Development of new Event Display and auxiliary software for BM@N experiment

Igor Dunaev

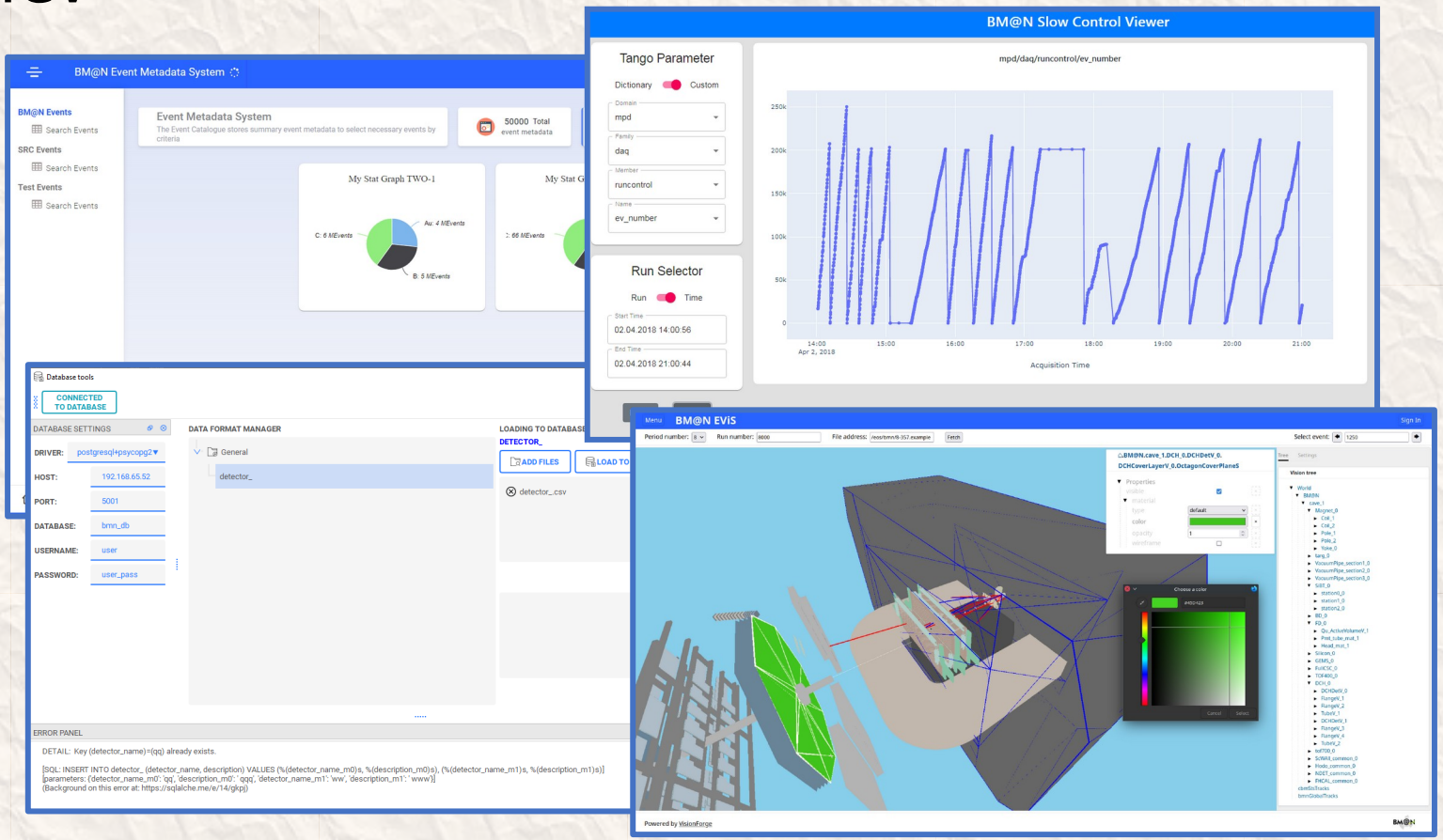
the MIPT team for the BM@N collaboration

MIPT Software for BM@N – Team

Supervision: T. A.-Kh. Aushev

Team members:

- P. Klimai
- A. Nozik
- I. Dunaev (student 6y)
- O. Nemova (student 6y)
- V. Kaplenko (student 6y)
- A. Degtyarev (PhD st. 2y)

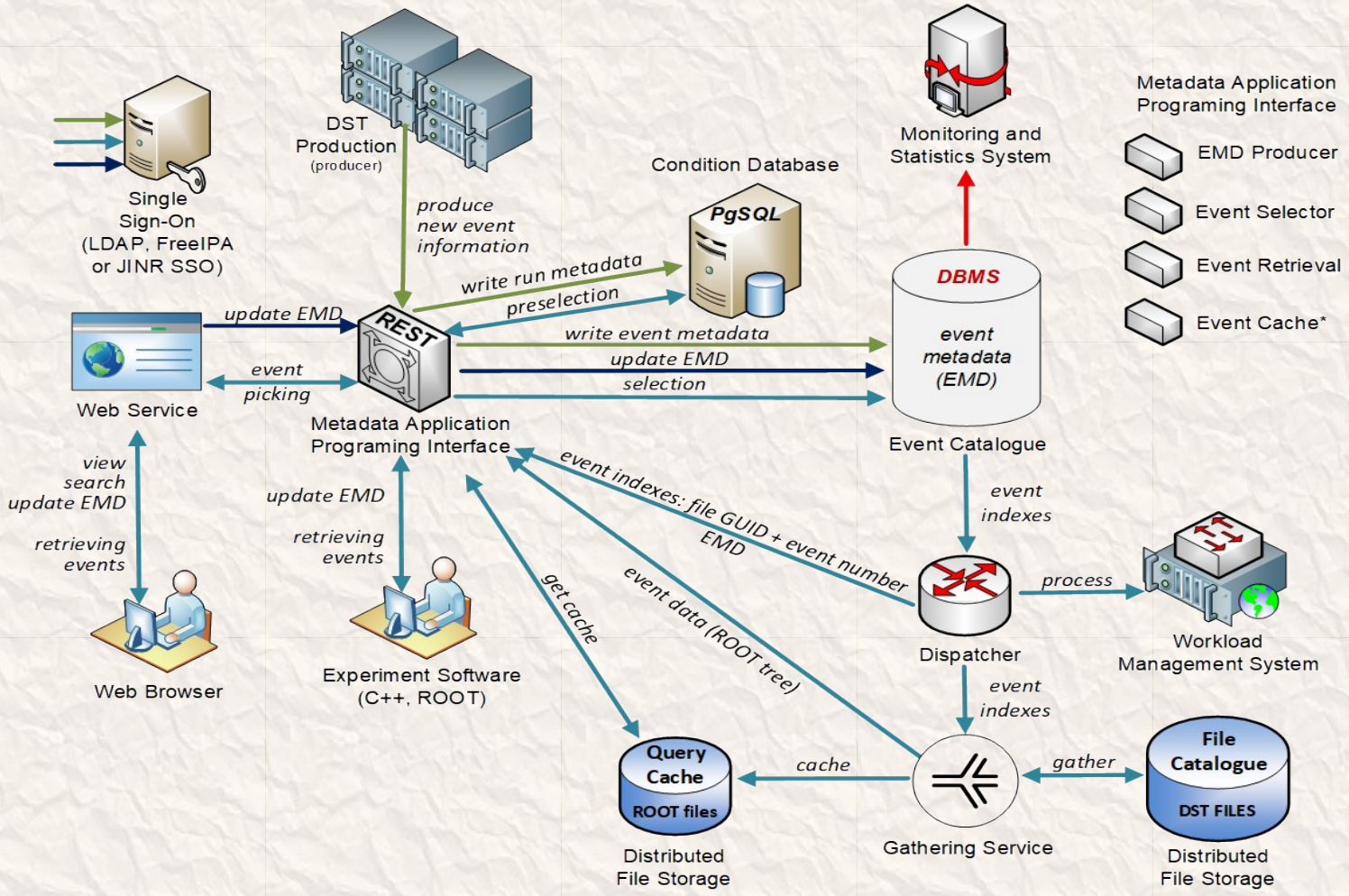


Main Projects Summary

Project	URL
Event Metadata System	https://git.jinr.ru/nica_db/emd https://git.jinr.ru/pklimai/ems-deploy
Next-generation Event Display	https://git.jinr.ru/nica-visualization/visionforge https://git.jinr.ru/nica-visualization/visapi https://git.jinr.ru/nica-visualization/evis
Monitoring Service	https://git.jinr.ru/nica-computing/mon-service-deploy
REST API and Web interface for slow control system	https://git.jinr.ru/nica_db/tango_web https://git.jinr.ru/nica_db/tango-rest

Event Metadata System (an update)

BM@N Event Metadata System



- Event Metadata System
  - Event Catalogue based on PostgreSQL
  - Integration with BM@N Condition database
  - REST API and Web UI based on Kotlin Multiplatform
  - Configurable to support different metadata
  - ROOT macro to fill the catalogue
  - Automatic deployment
  - High Availability solution
  - Statistics collection and display
  - Monitoring

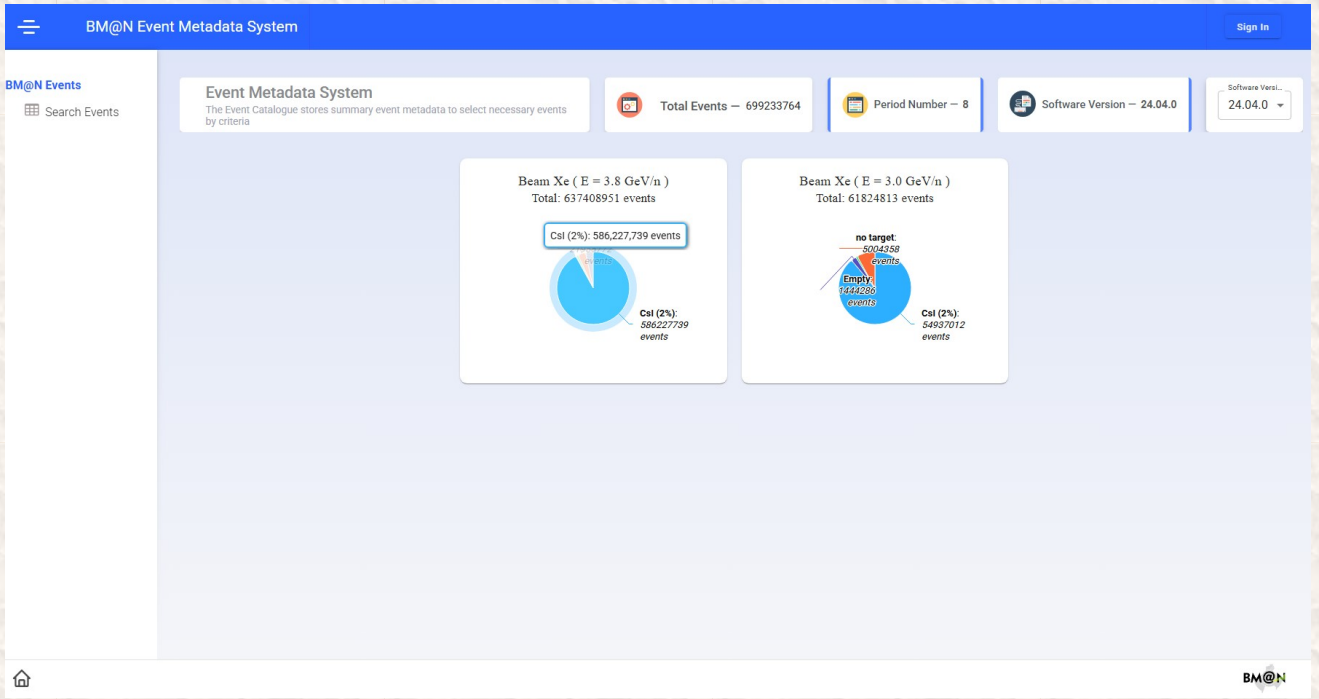
For more details:

E. Alexandrov, I. Alexandrov, A. Chebotov, A. Degtyarev, I. Filozova, K. Gertsenberger,

P. Klimai and A. Yakovlev, "Implementation of the Event Metadata System for physics analysis in the NICA experiments", J. Phys.: Conf. Ser. 2438, 012046 (2023).

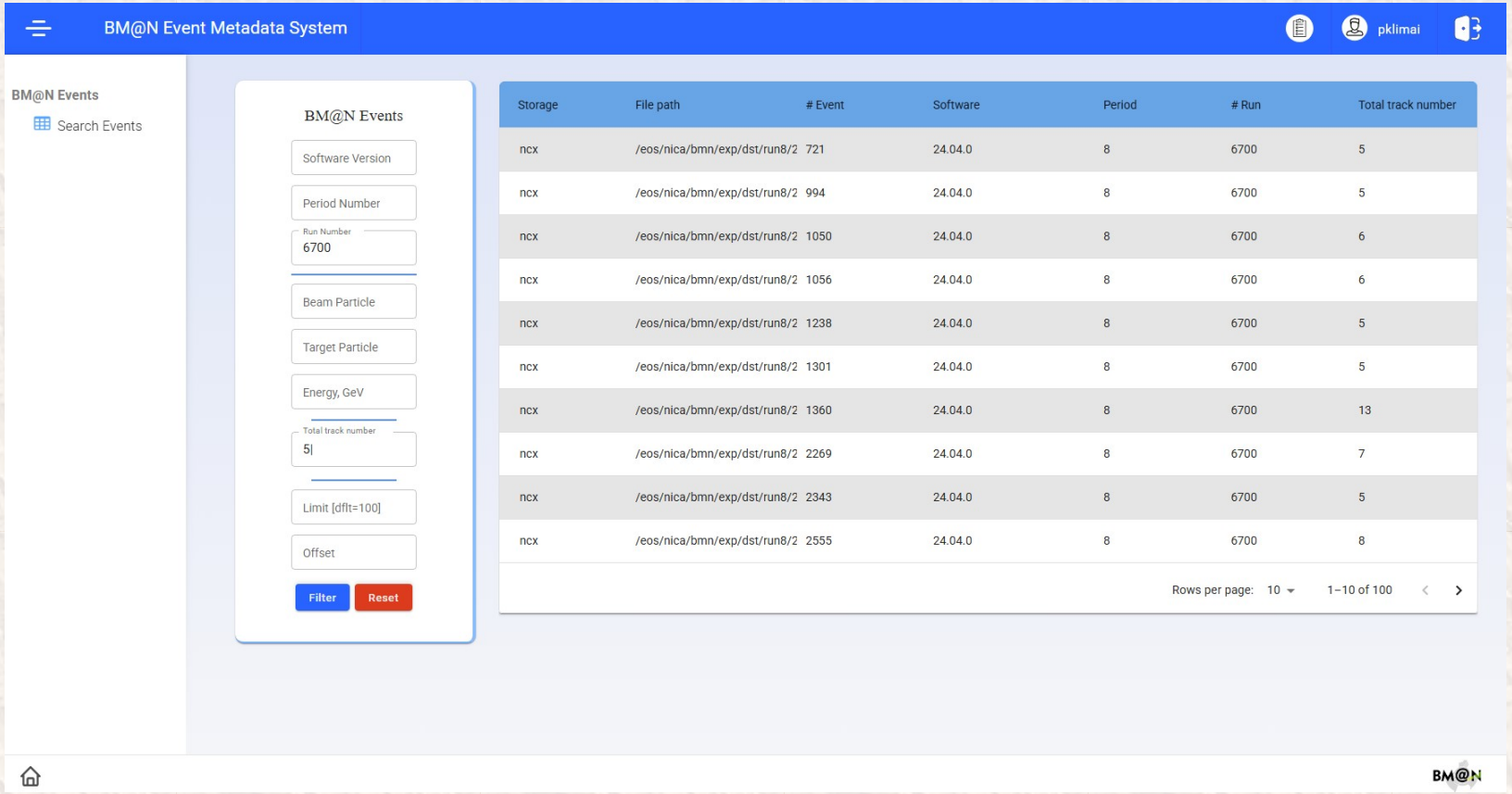
Recent Progress for the EMS

- EMS progress
  - New version deployed at https://bmn-event.jinr.int/ with about 700M records currently in it
  - Performance improved for bulk writing of new events
  - Improved error processing and display
  - Statistics collection script developed and included in the main repository



EMS Filtering Event Example

- Example obtaining events for run 6700, having 5+ tracks
  - Pipe symbol is used for ranges, so filter is to be specified as “5|”



Development of Next-Generation Event Visualization Platform for BM@N (an update)

VisionForge Project Overview

VisionForge – platform for creating next-gen visualization systems

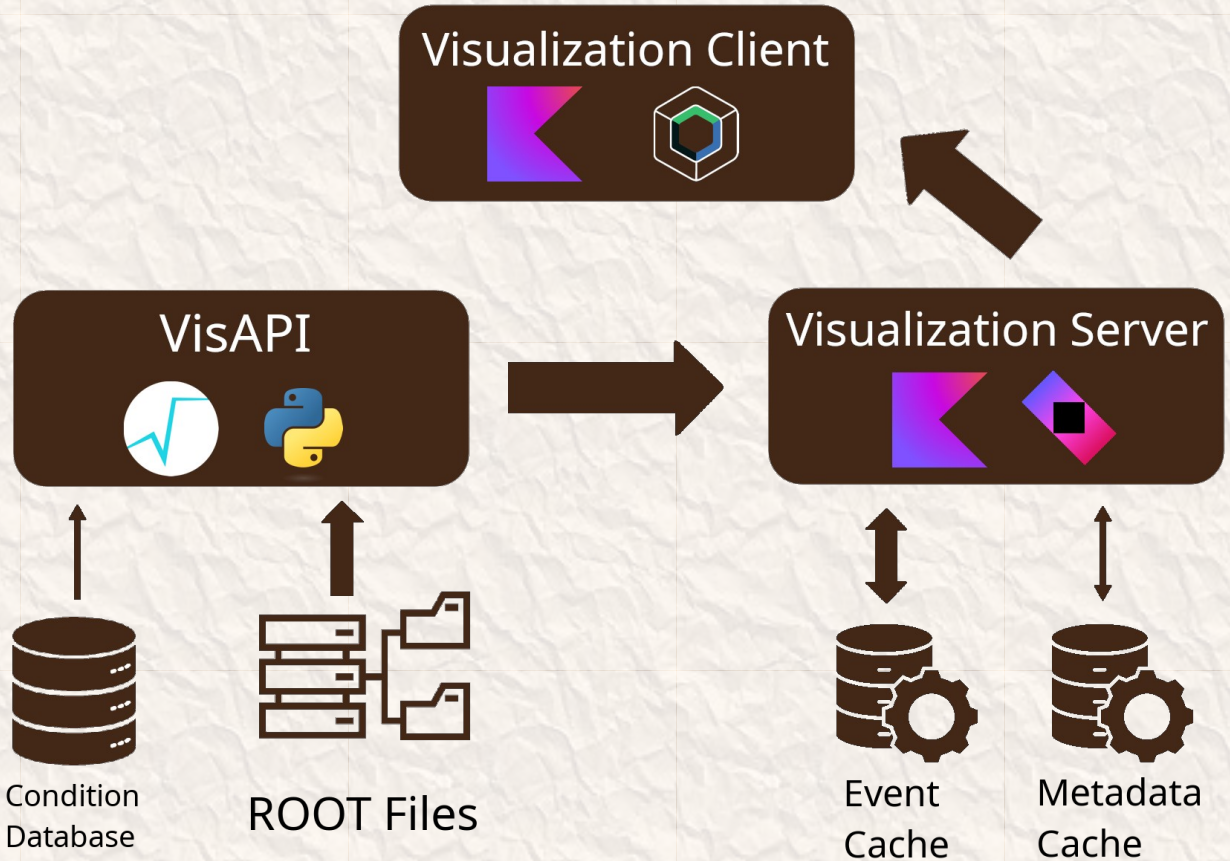
- Distributed dynamic system
  - Visualization model can be created on one node, transferred to another node and rendered there
  - Nodes can exchange **updates** to the model
  - Changing one element or attribute only requires sending this small change
- Performance and optimizations
  - BM@N geometry model includes more than 400 000 elements
  - Geometry can be defined as **prototype** that is used by a set of objects, in this case rendering is simplified – only required properties can be changed if needed

VisionForge Project Overview

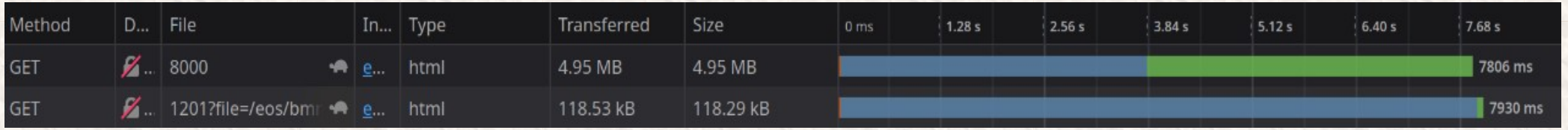
See also:

- VisionForge project
  - sciprog.center/projects/visionforge
- VisionForge source code
  - git.sciprog.center/kscience/visionforge/src/branch/dev/
- Alexander Nozik — Unbearable lightness of data visualization in Kotlin full stack
  - https://www.youtube.com/watch?v=uT5j-xOXC3E&ab_channel=JPoint%2CJoker%D0%B8JUGru

System Architecture



Event latency problem



- Each new event/geometry request takes a significant time to process (approximately 8 seconds in the test setup)
- In case of geometry, the main reason is the JSON format and the size of geometry objects themselves
- Events occupy much fewer space, but they take significantly more time to be extracted from ROOT files.
- Long event load latency is the main distraction for the user looking several sequential events

Event Cache

Simple caching algorithm (LRU with prefetching):

- Check if the requested event has previously been loaded (hit or miss);
  - If hit: return data from the cache;
  - If miss: send a request to the VisAPI, add results to the cache and return the data.
  - For prefetching: check neighbouring events as well and load them in case of misses;
- Disadvantages:
  - Multiple request to a single event produce multiple requests to the VisAPI (when serving several users);
  - With prefetching, multiple requests to sequential events may produce many requests to the VisAPI and block it (even when serving a single user)

Event Cache

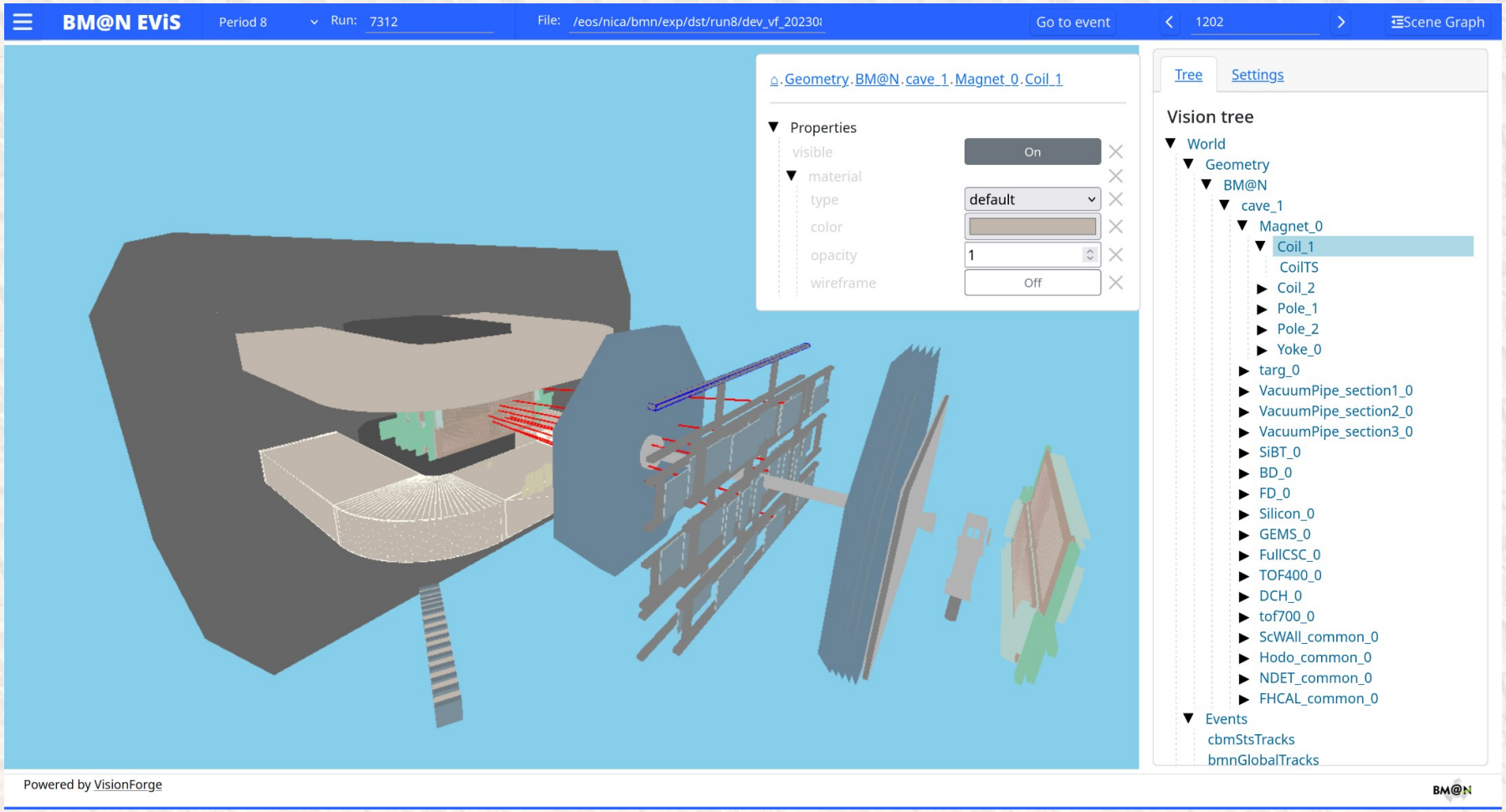
Event service uses an improved LRU cache with prefetching algorithm to reduce sequential event loading latency and improve user experience.

- Check if requested file has been loaded and its metadata are in the metadata cache;
- Atomically:
  - Check if requested event is in the cache;
  - If it is missing, or if the user is close to a non-loaded event, a request to the VisAPI is sent (asynchronously);
  - If the request has been sent, deferred values are added to the cache;
- At this point, a deferred object of the requested event is guaranteed to be present in the cache. Then the event object is awaited and returned.

Available for test now!

- Available online at http://10.220.16.81:8080/
- Example input:
  - Period number: **8**
  - Run number: **8000**
  - File address: **/home/lab/events/mpd_run_Top_8000_ev1_p8.root**
  - Select event: **1, 2, 3,…**
- Possible to build and run it on your own as well (source code available at git.jinr.ru/nica-visualization/evis)

Geometry, tracks, scene graph, tuning



Available Features

- Visualization of geometry of the detector with a detail level of choice.
- Working with the scene: the ability to scale, shift and rotate the scene, section by plane.
- Saving scene configuration (JSON) to a file.
- Ability to show/hide geometric elements, set color, transparency. For a solid detector, we loaded from a prepared scheme (XML or JSON) to replace the default.
- Visualization of particle collision events: display of tracks and hits. The source is a ROOT file ROOT) stored on a server.
- Event objects and detector geometry are presented as a hierarchical tree, with tracks grouped by particle type. When an object is selected in the tree, the object is highlighted, and vice versa, when an object is selected in the view, its properties are opened.

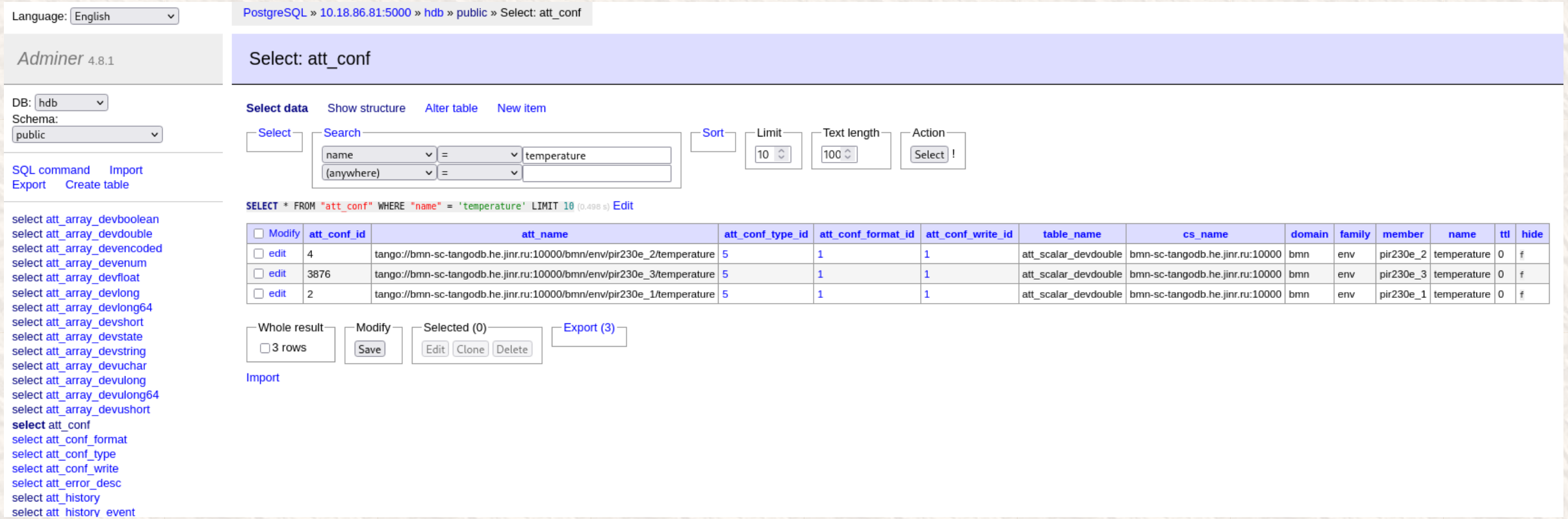
WIP Items

- Desktop application with a web UI based on the current client-server solution.
- Real time visualization soluiton: the present visulization solution using websockets to receive data for visualization and to update the data being visualized by web clients.
- Saving an image, saving scene parameters (without event or geometry data itself) to a file.
- Visualization hits and activated calorimeter towers. The source is either a file (initially ROOT), or a data stream from the socket for online monitoring.
- Selection of event objects with viewing of their properties, editing of color, visibility, marker, size/thickness. Selection/scrolling of transferred events in case of the source from a file.
- Filter of displayed event objects: particles by their code, energy range, only primary tracks. Show/hide separately simulated tracks/particles (before reconstruction), reconstructed tracks/particles
- Show general information: selected setup geometry, event number, number of events (if from file), number of displayed geometry objects.

Development of REST API and Web interfaces for slow control system

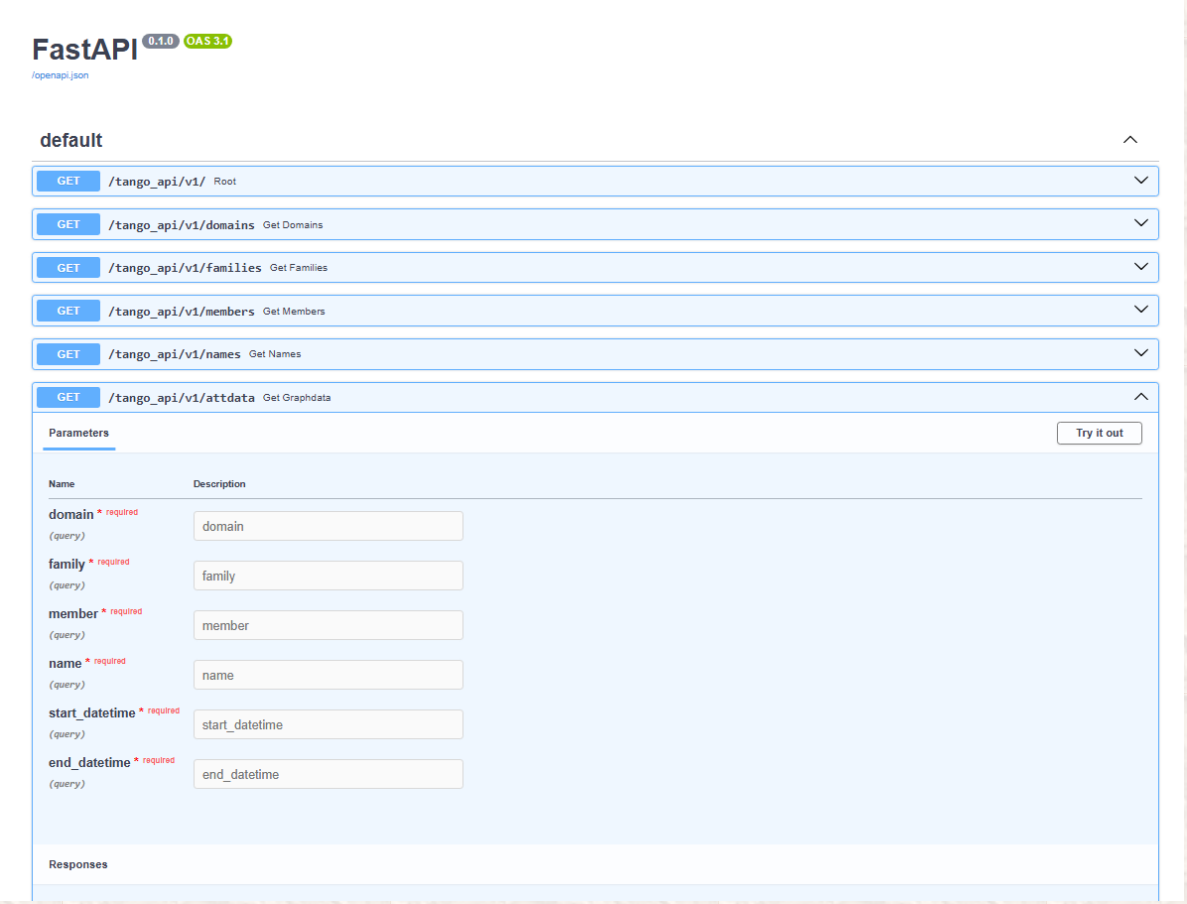
BM@N slow control system database

- Updated version of Tango slow control database uses PostgreSQL
- Convenient REST API and Web access is required



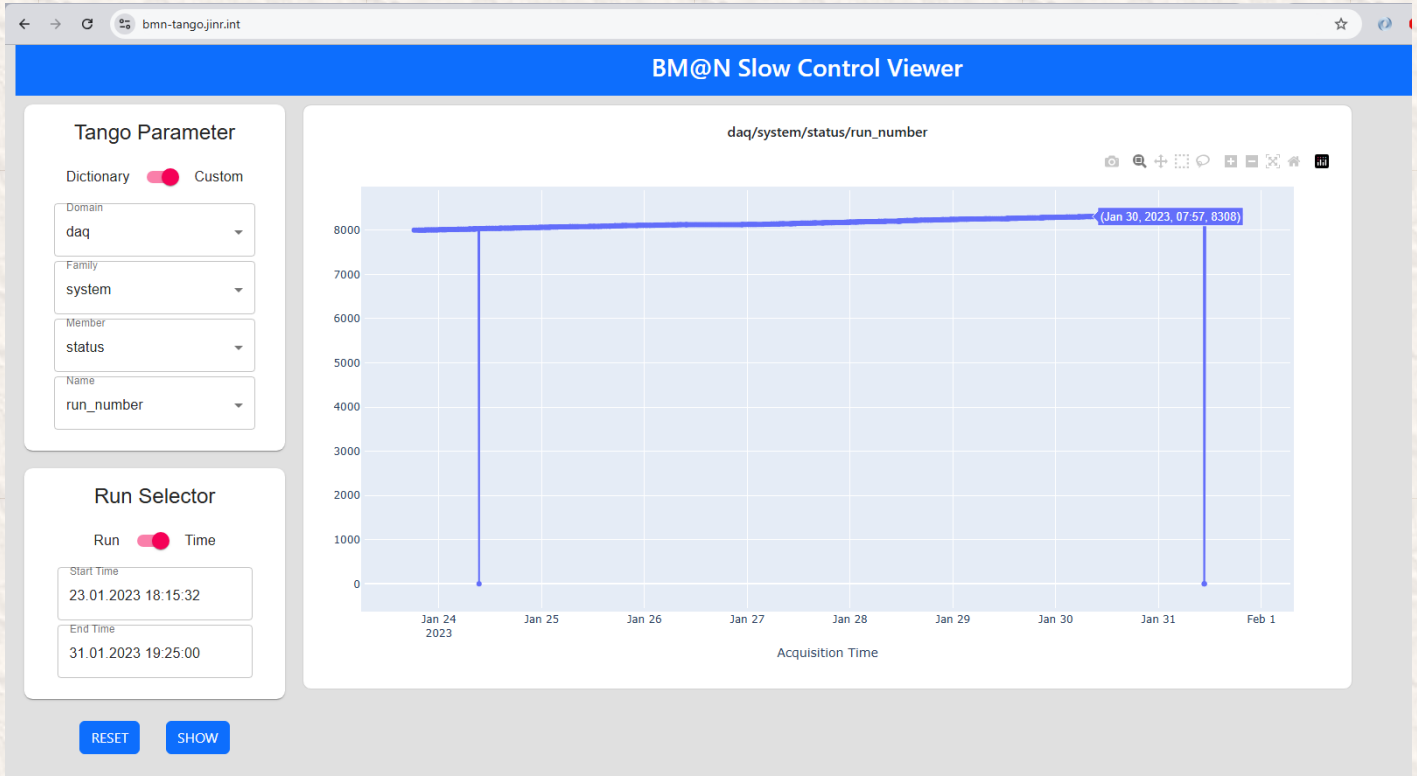
API Service Development

- REST API was developed
  - Using FastAPI framework
  - Deployed in JINR network
  - Used by Web interface (see next slide)
  - Sources available at https://git.jinr.ru/nica_db/tango-rest



Web Interface Development

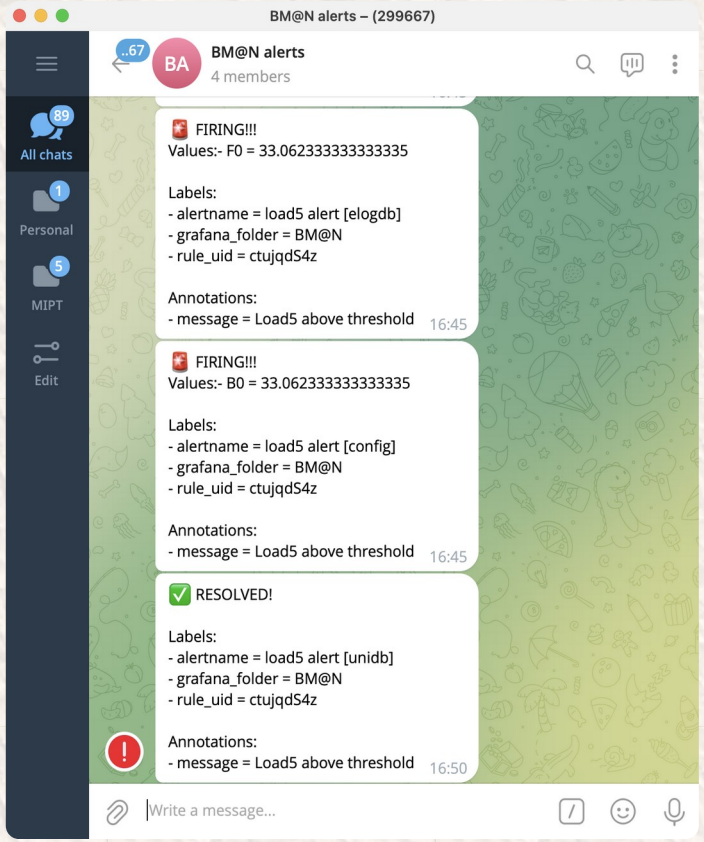
- Web-based viewer for SCS was updated to use the new API and, hence, the new database
- Available at https://bmn-tango.jinr.int/



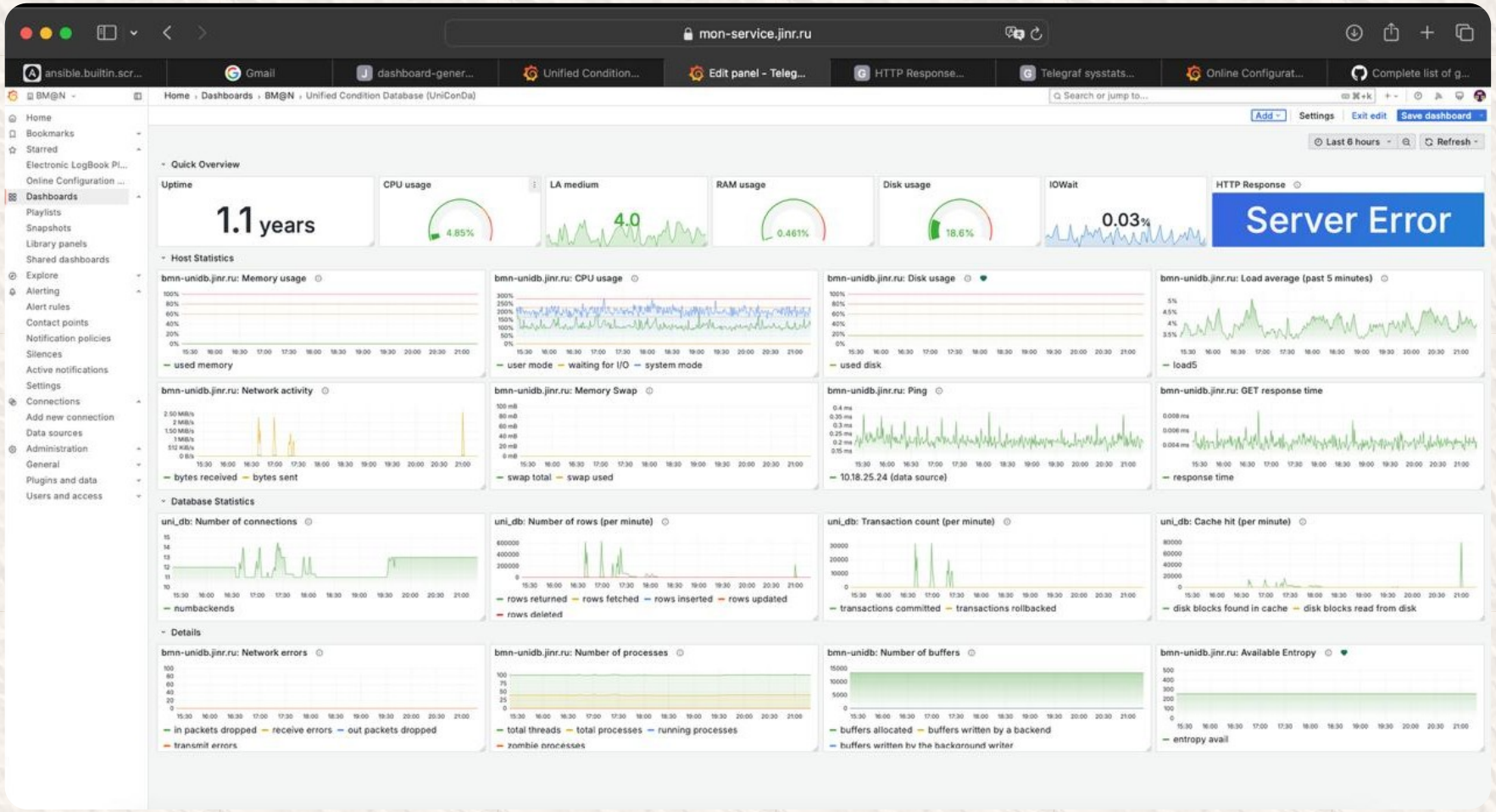
Monitoring Service (an update)

New and Planned Features

- Dasboard UI
  - Configurable Quick overview
  - Chart sorting
  - HTTP response code chart
- Telegram alerts formatting
  - [WIP] Quick alert summary
- Planned features
  - User Manual
  - Gateway Server, Log Collector Server, Keycloak and FreeIPA service healthchecks
  - Event Metadata system and other systems monitoring



New Dashboard UI



Thank You!