JOINT INSTITUTE FOR NUCLEAR RESEARCH

Software meeting

Design of the BM@N Production Request System for Mass Productions of Reconstructed Data

14:00 14/05 2026



Alexander Chebotov
Konstantin Gertsenberger
Ilya Romanov
Igor Pelevanyuk

BM@N

16th Collaboration Meeting of the BM@N Experiment at NICA

OUTLINE

- Motivation & Problem
- System Architecture
- Request Lifecycle
- Production
- Results & Demo
- Future Roadmap

„A new day provides new insights“



Motivation

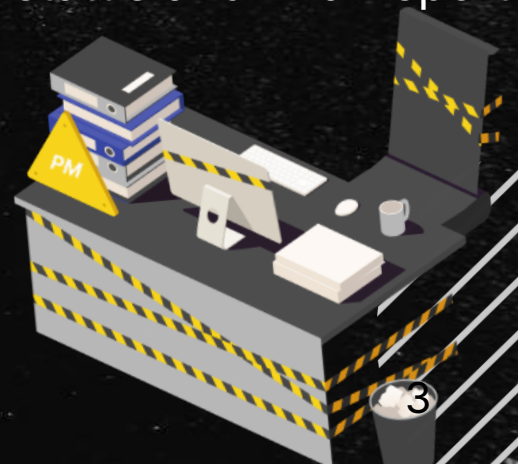


- Production Request System — single entry point for official mass processing of BM@N experimental and simulated data
- Submit — a request containing software version · dataset type · data location · file limit · additional information (keep intermediates + comment)
- Probe — a preliminary test before the main DST production
- Monitor — in real time: status, current stage, number of processed/failed files
- Manage — lifecycle: approve, cancel at any stage, restart failed jobs
- Browse — full history: all requests and its parameters, status and final report

DIRAC
THE INTERWARE

DIRAC
Manager
Igor Pelevanyuk





Current Workflow

DST Production Requests

Physics Analysis Issues

gertsen 12 Oct 2021

Requests on DST productions should be applied in separate topics.

Please, specify the following parameters required for a request in a new topic:

1. BmnRoot tag/release (or dev if there are some problems with the last release).
2. Directories at the official EOS storage with input generator files or raw experimental data to be processed for the given DST production.
   or
   Event Generator (for sim) / Run number (for exp data) : Beam : Target : Energy.
3. All events in the given directories or a desired number of events.
4. (optional) additional parameters of the processing if they differ from the default ones.
5. (optional for simulation data) a number of intermediate *sim* files to be not erased

NOTE: If no generator files exist for the given beam-target-energy for the DST production, please, contact the corresponding generator group before.

Request #15: 10M DCM-SMM modified Xe-CsI@3.8 AGeV

Physics Analysis Issues

merz Mar 30

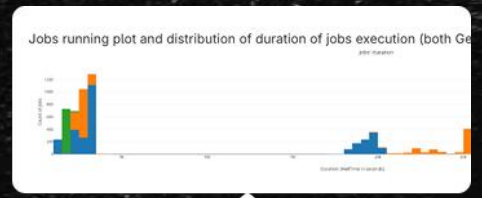
I would like to request the bmnsim and bmndst production

1. BmnRoot **dev**
2. /eos/nica/bmn/sim/gen/DCMSMM/DCMSMMmod_XeCsI_3.8AGeV_mb_10K
3. all events (1e7 events)
4. -
5. Save all sim files

DIRAC Workload Manager





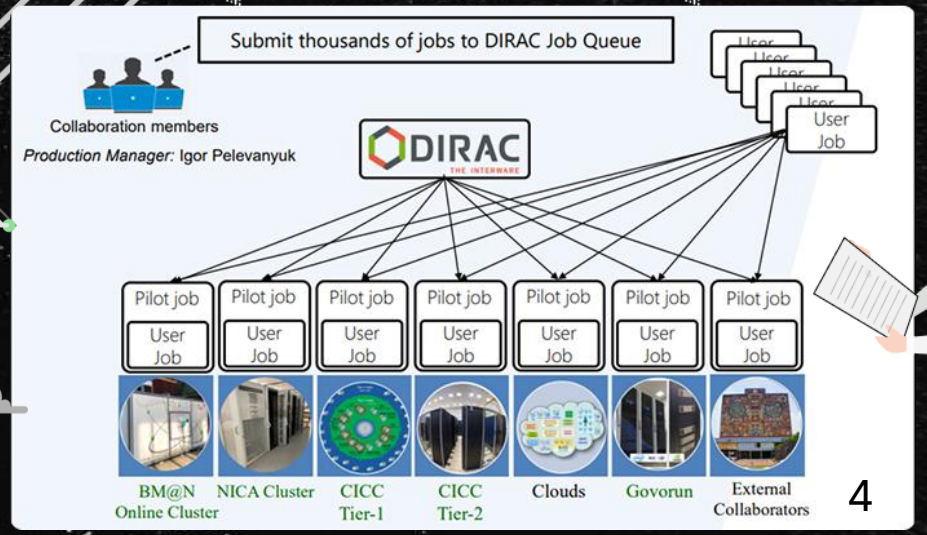


ipelevan 27d

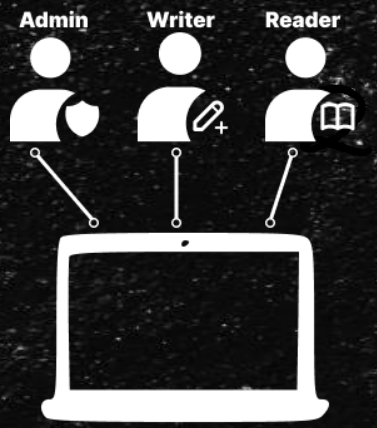
Done!

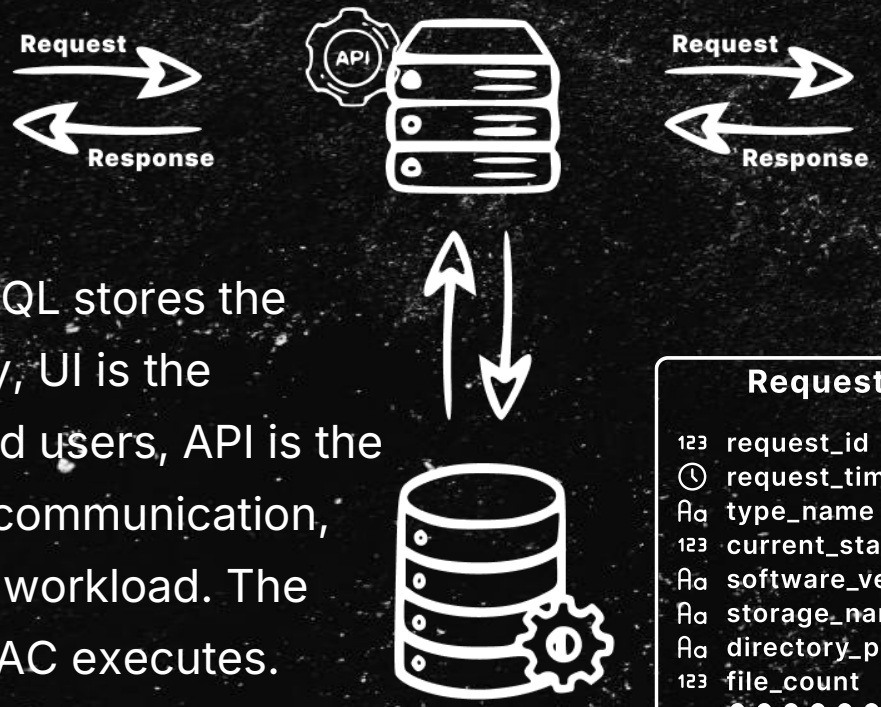
2026_03_30 #15 SIM DST DCMSMM modified XeCsI@3.8 AGeV · Wiki · IgorPelevanyuk _ DIRAC4BMN · GitLab.pdf (604.9 KB)

Reply



System Architecture





DIRAC



Three layers — PostgreSQL stores the request and its full history, UI is the frontend for operators and users, API is the interface for component communication, DIRAC scripts handle the workload. The system orchestrates; DIRAC executes.

Request

123	request_id
🕒	request_time
Aa	type_name
123	current_status_id
Aa	software_version
Aa	storage_name
Aa	directory_path
123	file_count

Request Stage

123	request_id
Aa	type_name
123	stage_step
123	is_test
123	processed_count
123	failed_count



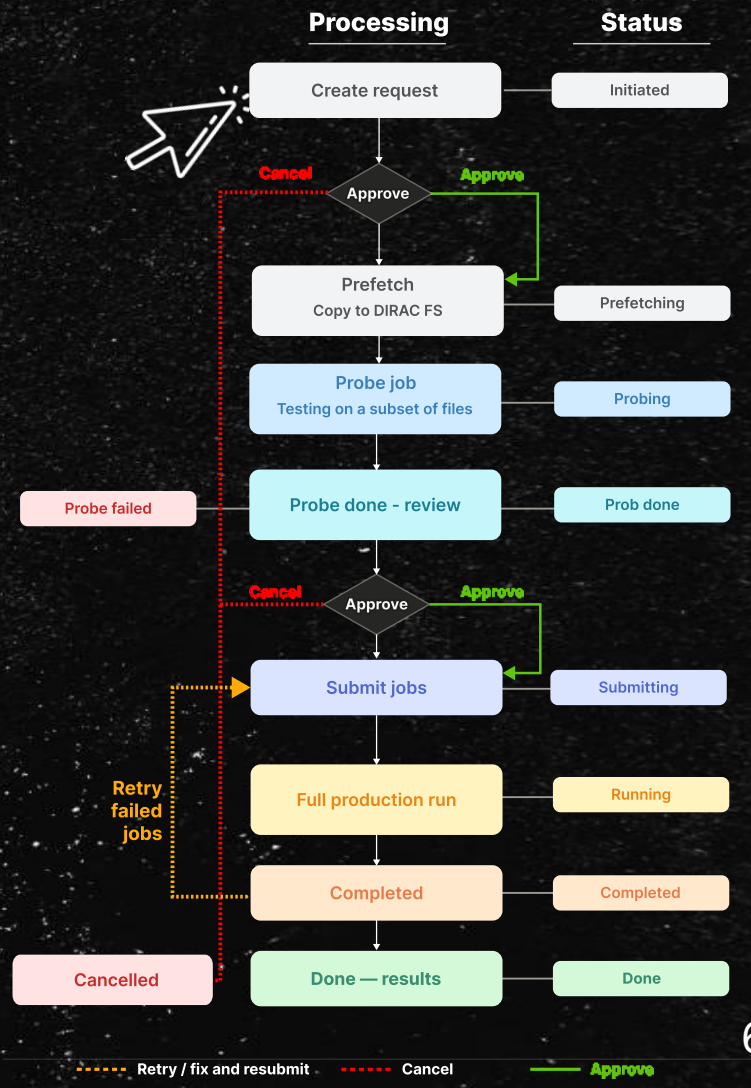
New Request Lifecycle

Probe phase — A test job runs on a subset of files first. Catches software or hardware errors and measures resource usage before committing to the full production.

Two approval gates — Operator confirms requests before the probe, then reviews probe output before approving the full production.

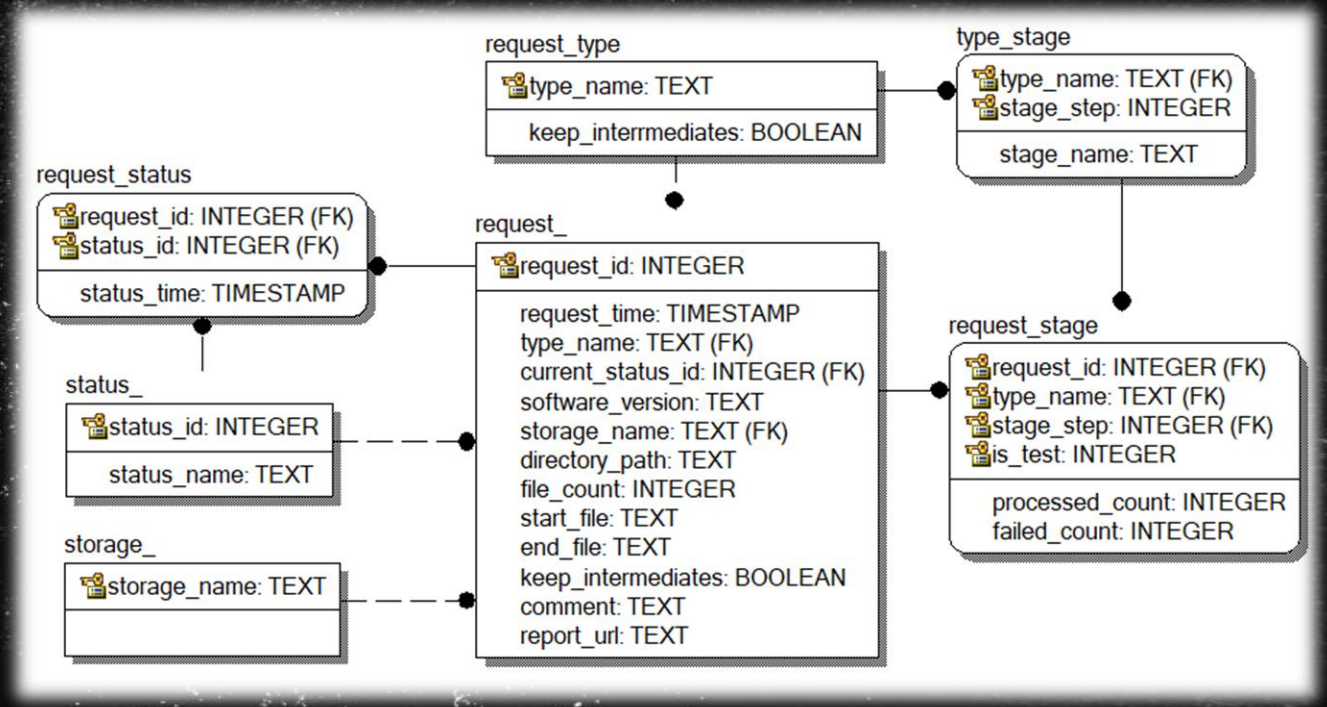
Restart — After a full production the failed jobs are re-run.

Cancel at any point — The request can be cancelled from any state.



Production Database

- PostgreSQL — relational DBMS
- Append-only history — every status transition is recorded with timestamp
- Two request types:
  - `exp` (RawToDigi → DigiToDst)
  - `sim` (GenToSim → SimToDst)
- Core tables:
  - `request_` (params + state)
  - `request_status` (status history) ·
  - `request_stage` (per-stage counters)



The structure allows us to reconstruct the full context of a request at any time: who, when, with what parameters, what was done, what failed.

API Layer

- Stack: APIKit microframework (BM@N standard, built on FastAPI) — async, idempotency, type validation, CRUD, KV storage
- Lifecycle enforcement — illegal state transitions rejected
- Auth (planned): Keycloak (JINR SSO via OIDC)
- Roles: Reader (view) · Writer (create requests) · Admin (approve, restart failed jobs, cancel)

Endpoints by purpose:

- *`POST /requests` — create + validate*
- *`GET /requests` — list, filter, paginate*
- *`GET /requests/{id}` — config + per-stage progress*
- *`POST /requests/{id}/approve` · `cancel` · `restart failed jobs `*



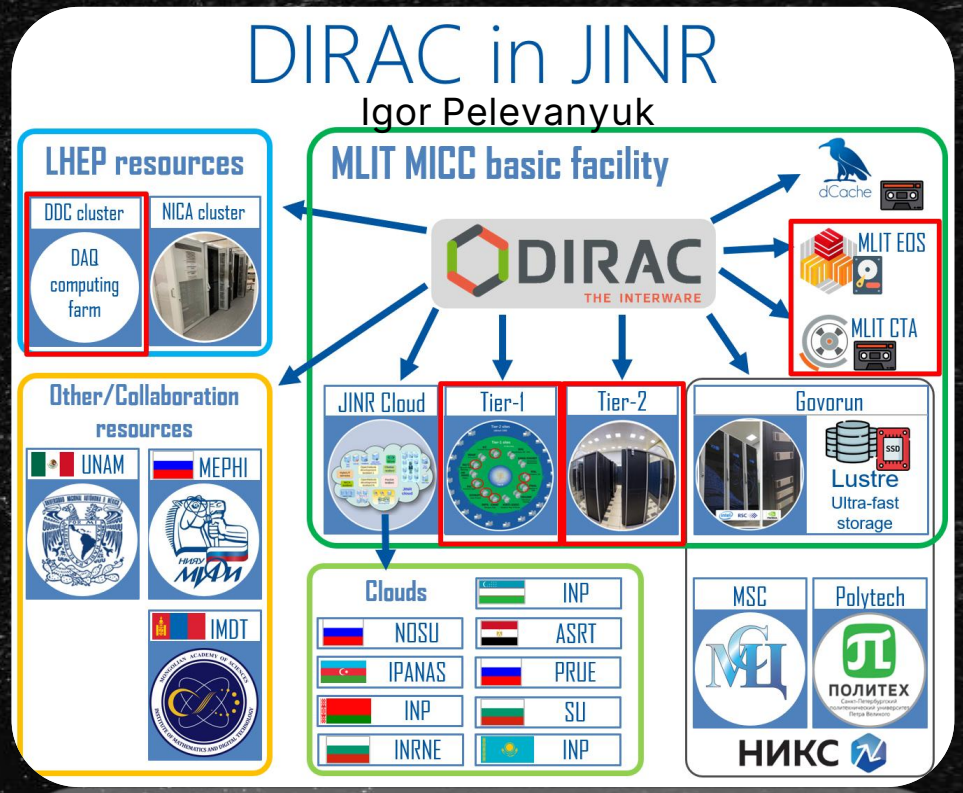


DIRAC

DIRAC Middleware

What DIRAC is — **Workload Manager on Distributed Infrastructure with Remote Agent Control** — *middleware that hides heterogeneous compute behind one submission API.*

Why we need it — BM@N compute is spread across separated computing platforms with different batch systems, auth and storage systems. DIRAC gives one API for all of them: submission, automatic placement, FileCatalog (logical → physical paths), per-experiment accounting.



BM@N computing platforms accessible via DIRAC:

- CICC (MLIT) — GRID Tier1/Tier2: own SLURM batch and EOS storage
- NICA cluster (`NCX`, VBLHEP) — own SLURM batch and EOS storage
- Govorun (HybriLIT, MLIT) — own SLURM `bmn` queue
- DAQ Data Center (DDC, VBLHEP) — own SLURM batch and CephFS storage

Web Interface

Stack: React (library) + Mantine (component library) + Feature-Sliced Design (architectural methodology) + Bun (toolkit) — BM@N standard

Type: Single Page Application, served behind reverse proxy alongside the API

Talks only to the API — no direct DB or DIRAC access; clean separation of concerns

Auth (planned): OIDC redirect to Keycloak (JINR SSO); session via API's BFF layer (tokens stay server-side)

Role-based views: Reader (view) · Writer (create requests) · Admin (approve, restart failed jobs, cancel)







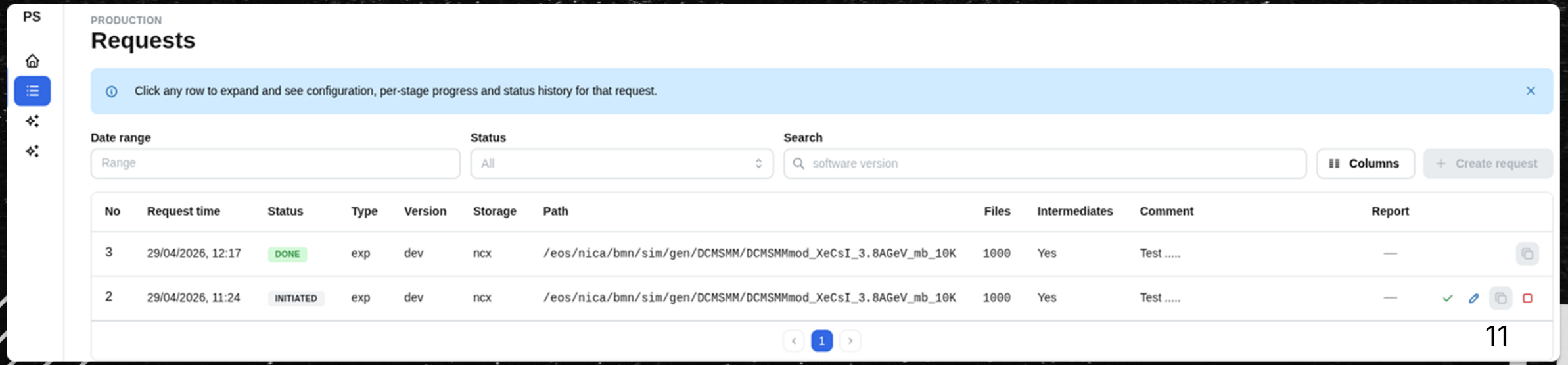
Production Features

- Full production runs on BM@N resources including the NICA Cluster, MLIT CICC, Govorun Supercomputer and DAQ Data Center via DIRAC
- The system polls DIRAC and updates information on request status, the current stage and numbers of processed & failed files
- Failed jobs surfaced per stage in the operator UI; re-run resubmits only the failed ones
- Auto-generated final report → uploaded to S3, attached to the request



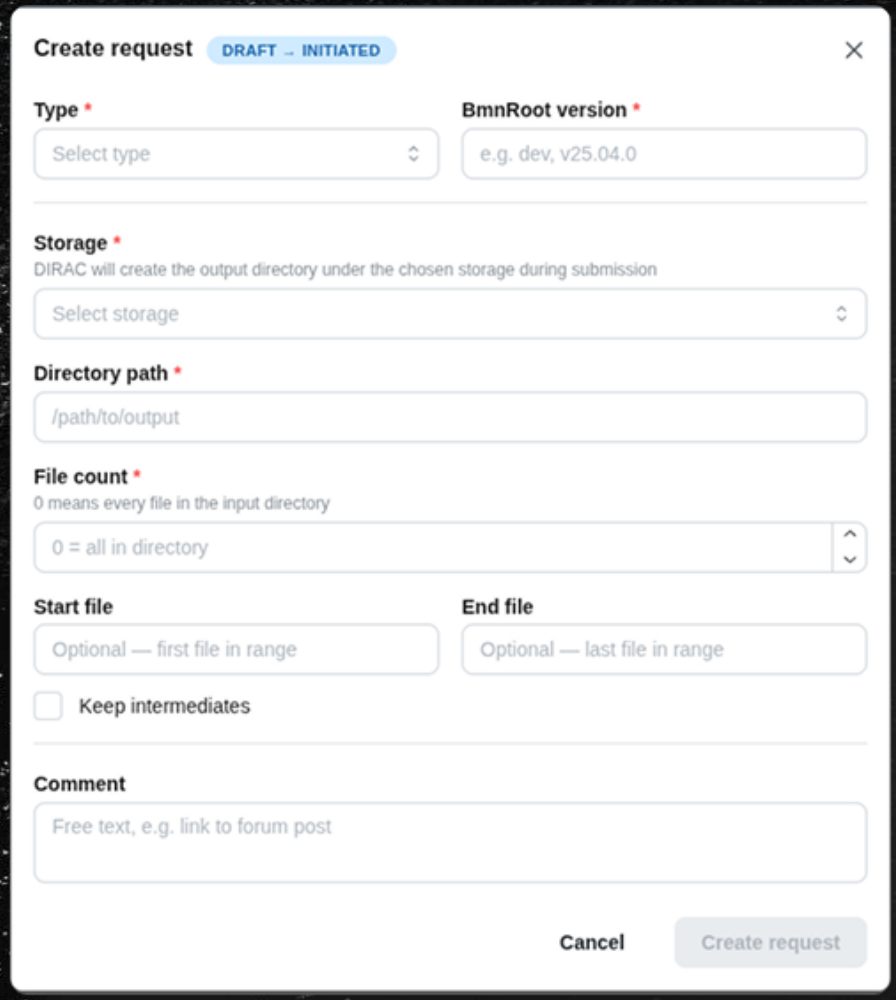
Request

- request_id
- request_time
- type_name
- current_status_id
- software_version
- storage_name
- directory_path
- file_count



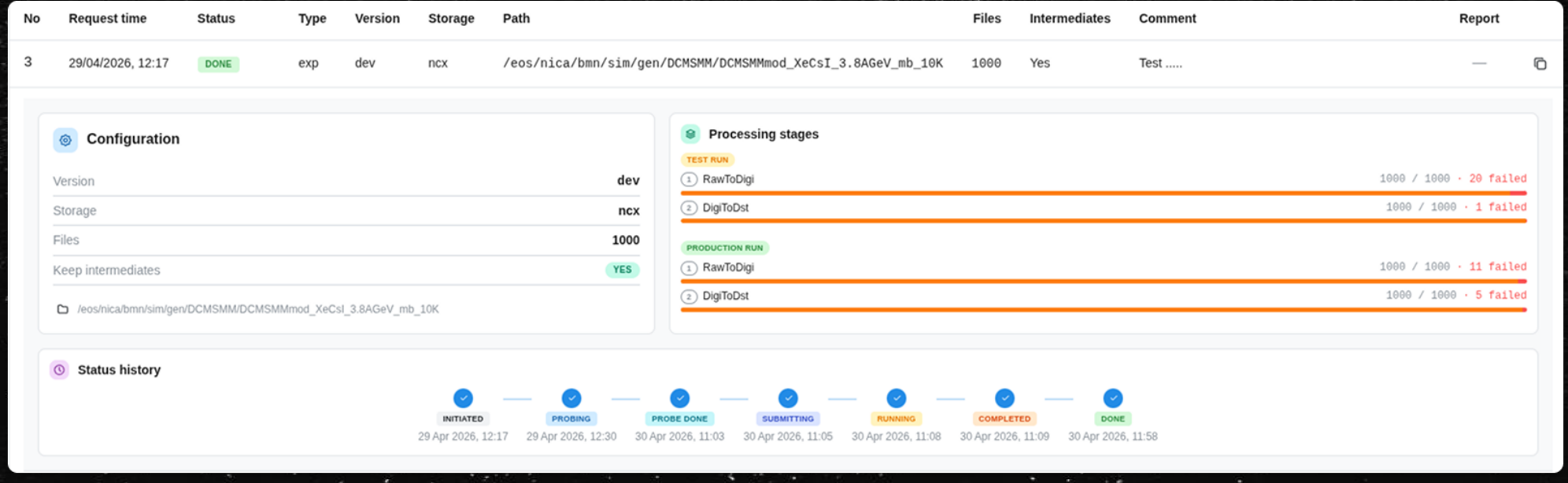
Create Production Request

- Web form replaces the forum-post spec
- Structured input — request type, software version, data location...
- Automatic validation — no missing required fields, no path typos...



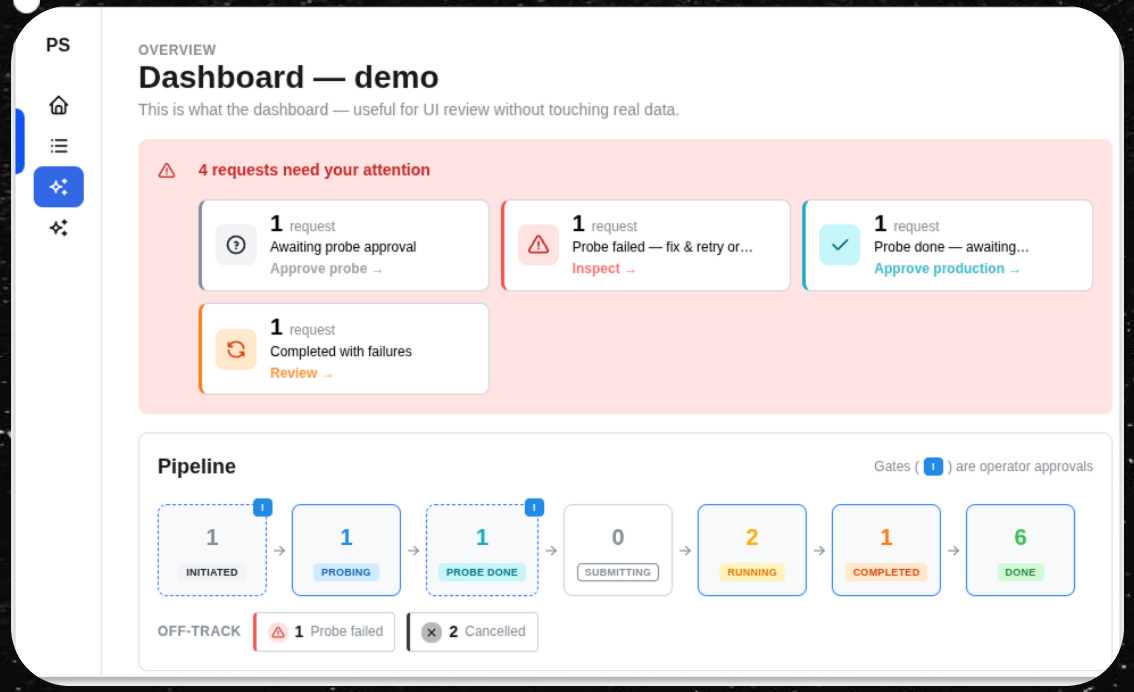
Browse & Monitor

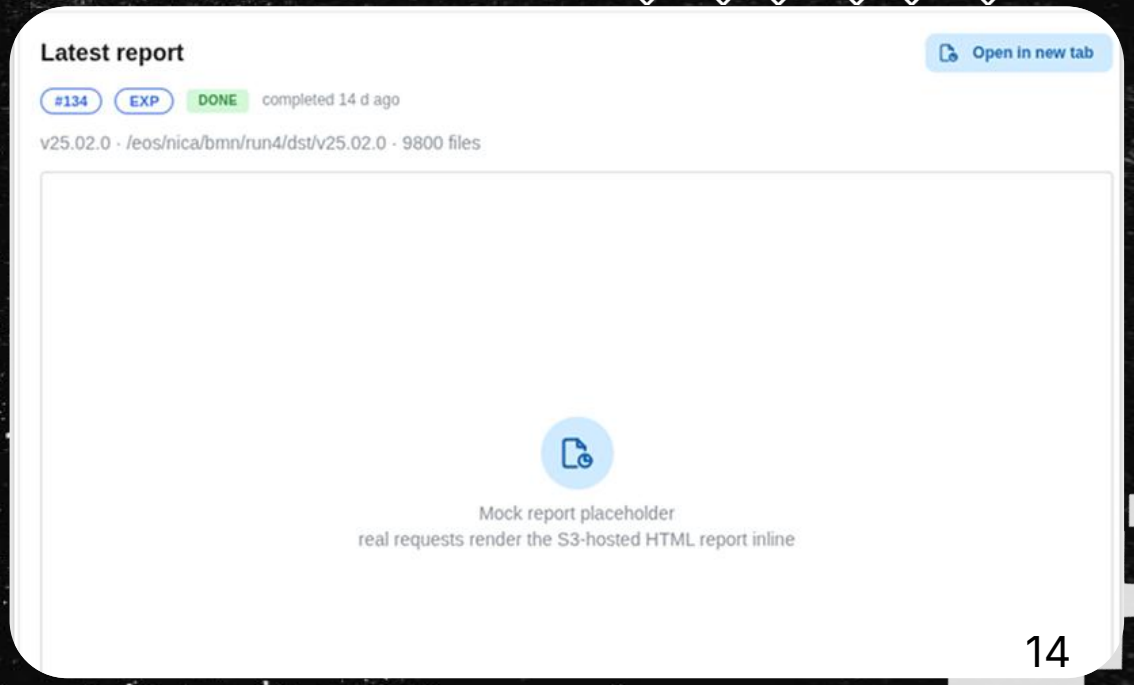
- ❖ All requests in one table — sortable, filterable, searchable
- ❖ Per-request detail view — configuration + per-stage progress in one place
- ❖ Centralized history — no more separated forum threads
- ❖ Anyone can check status — no more asking the operator
- ❖ Dashboard with statistics — full statistics is available at any time



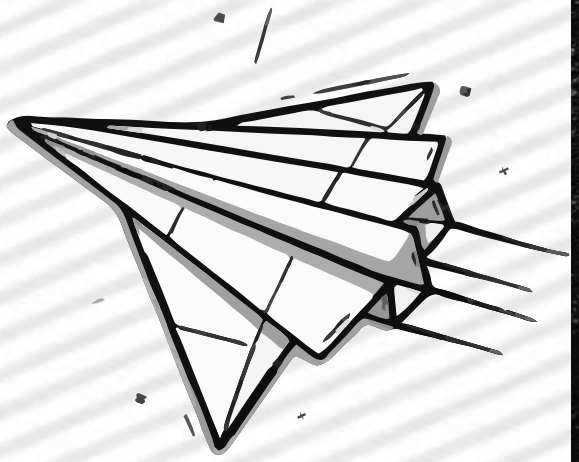
Dashboard Demo

- Single-page dashboard — no more tracking state in the operator's head
- Action-driven cards — what needs your attention right now
- Visual pipeline — every request and states are visualized
- Latest report — quick access to the finished request by one click





Future Roadmap



- DIRAC API integration — connect the submission/polling layer; primary next milestone
- Integration of notification on state transitions – email / messenger
- Integration with JINR SSO / Keycloak for user identification





CONCLUSIONS

- **Motivation & Problem**
- **System Architecture**
- **Request Lifecycle**
- **Production**
- **Results & Demo**
- **Future Roadmap**

- ❖ Project plan + architecture defined
- ❖ Request state machine designed and implemented
- ❖ PostgreSQL database deployed with full schema
- ❖ REST API on APIKit — major part implemented (CRUD, lifecycle enforcement)
- ❖ Web Interface — major part developed (dashboard, create form, browse & detail)

JOINT INSTITUTE FOR NUCLEAR RESEARCH

Software meeting

Thank you for your attention!



Alexander Chebotov
Konstantin Gertsenberger
Ilya Romanov
Igor Pelevanyuk

BM@N

16th Collaboration Meeting of the
BM@N Experiment at NICA