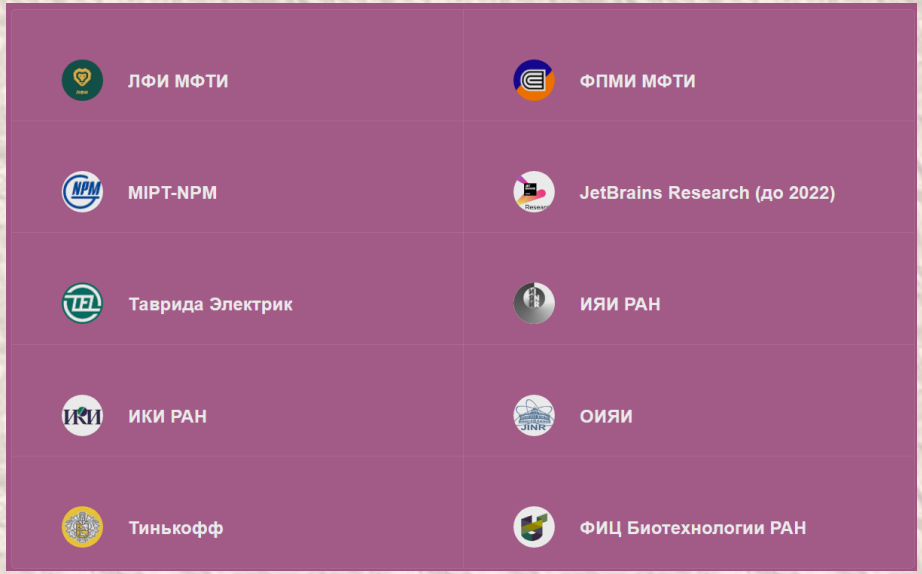
Software contribution from MIPT: Development of software systems and services for BM@N

Peter Klimai

MIPT NPM and SPC

- MIPT Nuclear Physics Methods (NPM) lab
  - Since 2015
  - HEP experiment data analysis
  - Bring modern IT to experimental physics
  - https://npm.mipt.ru/ru/
- Scientific Programming Centre (SPC)
  - Since 2022
  - Education, including Master's program
  - Fundamental and applied research
  - Consulting in field of scientific software systems
  - https://sciprog.center

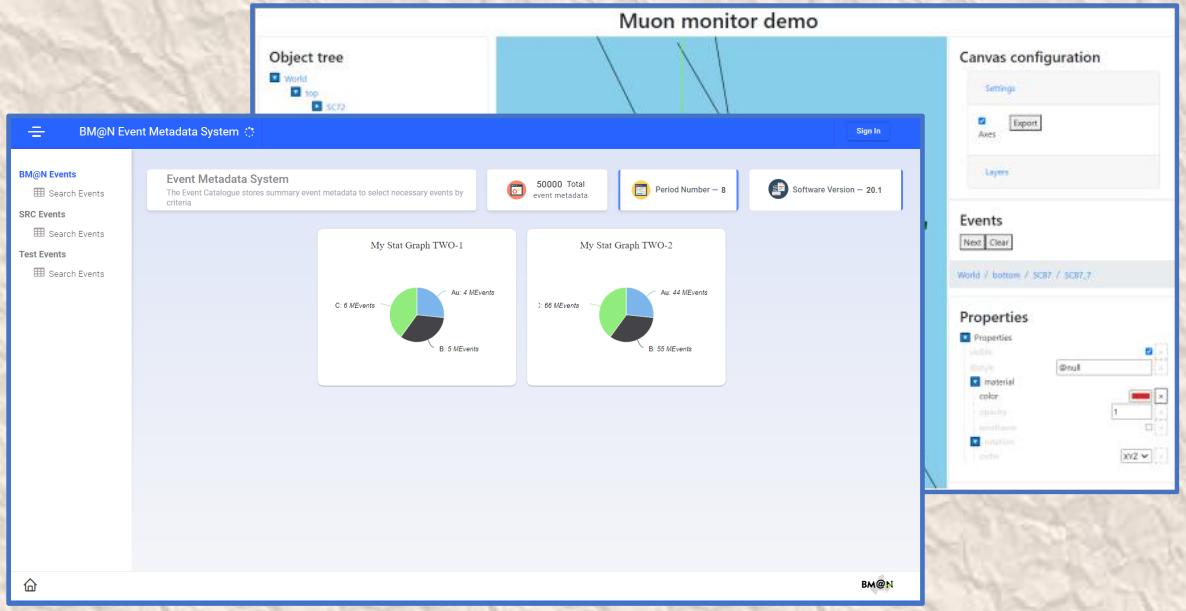


Partners of SPC Master's program "Scientific Software"

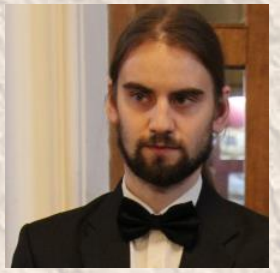
Projects for BM@N

MIPT projects for BM@N include services for:

- Visualization
- Monitoring
- Statistics collection
- Database management

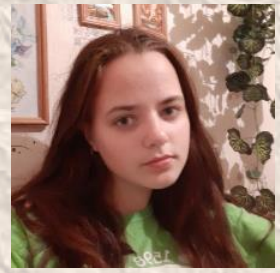


Alexander Nozik



Mihail Zelenyy

Aleksandr Svetlichnyi



Elya Blinova

Artyom Degtyarev

Peter Klimai

Project Summary

Project	URL	Status
NICA scheduler configurator GUI	https://bmn-scheduler.jinr.ru	In production
Monitoring service	https://mon-service.jinr.ru	In production
Slow control system viewer	https://bmn-tango.jinr.ru	In production
Statistics collection script	https://git.jinr.ru/nica/bmnroot/-/tree/dev/services/statistics	Available as part of BmnRoot
Next-generation event display	https://github.com/SciProgCentre/visionforge	ROOT integration WIP
Event Metadata System	https://git.jinr.ru/nica_db/emd	First version deployed
Condition database data migration tool	https://github.com/SciProgCentre/jinr-database-tools	First version delivered

Condition Database Data Migration Tool

By Mihail Zelenyy <mihail.zelenyy@phystech.edu>

Database Data Migration Tool

- Task
  - Transfer data from CSV, XML files to database
  - Data model defined as JSON file
- Implementation
  - Python
  - CLI and Qt-based GUI
  - JSON schema used to check each JSON model file
  - https://github.com/SciProgCentre/jinr-database-tools

CLI Options and config

```
# ./main_cli.py
usage: python /home/lab/jinr-database-tools/main_cli.py [-h] command ...

Parse input data file and load to database

positional arguments:
  command
    validate  Validate JSON file with input data format
    load      Parse input data file and load to database
    format    Open description of the JSON schema for input data
    generate  Generate JSON templates from database

options:
  -h, --help  show this help message and exit

Process finished with exit code 0
```

config.json :

```
{
  "database" : {
    "username" : "user",
    "password" : "user_pass",
    "host" : "192.168.65.52",
    "port": 5001,
    "database": "bmn_db"
  }
}
```

validate option

- Validation is based on JSON schema

detector_.json :

```
{
  "format": "CSV",
  "table": "detector_",
  "columns": [
    {
      "name": "detector_name",
      "type": "string",
      "type_properties" : {
        "length" : 10
      }
    },
    {
      "name": "description",
      "type": "string",
      "type_properties" : {
        "length" : 30
      }
    }
  ]
}
```

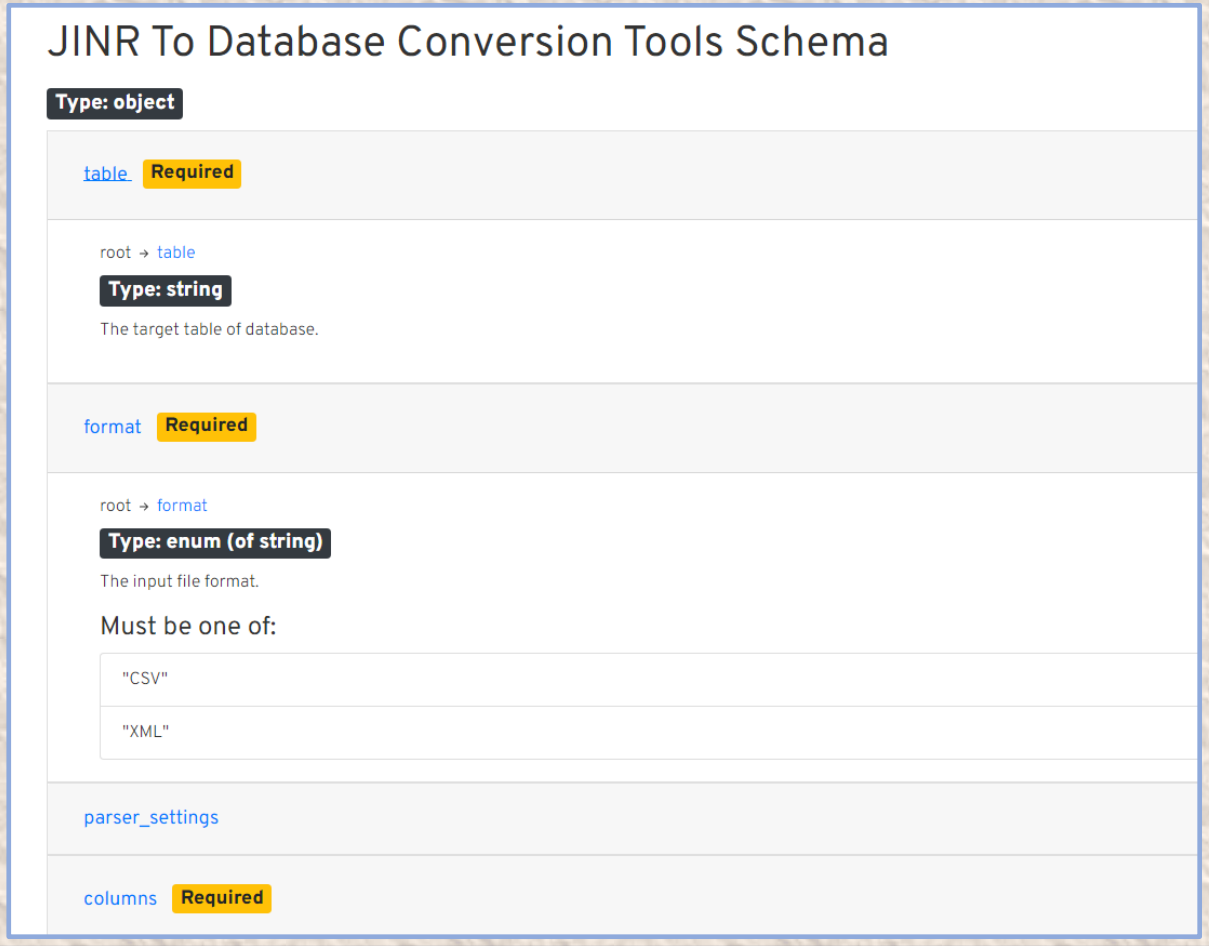
schema.json :

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "title": "JINR To Database Conversion Tools Schema",
  "description": "",
  "type": "object",
  "required": [
    "table",
    "format",
    "columns"
  ],
  "properties": {
    "table": {
      "description": "The target table of database.",
      "type": "string",
      "uniqueItems": true
    },
    "format": {
    ...
```

```
# ./main_cli.py validate tests/data/detector_.json
tests/data/detector_.json successfully validated.
```

`format` option

- Opens HTML description of schema



JINR To Database Conversion Tools Schema

Type: object

table **Required**

root → table

Type: string

The target table of database.

format **Required**

root → format

Type: enum (of string)

The input file format.

Must be one of:

"CSV"

"XML"

parser_settings

columns **Required**

generate option

```
# psql -U user -h 192.168.65.52 -p 5001 -d bmn_db
bmn_db=# \dt
              List of relations
 Schema |         Name         | Type  | Owner
--------+----------------------+-------+-------
 public | detector_            | table | user
 public | detector_parameter   | table | user
 public | parameter_           | table | user
 public | run_                 | table | user
 public | run_geometry         | table | user
 public | run_period           | table | user
 public | simulation_file      | table | user
…
bmn_db=# \q


# ./main_cli.py generate
Generated file simulation_file.json for table simulation_file
Generated file detector_parameter.json for table detector_parameter
Generated file detector_.json for table detector_
Generated file run_.json for table run_
Generated file run_geometry.json for table run_geometry
Generated file run_period.json for table run_period
Generated file parameter_.json for table parameter_
…
```

Resulting run_.json :


```
{
  "format": "CSV",
  "table": "run_",
  "columns": [
    {
      "name": "period_number",
      "type": "integer"
    },
    {
      "name": "run_number",
      "type": "integer"
    },
    {
      "name": "file_path",
      "type": "string"
    },
    {
      "name": "beam_particle",
      "type": "string"
    },
    {
      "name": "target_particle",
      "type": "string"
    },
    {
      "name": "energy",
      "type": "float"
    },
…
    {
      "name": "start_datetime",
      "type": "datetime",
      "type_properties": {
        "datetime_flavour": "iso"
      }
    },
    {
      "name": "end_datetime",
      "type": "datetime",
      "type_properties": {
        "datetime_flavour": "iso"
      }
    },
    {
      "name": "event_count",
      "type": "integer"
    },
    {
      "name": "field_voltage",
      "type": "float"
    },
    {
      "name": "file_size",
      "type": "float"
    },
    {
      "name": "geometry_id",
      "type": "integer"
    },
    {
      "name": "file_md5",
      "type": "string"
    }
  ]
}
```

load option

- Load test data example:

detector_.json :

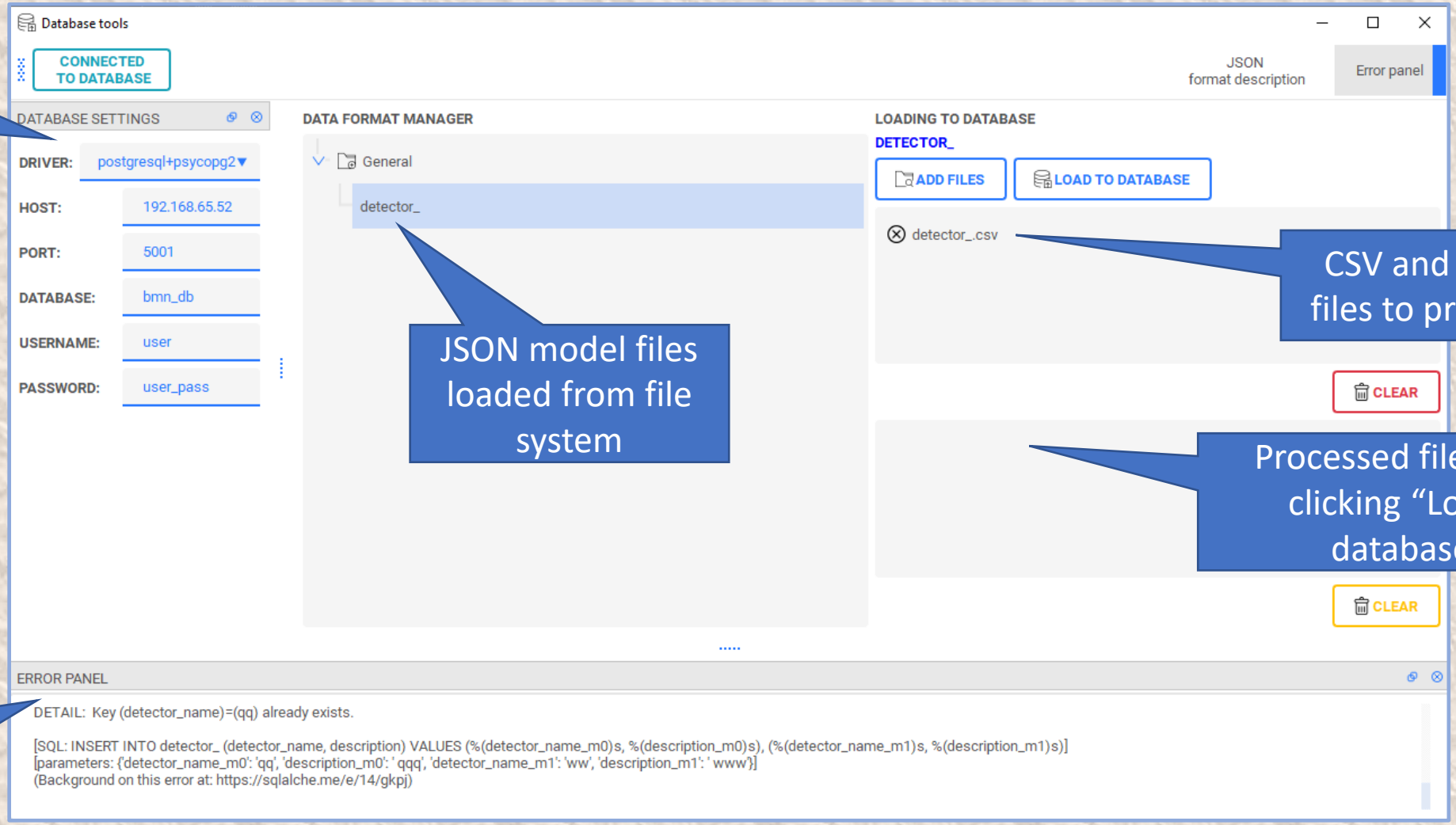
```
{
  "format": "CSV",
  "table": "detector_",
  "columns": [
    {
      "name": "detector_name",
      "type": "string",
      "type_properties" : {
        "length" : 10
      }
    },
    {
      "name": "description",
      "type": "string",
      "type_properties" : {
        "length" : 30
      }
    }
  ]
}
```

detector_.csv :

```
'0','0'
'1','1'
'2','2'
'3','3'
'4','4'
'5','5'
'6','6'
'7','7'
'8','8'
'9','9'
```

```
# ./main_cli.py load -s tests/data/detector_.json tests/data/detector_.csv
Loading tests\data\detector_.csv: SUCCESS…
```

GUI Example View



Database settings

JSON model files loaded from file system

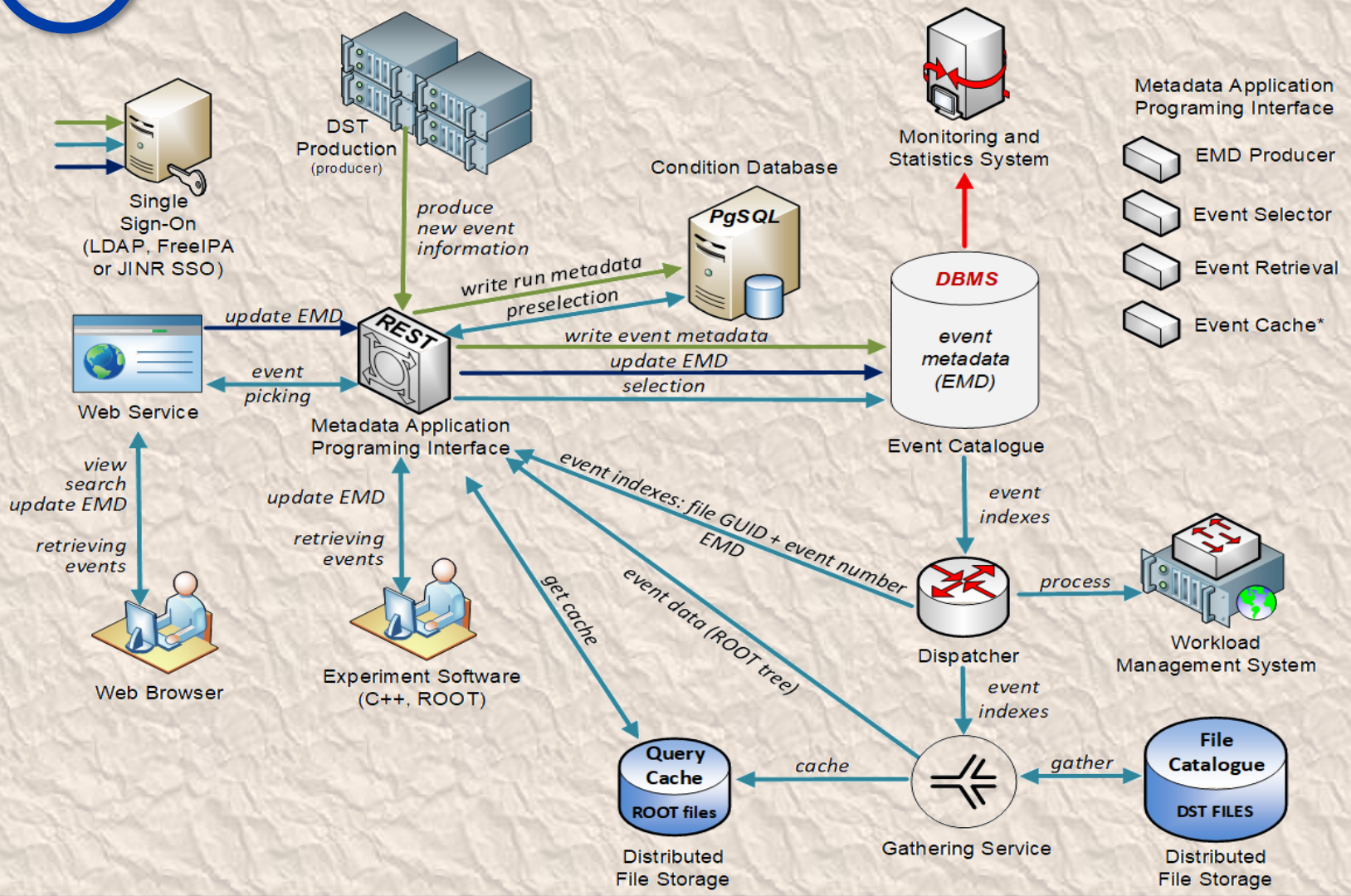
CSV and XML files to process

Processed files (after clicking “Load to database”)

Error log

Event Metadata System - Update

EMS Architecture

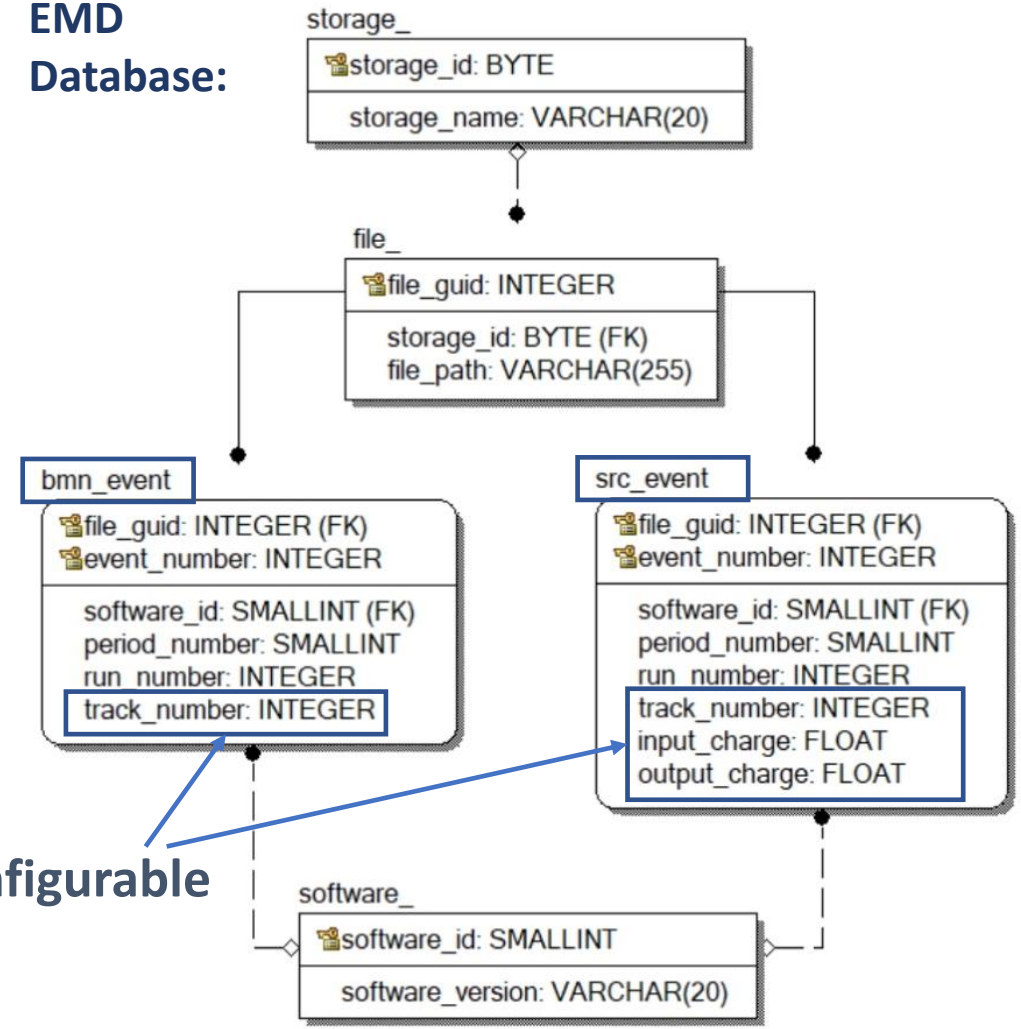


For more details:

E. Alexandrov, I. Alexandrov, A.Degtyarev, K. Gertsenberger, I.Filozova, P. Klimai, A. Nozik and A. Yakovlev, “Design of the Event Metadata System for the Experiments at NICA”, Phys. Part. Nuclei Lett. 18, 603–616 (2021).

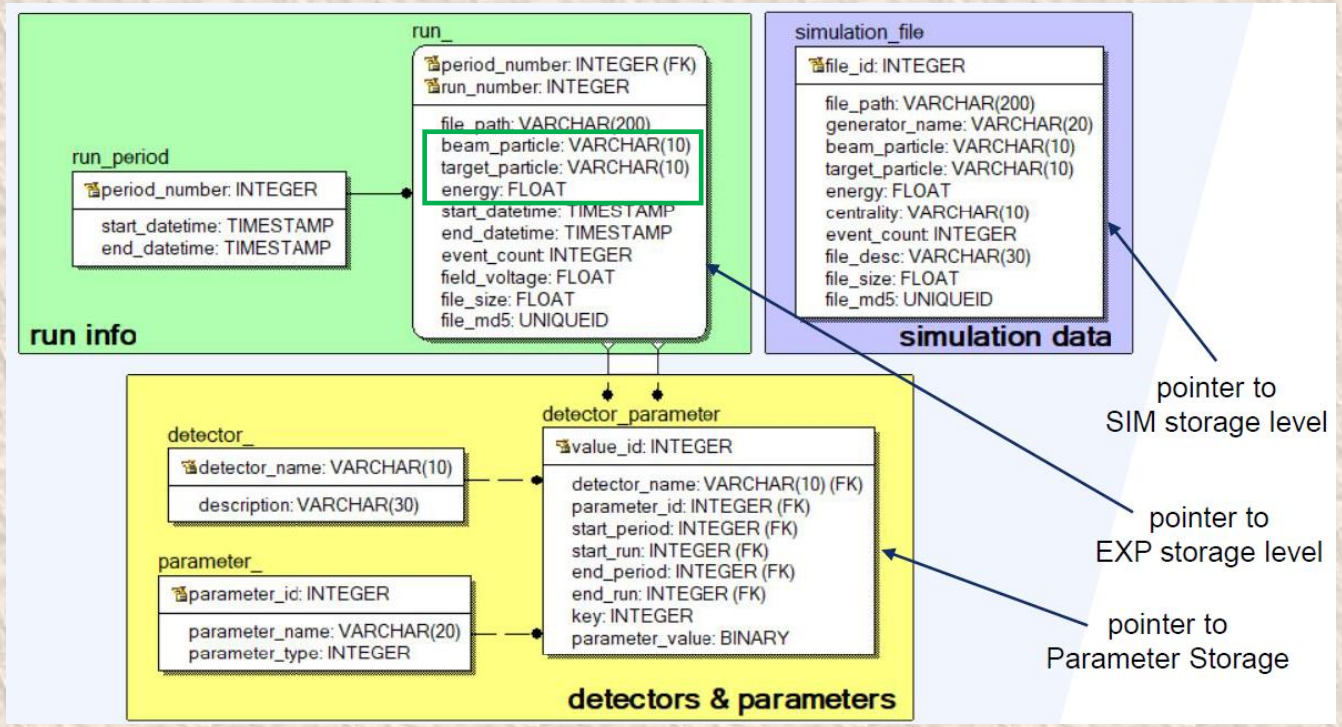
Current BM@N Database Schema

EMD Database:



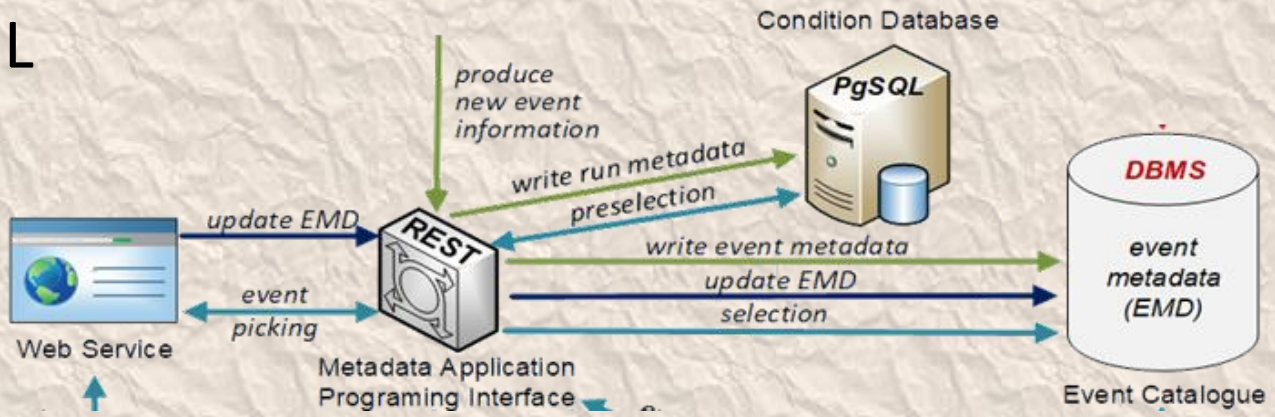
Configurable

Condition Database:

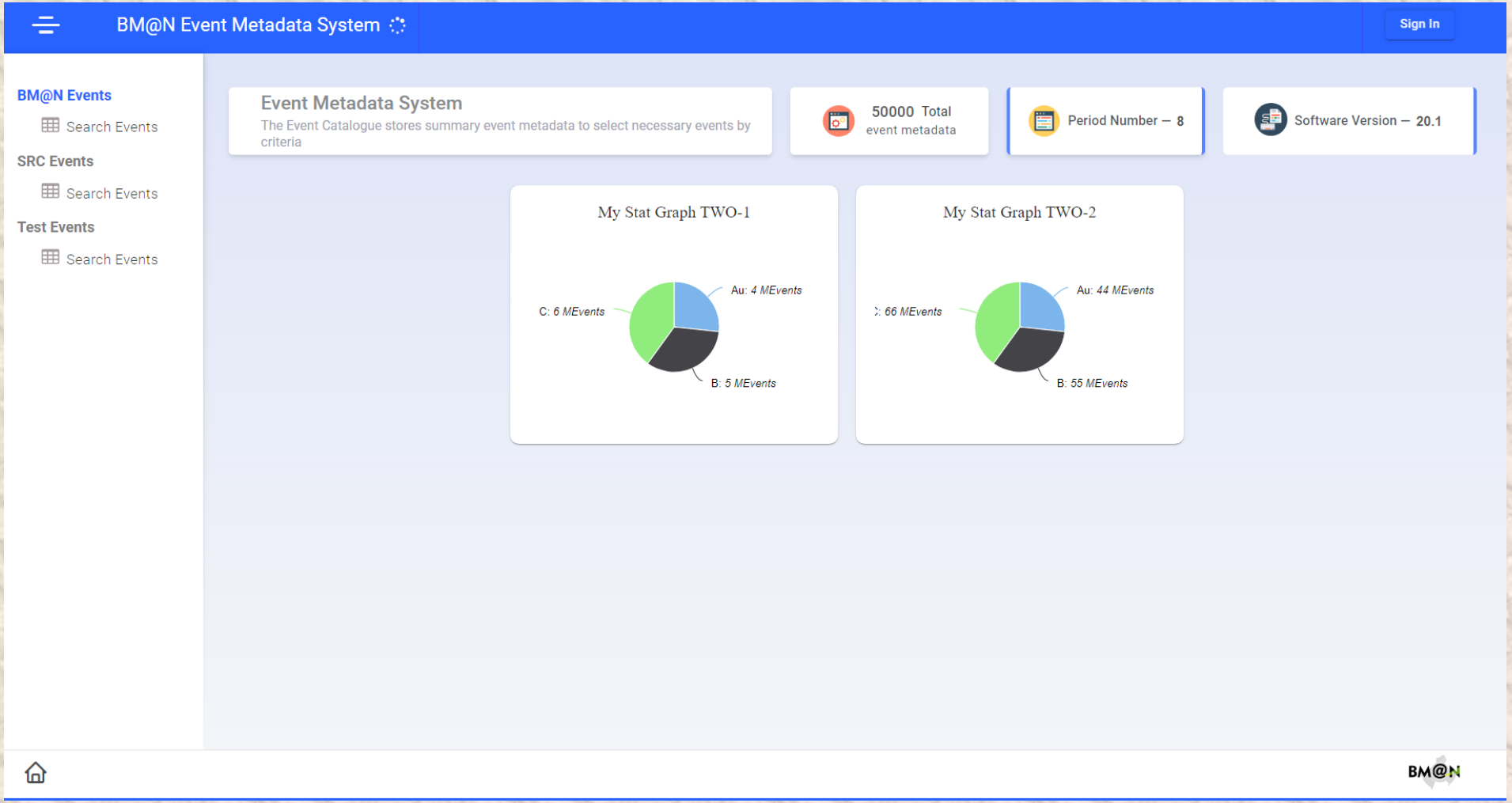


REST API and Web UI Implementation

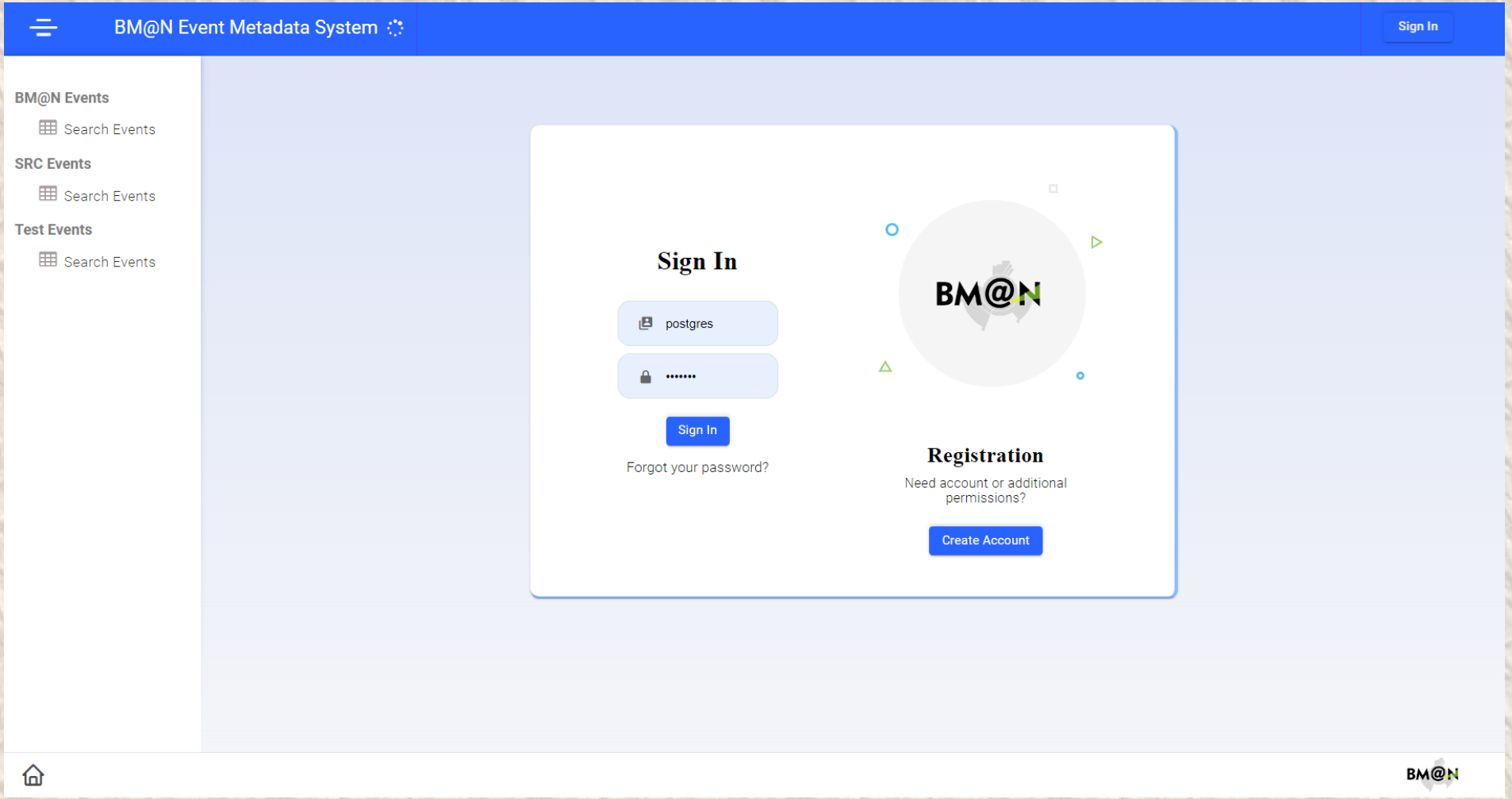
- Using Kotlin programming language
  - Multiplatform
  - JVM runtime back end
  - ktor framework for REST API
  - React front end
- Packed in Docker
- Configuration file provided in YAML



Web UI Main Page



Sign-In


BM@N Event Metadata System
Sign In
BM@N Events
Search Events
SRC Events
Search Events
Test Events
Search Events
Sign In
postgres
Sign In
Forgot your password?
BM@N
Registration
Need account or additional permissions?
Create Account
BM@N

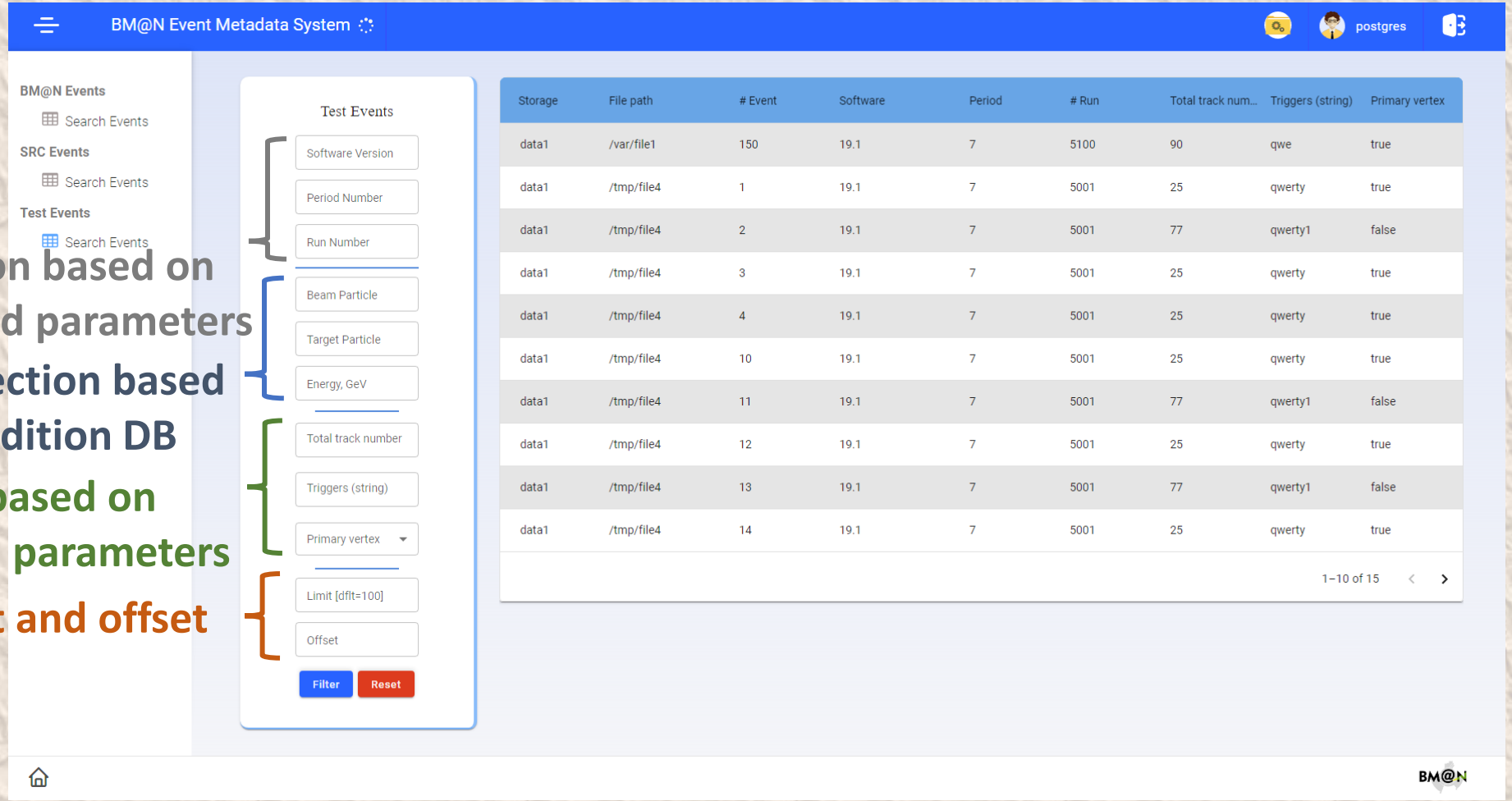
Main search page

Selection based on standard parameters

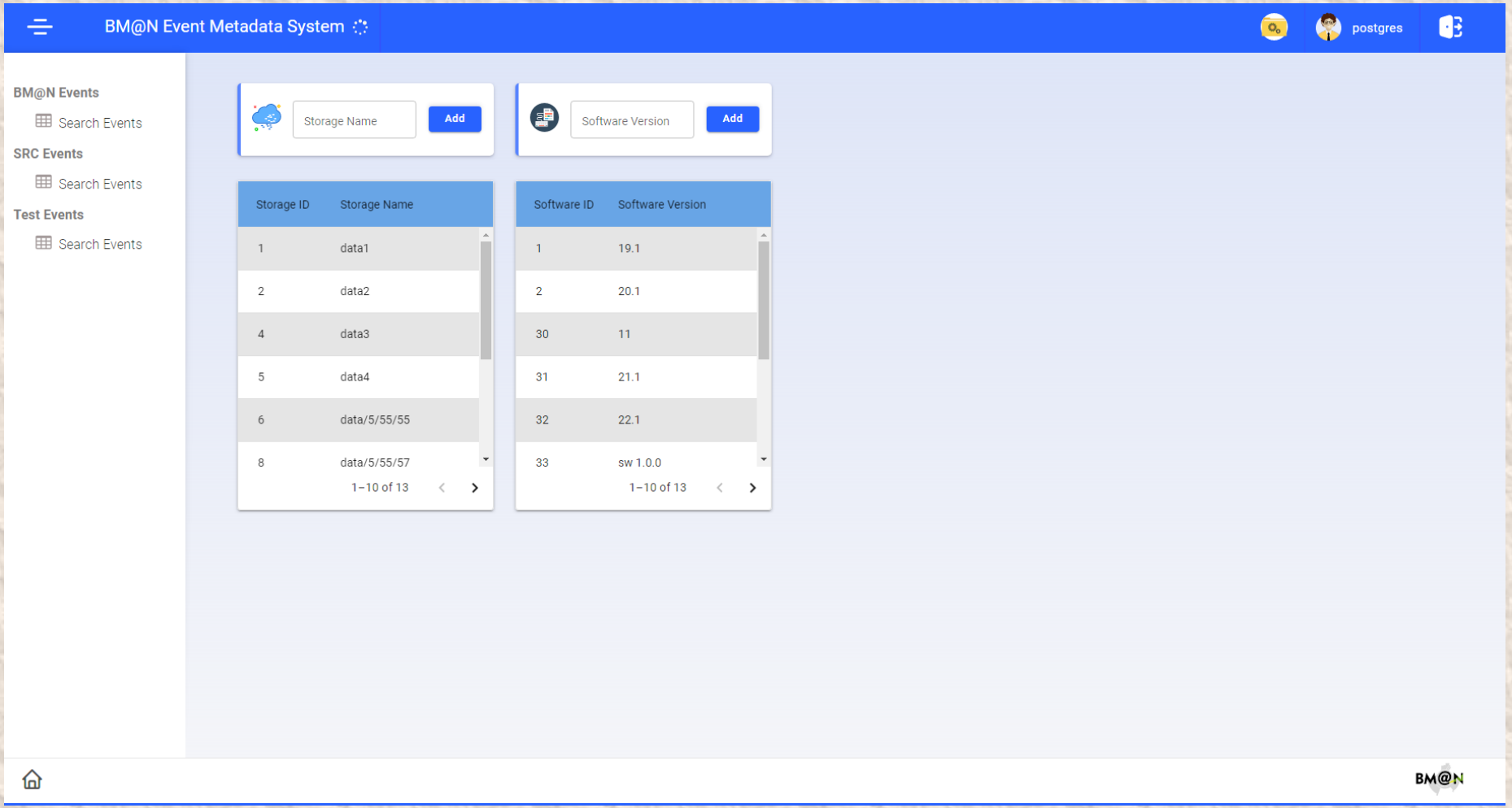
Preselection based on Condition DB

Selection based on configured parameters

Limit and offset



Dictionaries



EMS Current Status

- Event Metadata System current status
  - EMS Database based on PostgreSQL is deployed
  - Integrates with BM@N Condition database
  - REST API and Web UI
  - Macro to write BM@N events in the catalogue
  - Role-based access control implemented
  - Monitoring
- Work in progress
  - Fill the event catalogue with actual BM@N events
  - HA and Backup
  - Automated Deployment

Thank You!