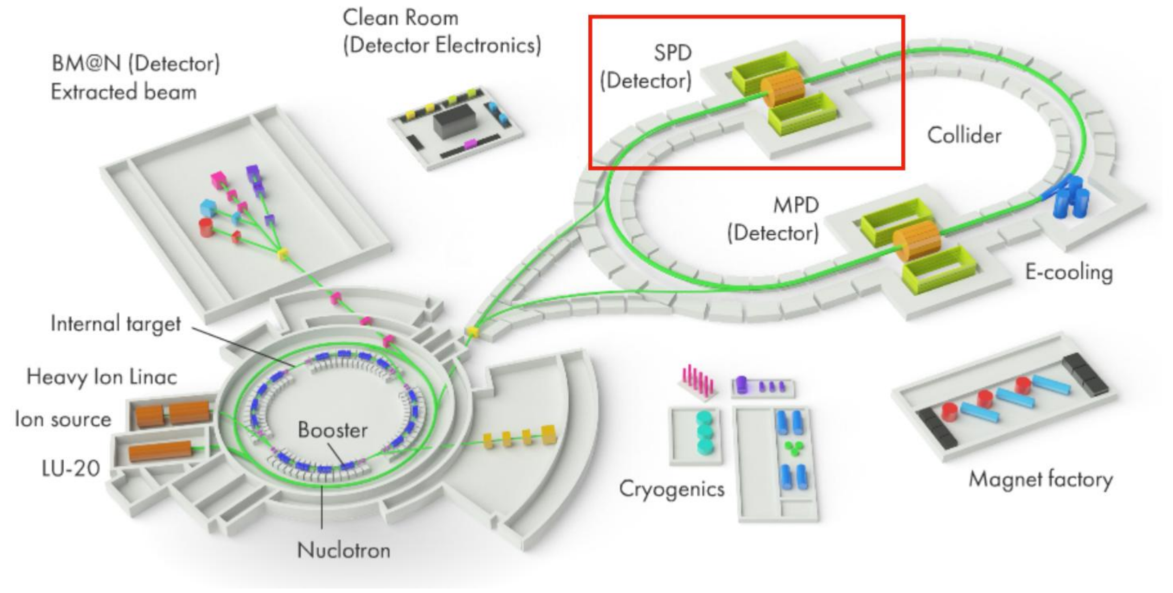
Data Management System for SPD Online Filter

Korshunova Polina

JINR MLIT

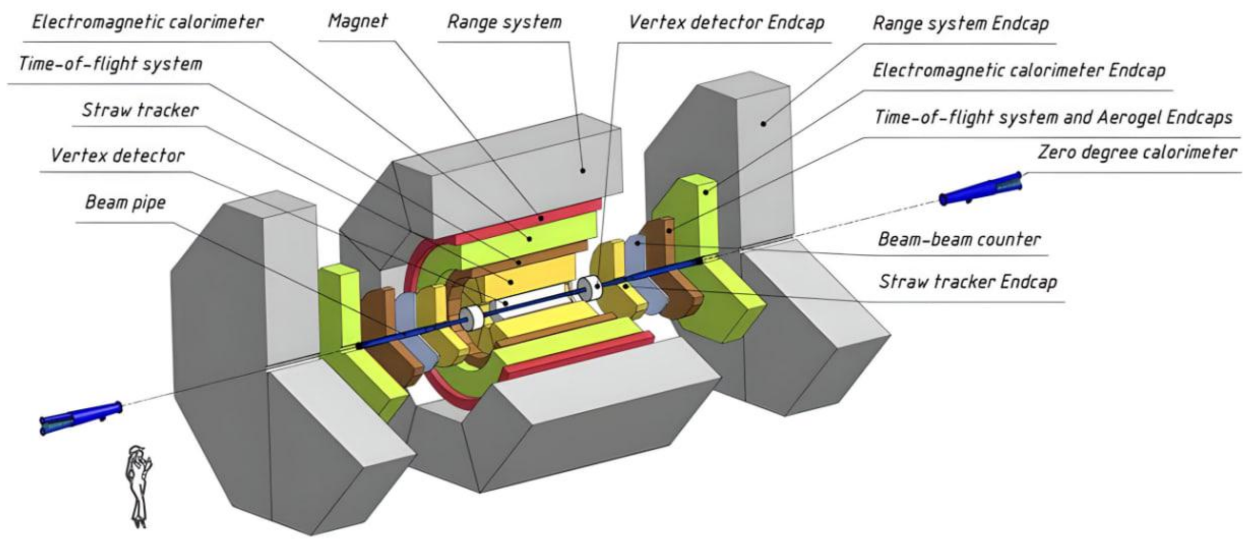
2025 GRID

Spin Physics Detector

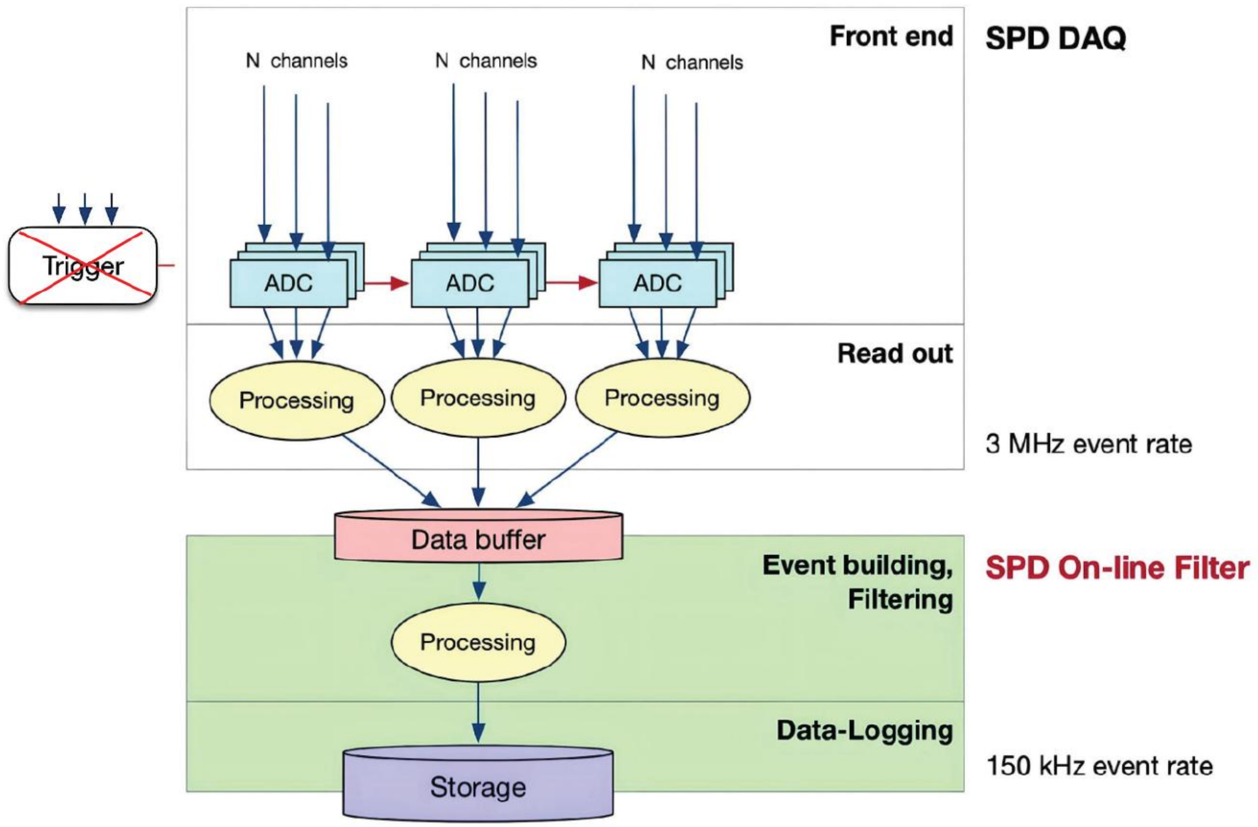


The SPD detector will be used to study the spin structure of the proton and deuteron and other spin-related phenomena

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

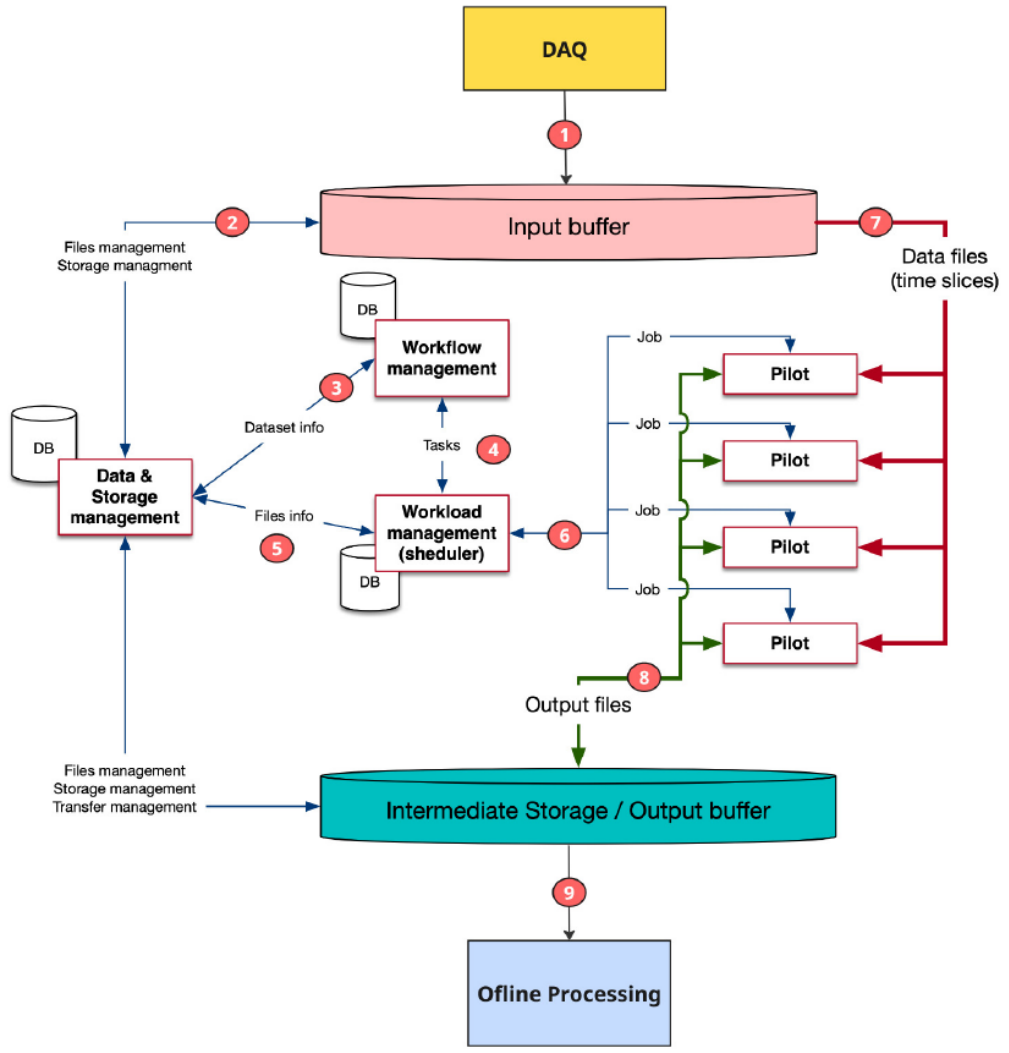


- A simple selection of physical events at the hardware level is not possible
- The need to collect the entire set of generated signals from subsystems combined in time blocks
- Large data flow up to 20 GB/s (~200 PB/year)
- The need to reduce the amount of data for subsequent analysis
- Software Trigger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

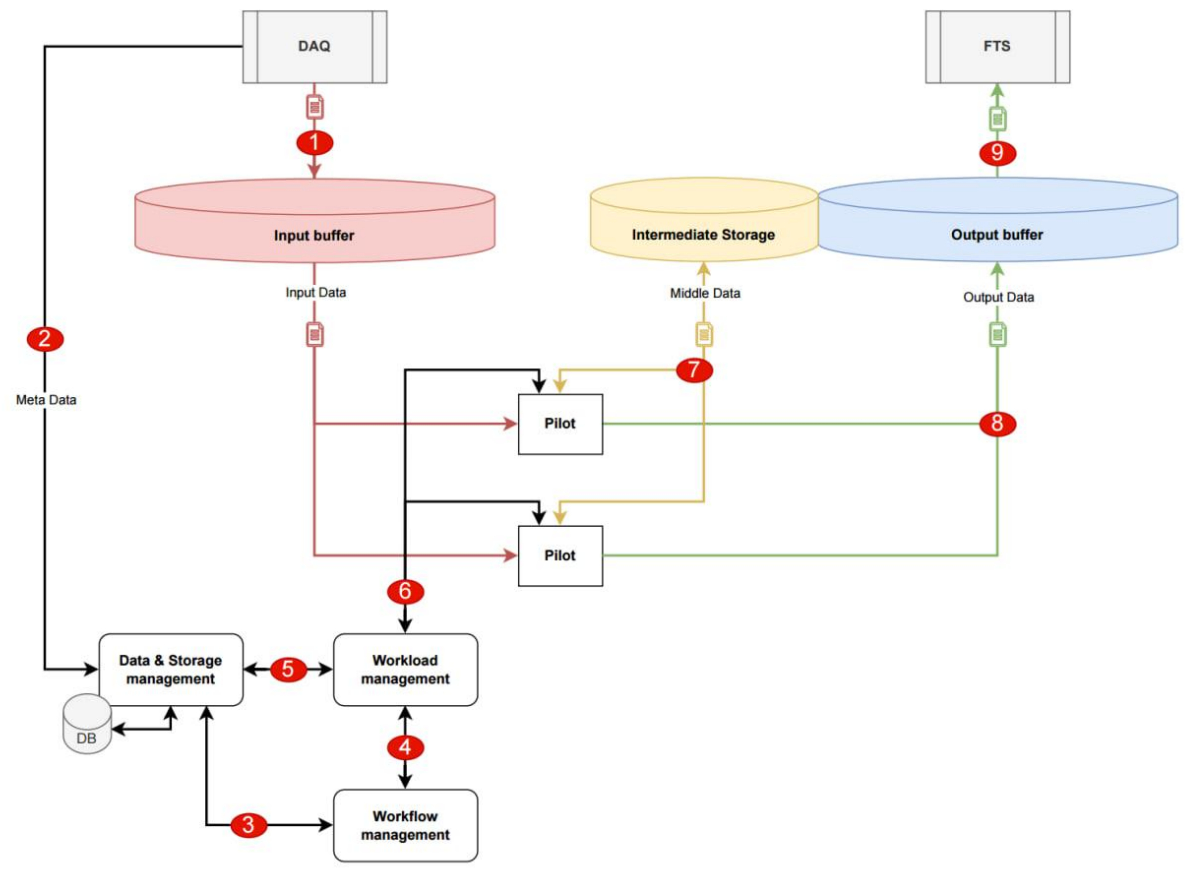
Workflow Management System

- 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

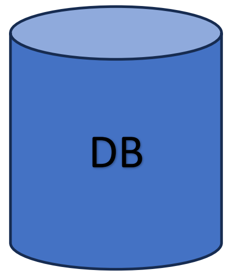


Data management system: tasks

- ✓ Registration of new data
- ✓ Cataloging
- ✓ File integrity and upload control, file deletion on storages, storage monitoring

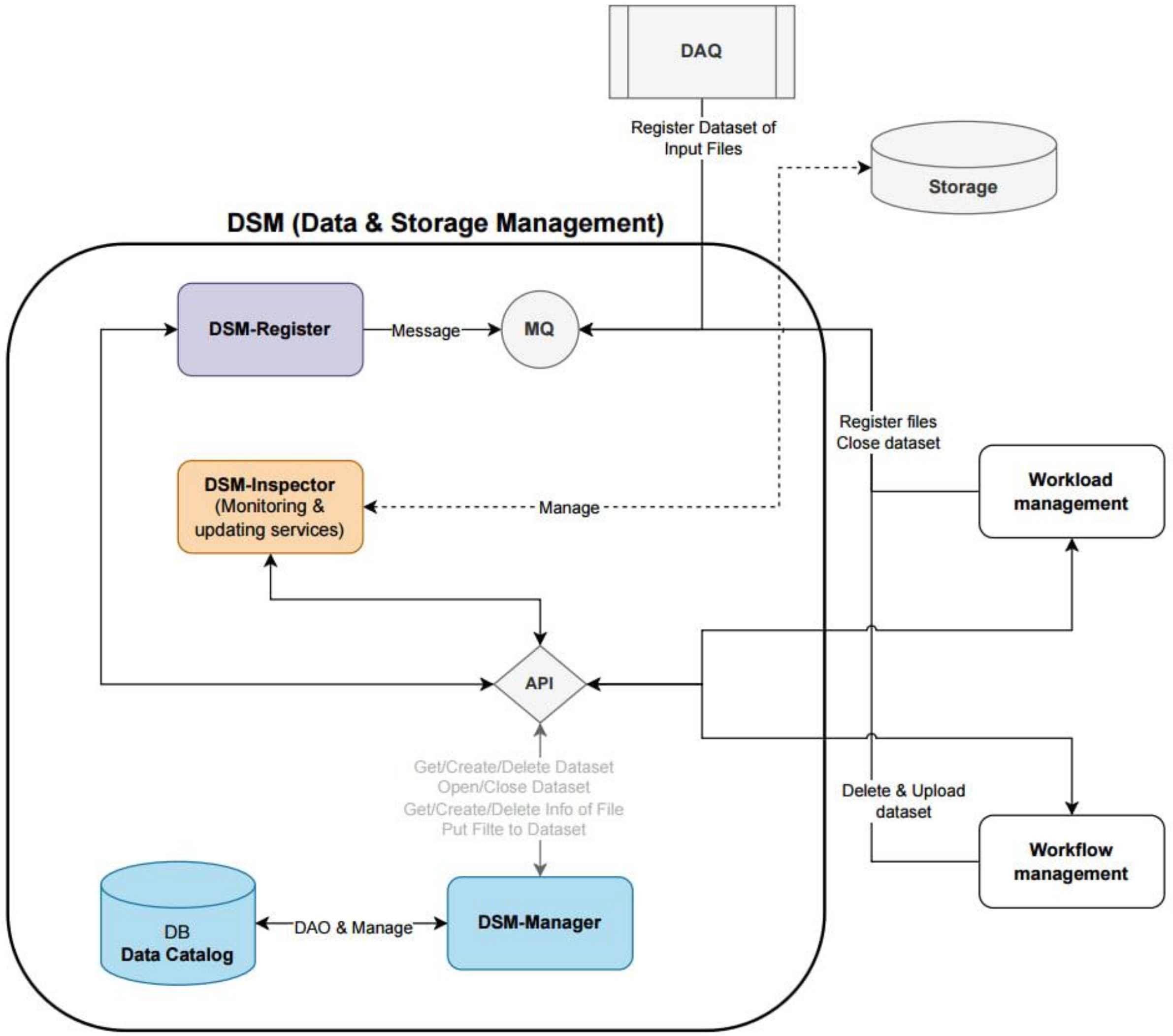
Interface for accepting requests for adding/deleting data in the system

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)

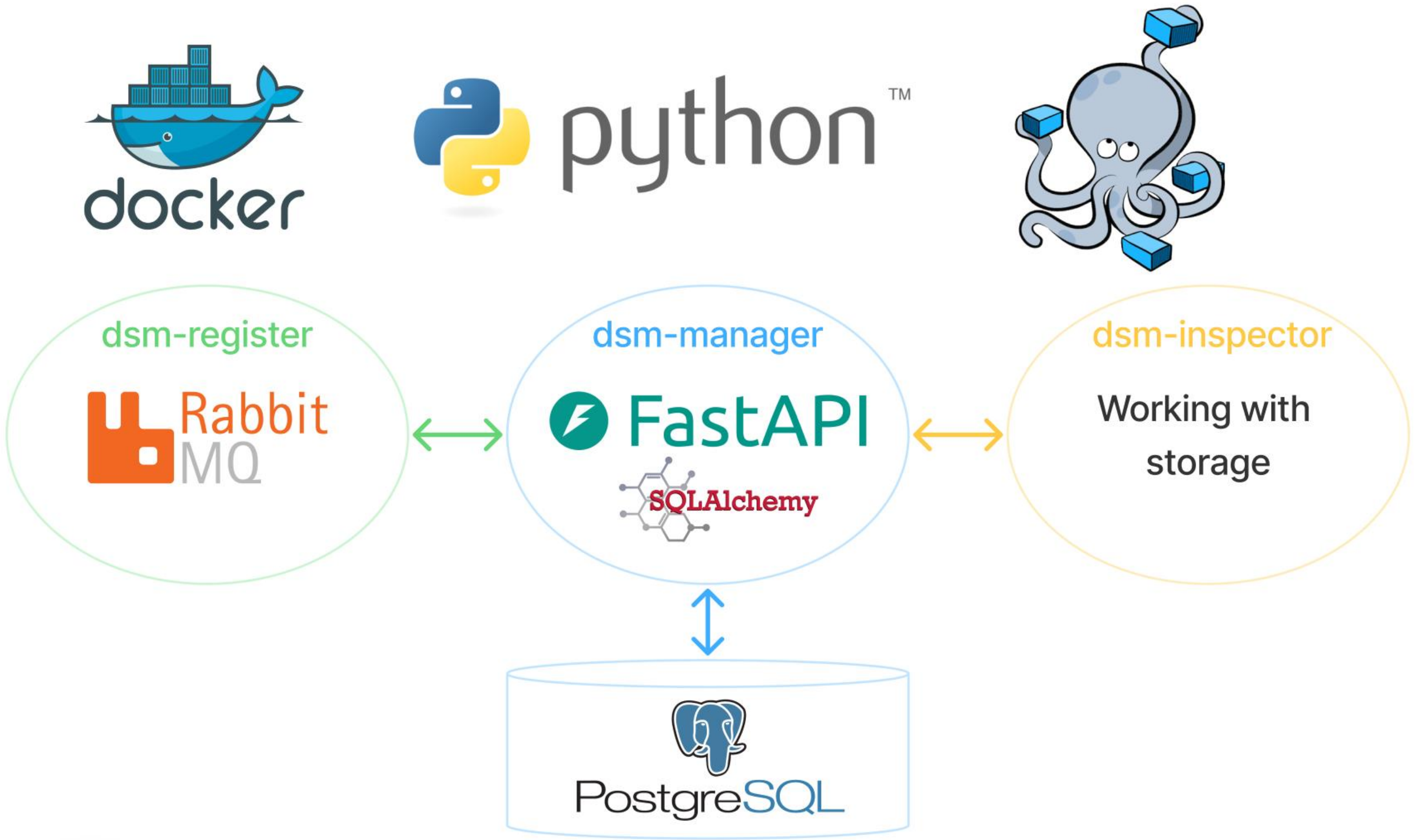
dsm-manager (REST API of data catalog)

- 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)

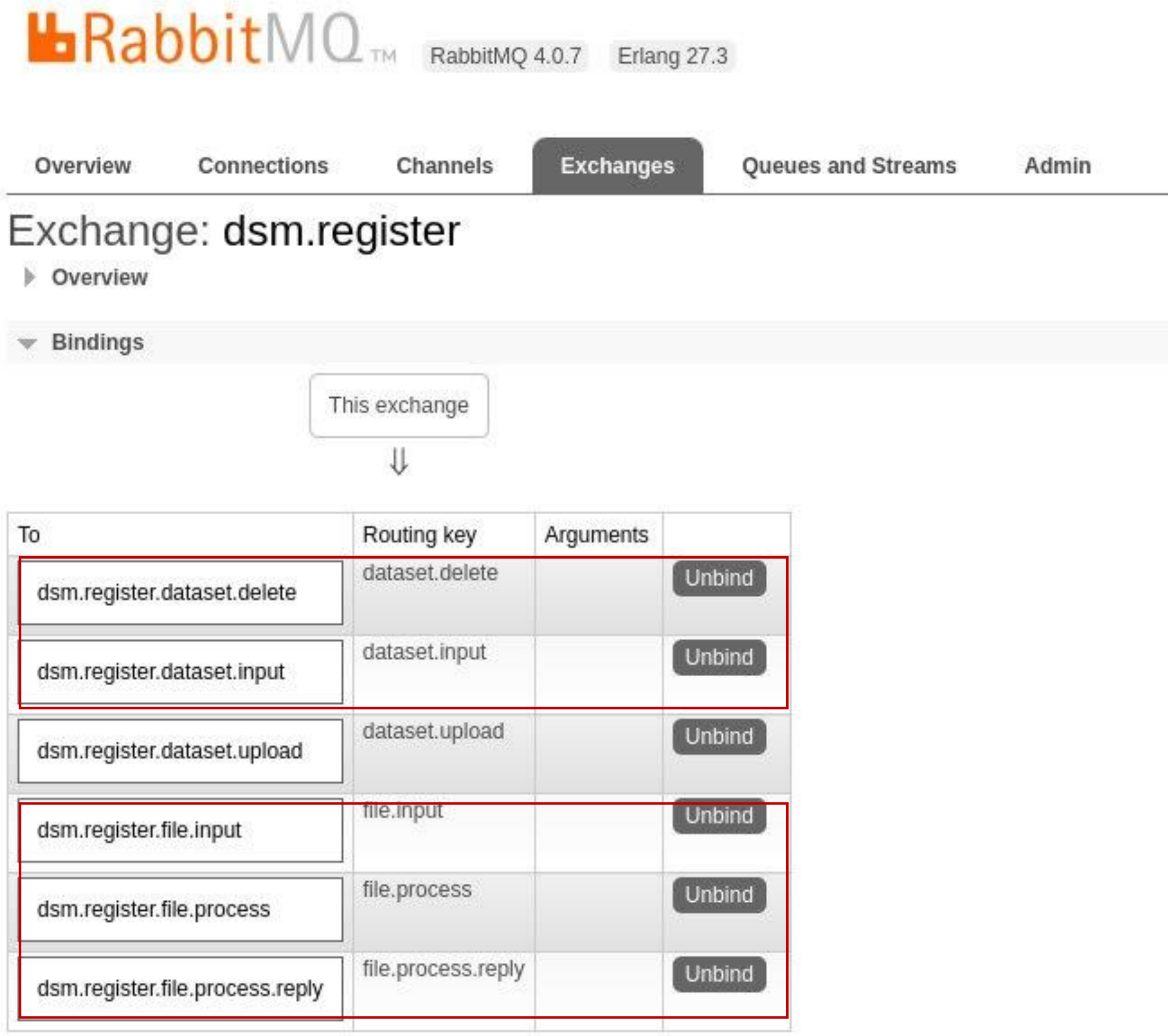
Technology stack



dsm-register: configured queues

The service should listen to 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



Exchange: dsm.register

Overview

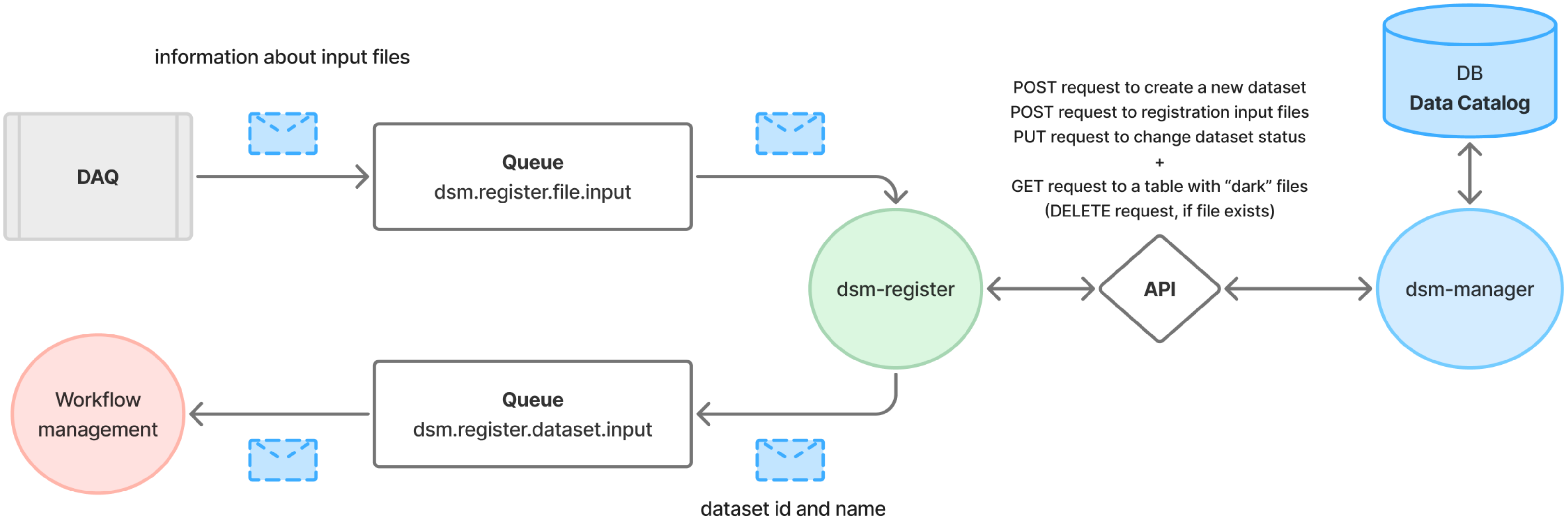
Bindings

This exchange

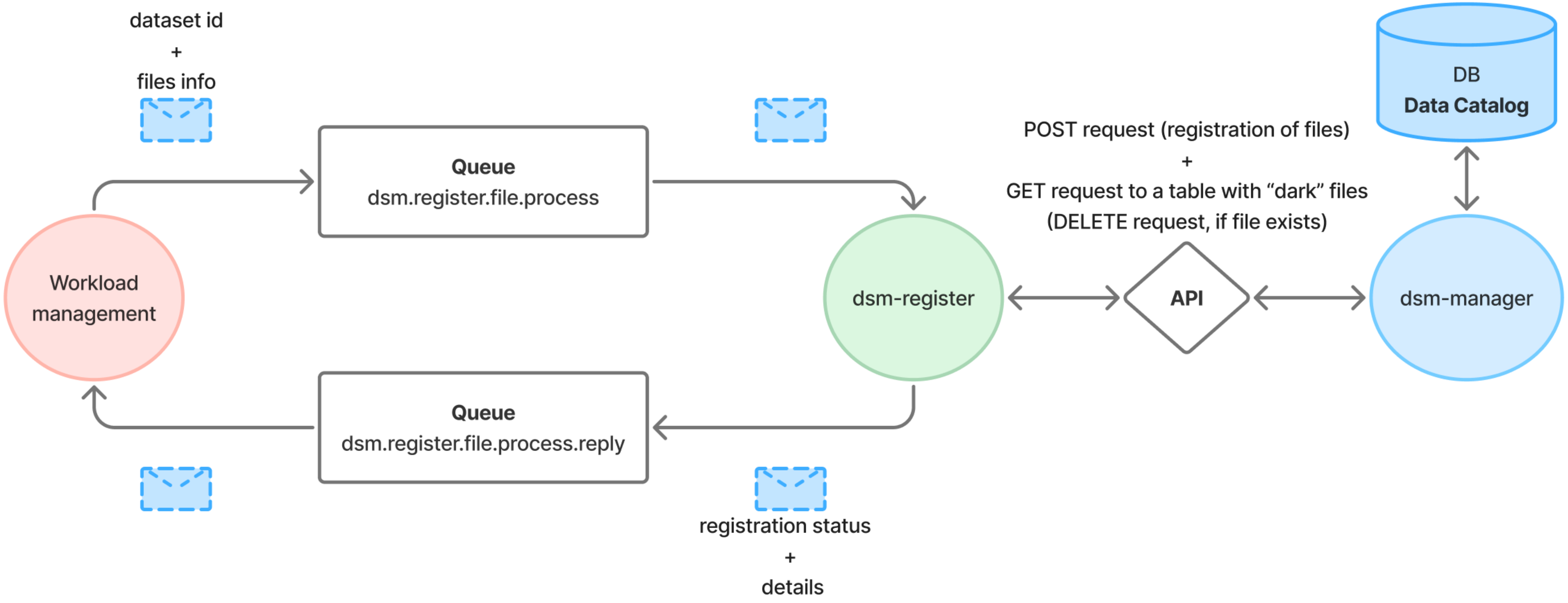
⇓

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

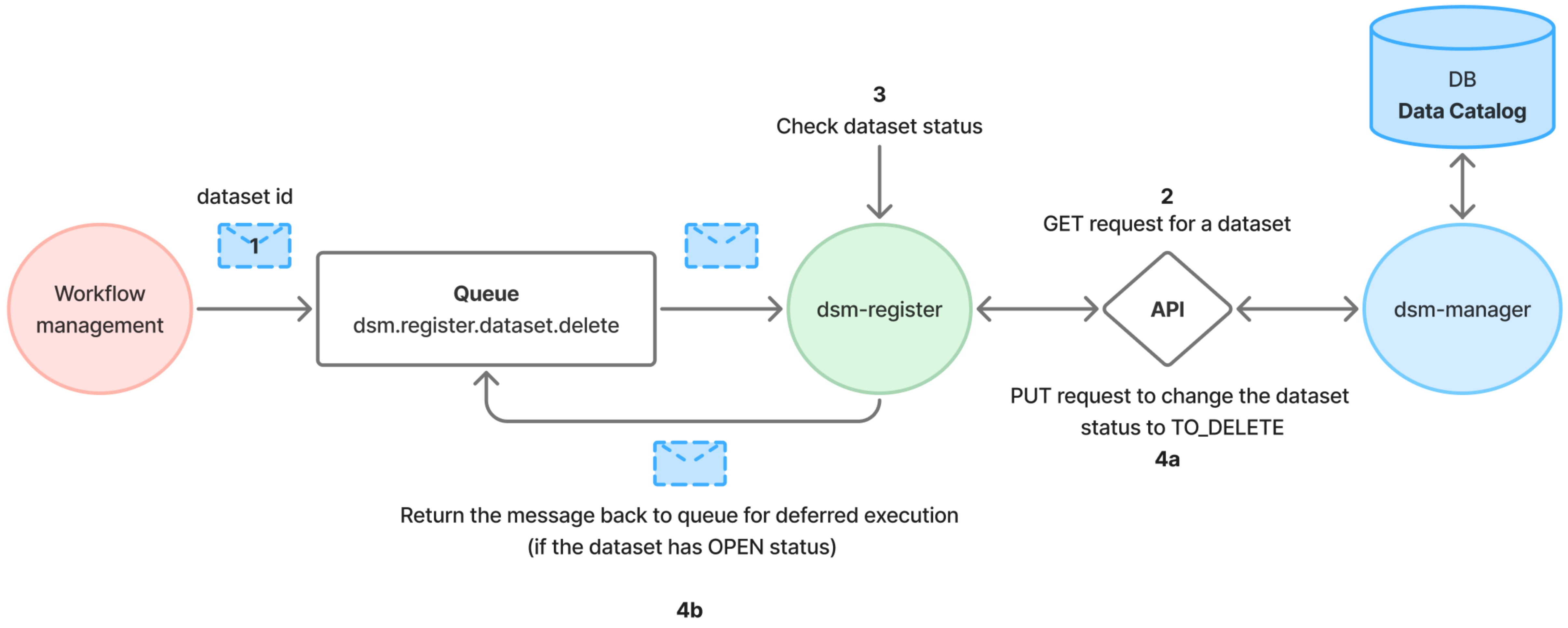
dsm.register.file.input и dsm.register.dataset.input


information about input files
DAQ
Queue
dsm.register.file.input
dsm-register
POST request to create a new dataset
POST request to registration input files
PUT request to change dataset status
+
GET request to a table with "dark" files
(DELETE request, if file exists)
API
DB
Data Catalog
dsm-manager
Workflow management
Queue
dsm.register.dataset.input
dataset id and name

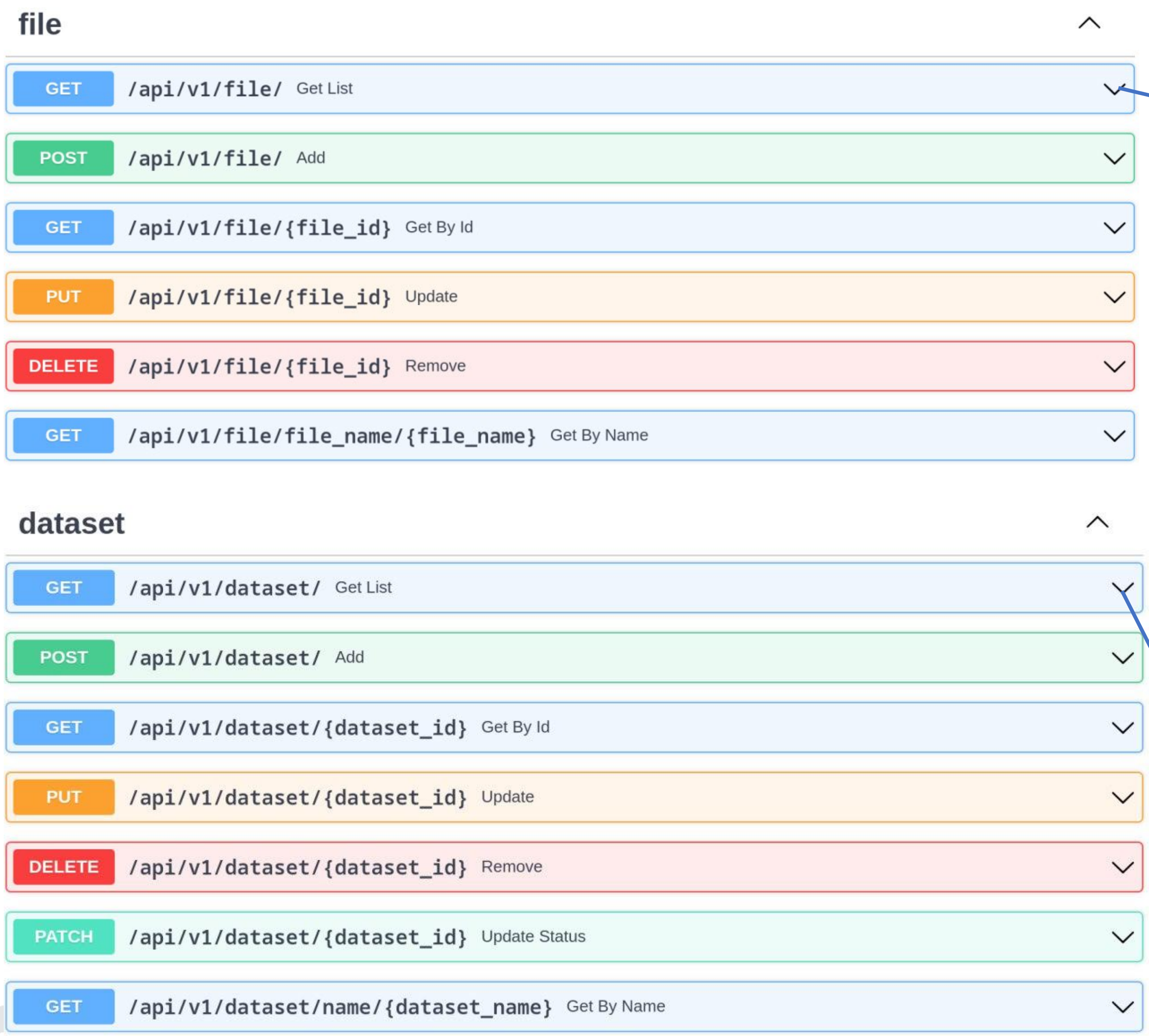
dsm.register.file.process and dsm.register.file.process.reply


dataset id
+
files info
Queue
dsm.register.file.process
DB
Data Catalog
POST request (registration of files)
+
GET request to a table with “dark” files
(DELETE request, if file exists)
Workload management
dsm-register
API
dsm-manager
Queue
dsm.register.file.process.reply
registration status
+
details

dsm.register.dataset.delete


3
Check dataset status
DB
Data Catalog
dataset id
1
2
GET request for a dataset
Workflow management
Queue
dsm.register.dataset.delete
dsm-register
API
dsm-manager
PUT request to change the dataset status to TO_DELETE
4a
Return the message back to queue for deferred execution (if the dataset has OPEN status)
4b

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

FastAPI

The service must provide a REST API to the database

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

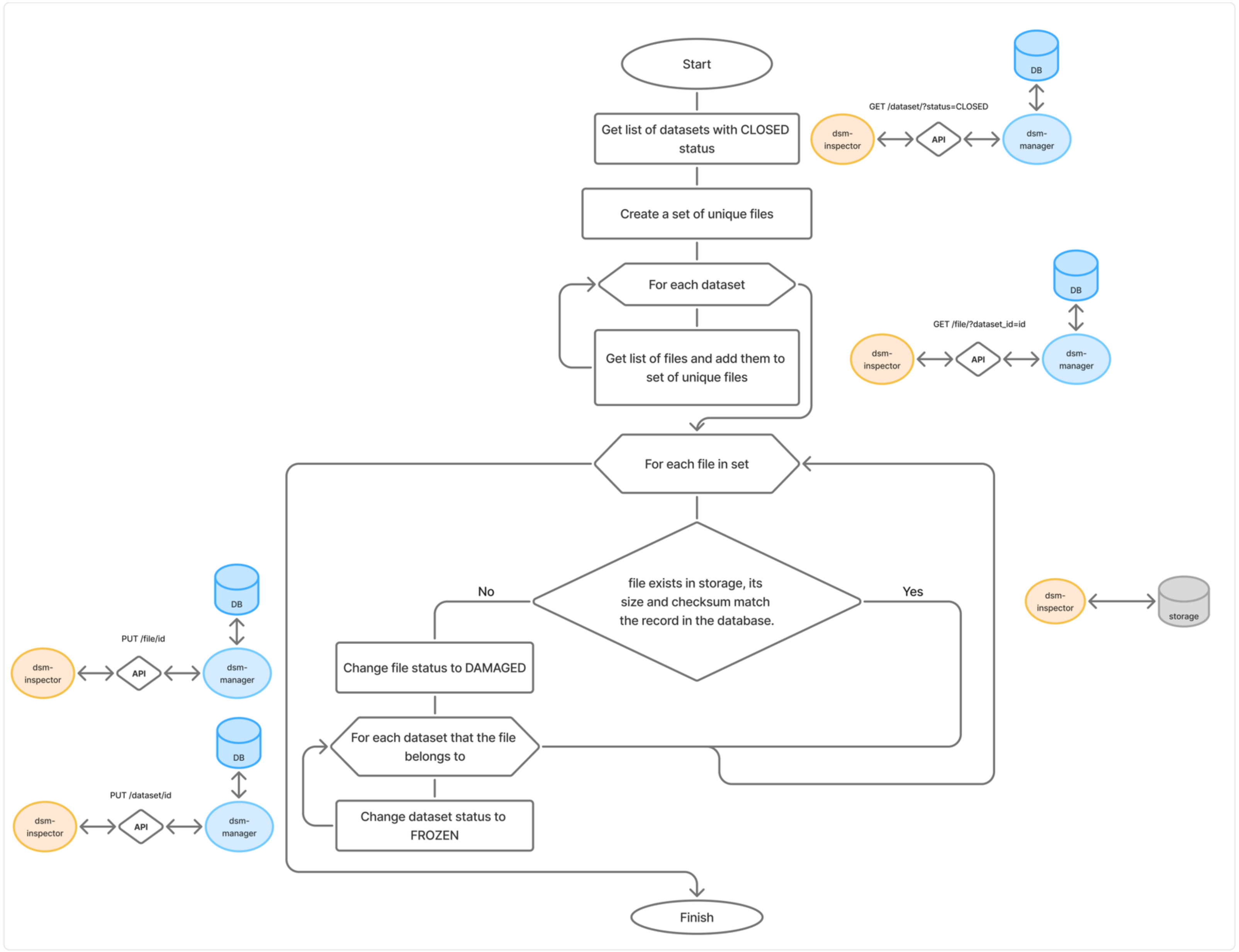
- ✓ Getting a list of datasets that contain a specific file
- ✓ Getting a list of datasets with a specific status

dsm-inspector

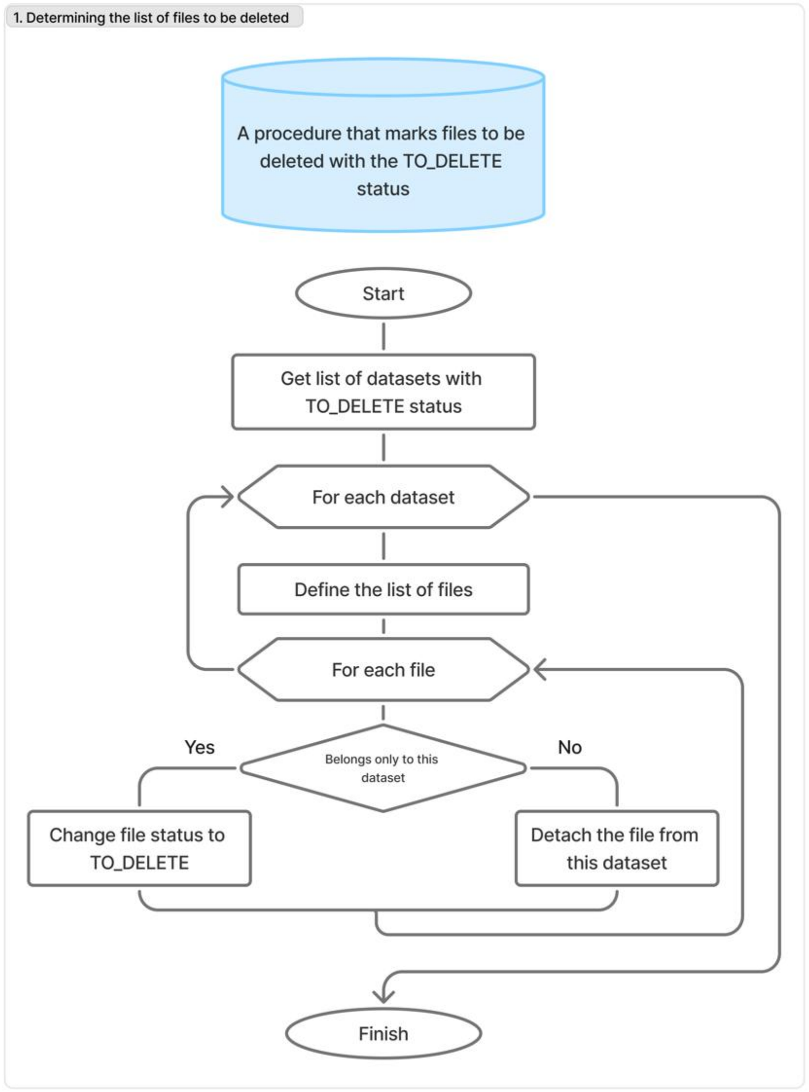
The service consists of a set of **background** tasks:

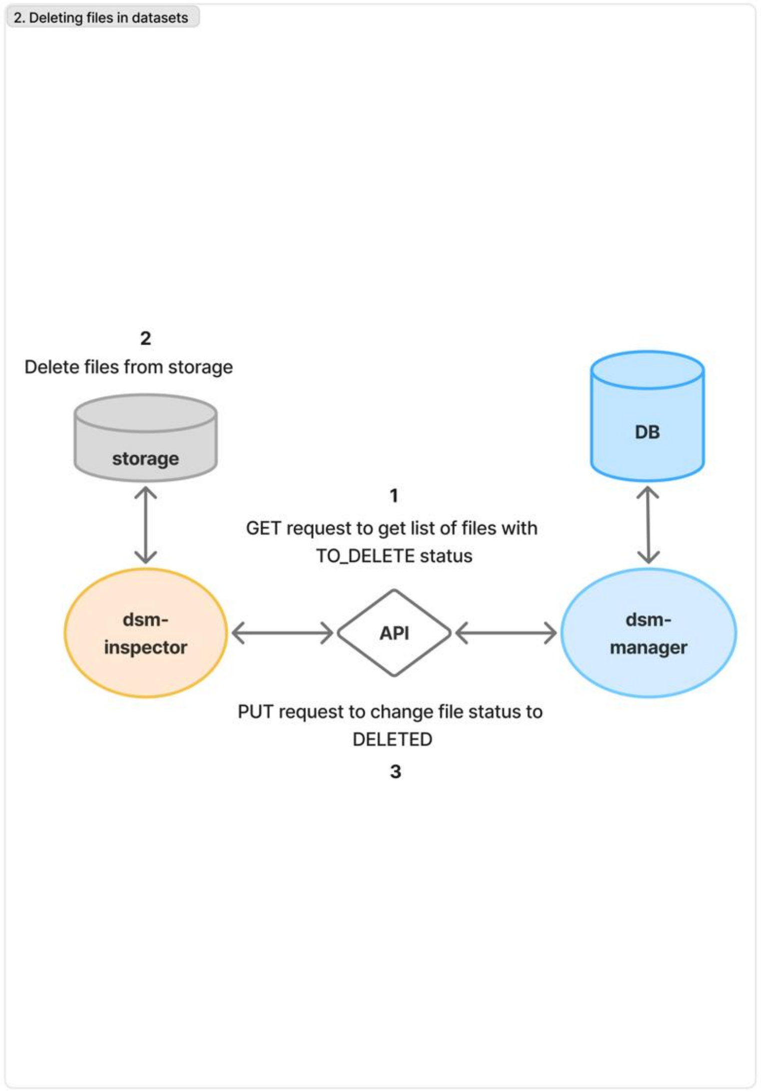
- Deleting files on storages
- File upload control
- File integrity check
- Monitoring storage usage

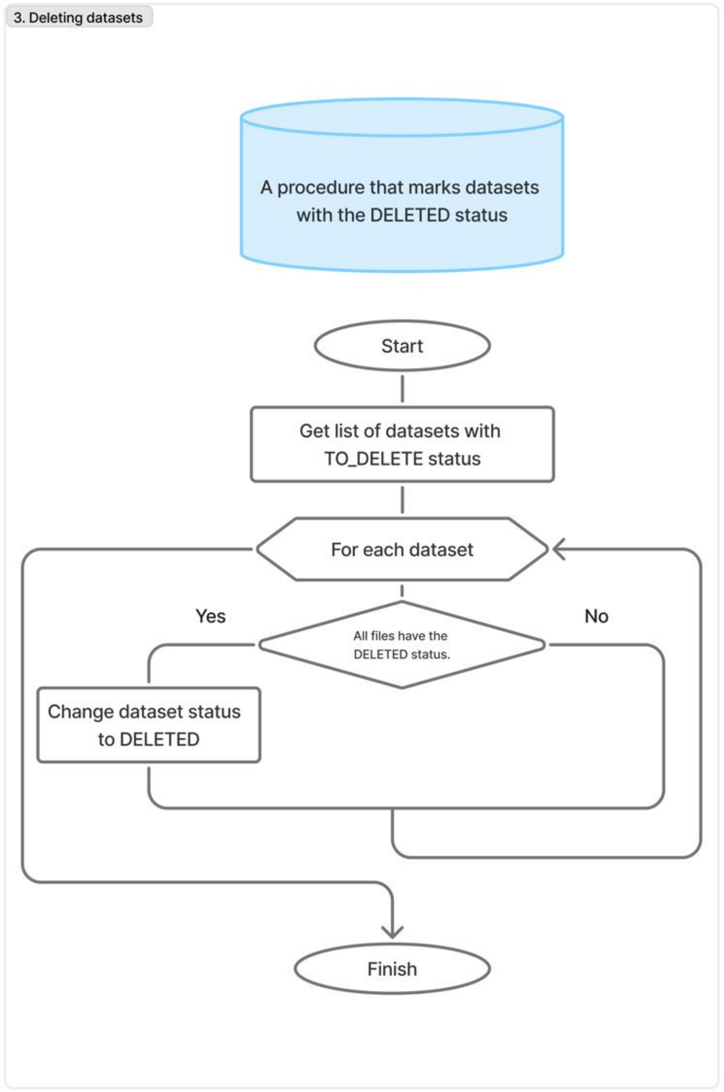
File Integrity Check


Start
Get list of datasets with CLOSED status
GET /dataset/?status=CLOSED
DB
dsm-inspector
API
dsm-manager
Create a set of unique files
For each dataset
Get list of files and add them to set of unique files
GET /file/?dataset_id=id
DB
dsm-inspector
API
dsm-manager
For each file in set
file exists in storage, its size and checksum match the record in the database.
No
Yes
dsm-inspector
storage
Change file status to DAMAGED
PUT /file/id
DB
dsm-inspector
API
dsm-manager
For each dataset that the file belongs to
Change dataset status to FROZEN
PUT /dataset/id
DB
dsm-inspector
API
dsm-manager
Finish

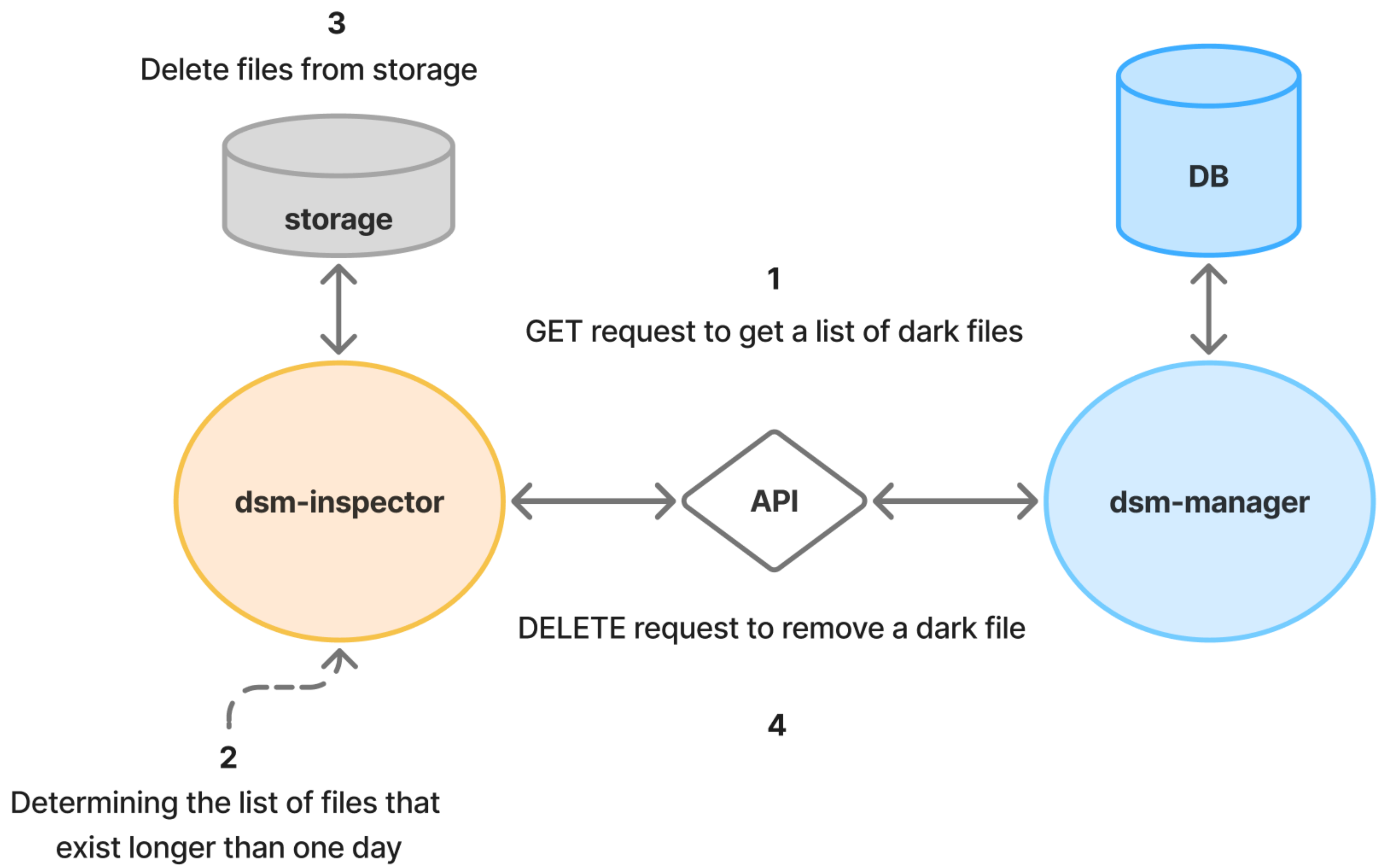
Deleting datasets and files


1. Determining the list of files to be deleted
A procedure that marks files to be deleted with the TO_DELETE status
Start
Get list of datasets with TO_DELETE status
For each dataset
Define the list of files
For each file
Belongs only to this dataset
Yes
No
Change file status to TO_DELETE
Detach the file from this dataset
Finish

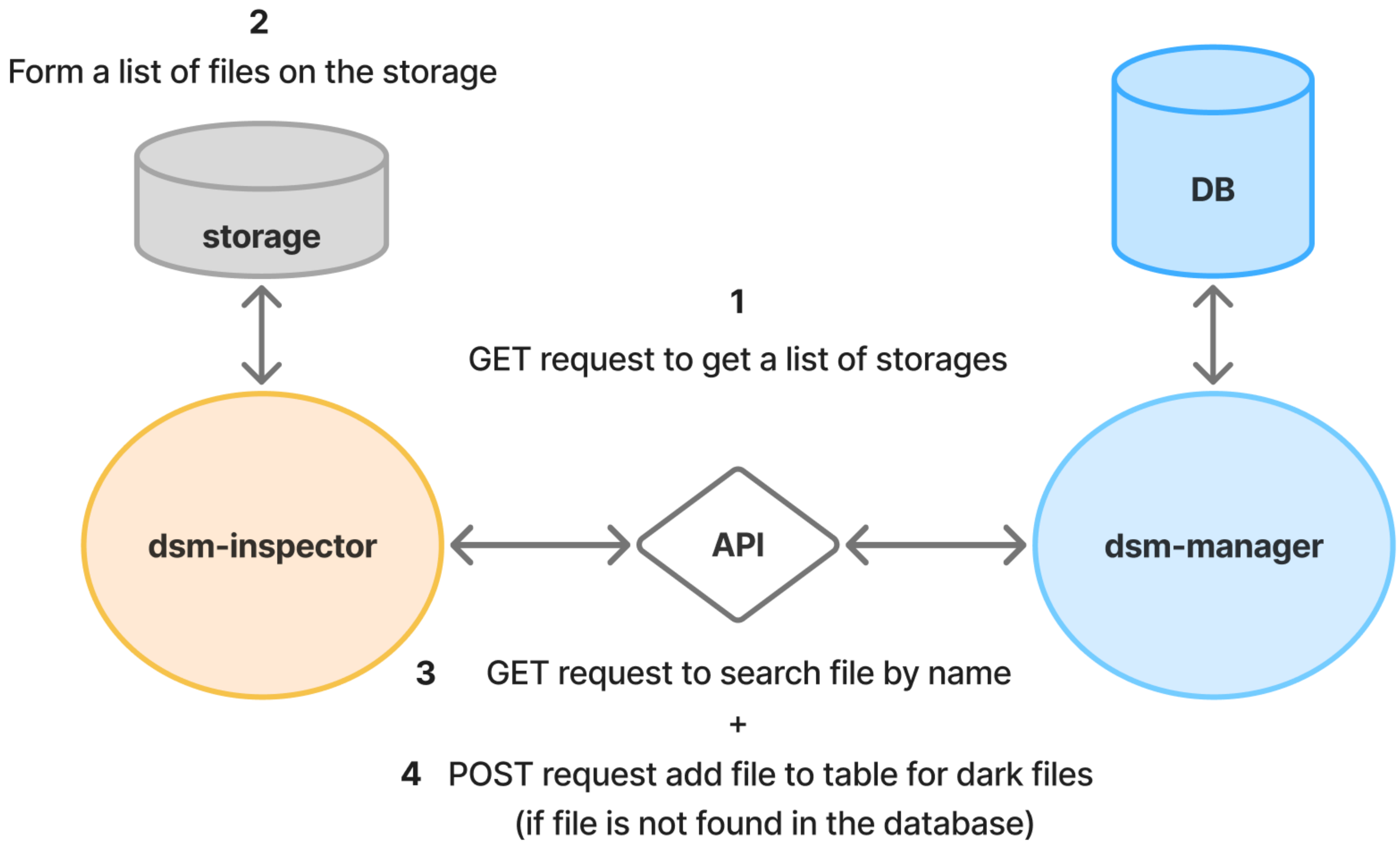

2. Deleting files in datasets
2
Delete files from storage
storage
DB
1
GET request to get list of files with TO_DELETE status
dsm-inspector
API
dsm-manager
PUT request to change file status to DELETED
3


3. Deleting datasets
A procedure that marks datasets with the DELETED status
Start
Get list of datasets with TO_DELETE status
For each dataset
Yes
No
All files have the DELETED status.
Change dataset status to DELETED
Finish

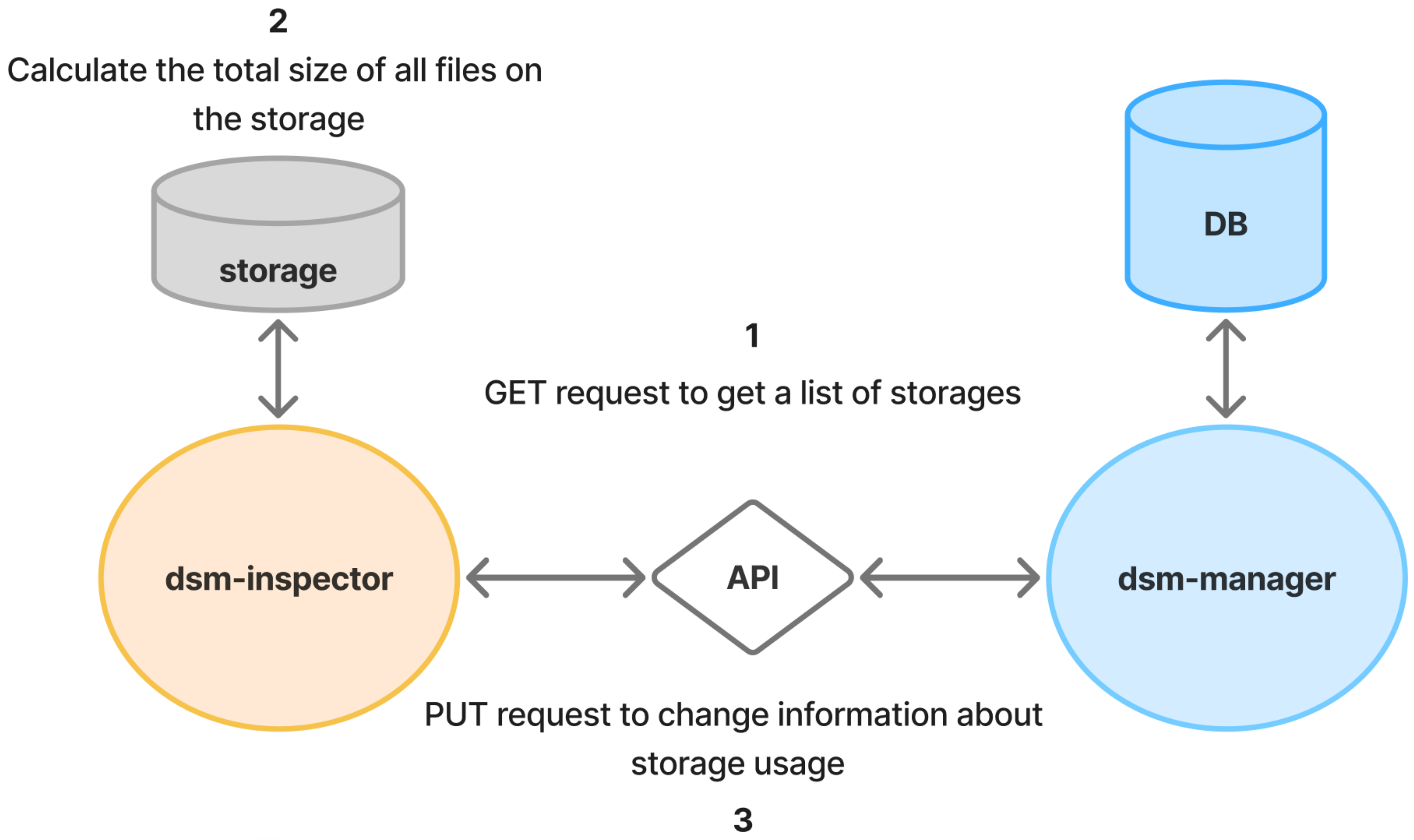
Deleting dark files


3
Delete files from storage
storage
DB
1
GET request to get a list of dark files
dsm-inspector
API
dsm-manager
DELETE request to remove a dark file
4
2
Determining the list of files that exist longer than one day

Storage usage monitoring: monitoring of dark files


2
Form a list of files on the storage
storage
DB
1
GET request to get a list of storages
dsm-inspector
API
dsm-manager
3 GET request to search file by name
+
4 POST request add file to table for dark files
(if file is not found in the database)

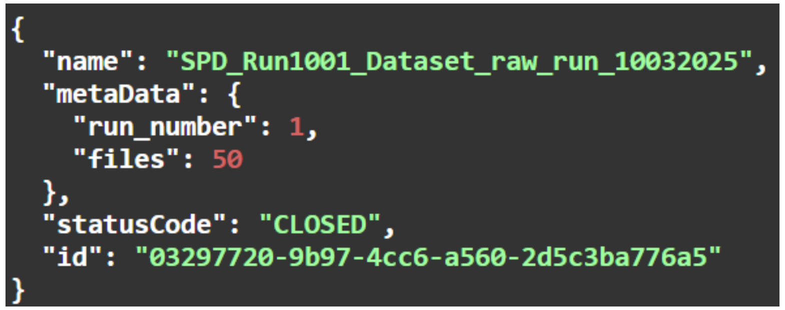
Storage usage monitoring


2
Calculate the total size of all files on the storage
storage
DB
1
GET request to get a list of storages
dsm-inspector
API
dsm-manager
PUT request to change information about storage usage
3

SPD DAQ generator

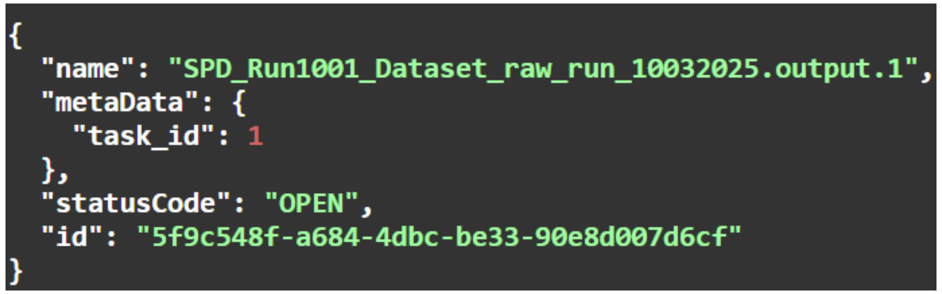
- Using SPD DAQ data generator, 50 2 GB files were generated (~ 7 minutes per file)

- An input dataset with these files has been created



```
{
  "name": "SPD_Run1001_Dataset_raw_run_10032025",
  "metaData": {
    "run_number": 1,
    "files": 50
  },
  "statusCode": "CLOSED",
  "id": "03297720-9b97-4cc6-a560-2d5c3ba776a5"
}
```

- A task has been generated and an output dataset has been created



```
{
  "name": "SPD_Run1001_Dataset_raw_run_10032025.output.1",
  "metaData": {
    "task_id": 1
  },
  "statusCode": "OPEN",
  "id": "5f9c548f-a684-4dbc-be33-90e8d007d6cf"
}
```

- The files were successfully processed by the pilots

Conclusion

Current results:

- ✓ **dsm-manager** fully functional for this stage of implementation
- ✓ **dsm-register** implemented for this stage
- ✓ **dsm-inspector** is implemented by 75%

Further plans:

dsm-inspector:

Implement background services for

- ➢ File upload control

dsm-register:

Realize processing of messages from queues:

- ➢ dsm.register.dataset.upload