

JOINT INSTITUTE FOR NUCLEAR RESEARCH

[Dzhelepov](https://web.archive.org/web/20031204011713/http:/dzhelepov.jinr.ru/) laboratory of Nuclear Problems

**FINAL REPORT ON THE**

**START PROGRAMME**

*SPD Hardware Database*

**Supervisor:**

Khramov Evgeny Vladimirovich

**Student:**

Gussalov Soslan Russia  
North Ossetian State University

**Participation period:**

July 22 – September 14,

Summer Session 2024

Dubna, 2024

TABLE OF CONTENTS

Abstract…………………………………………………..………………………3

Introduction……………………………………………………………………...4

SPD information ecosystem……………………………………………………..5

Languages and frameworks……………………………………………………...8

Rest API service………………………………………………………………….9

Mapping………………………………………………………………………...12

Service Design………………………………………………………………….13

Visualization of connections between devices………………………………….14

Conclusion……………………………………………………………………...15

Bibliography……………………………………………………………………16

Acknowledgments……………………………………………………………...17

АBSTRACT

Spin Physics Detector (SPD) is a experiment at the NICA accelerator complex at JINR, designed for studies in spin physics with a polarized beams. It will be composed of number of subsystem, using various types of detectors: drift chambers and tubes, MRPCs, Cherenkov counters, scintillation and silicon detectors. Signals from the detectors will be collected by the data acquisition system having few hundred thousand channels. Every component of the detector and DAQ systems will have its sets of parameters and configuration settings that have to be kept for use in operating and maintenance of these systems and especially helpful in knowledge transfer between team members. Some of this data must be also used for the processing and analysis of data collected at the the facility.

A system is being developed to store information about connection between components and to make it available to users and automatic systems. It will be usefu to ensure correct identification of the origin of signals from detector components.

NTRODUCTION

The SPD experiment is a research project aimed at studying the physics of elementary particles and searching for new fundamental knowledge about the interactions and structure of particles. The main goal of this project is to analyze the data obtained from the experimental SPD detector installed at the NICA particle accelerator, created at the Joint Institute for Nuclear Research, located in the city of Dubna, Moscow region.

The SPD installation is a large-scale system of detectors capable of registering and analyzing various events that occur during collisions of polarized deutron and protons with high energy.

To store links between equipment components and provide access to them both for personnel and automatic systems, it is necessary to create complex information systems (IS). One of them should be Hardware Mapping - a service for storing and providing information about the interconnection of equipment components and their links with other components.

SPD INFORMATION ECOSYSTEM

There are various databases and information systems that will be used for the SPD experiment:

• Database of event-level metadata (Event Index);

• Databases of data-taking conditions and calibrations;

• Distributed computing and data storage management systems;

• Database of physical metadata;

• Monitoring systems;

• Logging and accounting systems;

These information and computing systems are in the various stages of development, mostly in the very beginning. The design of the Data Acquisition System is yet in the development stage, and the component selection is still in progress, so due to the lack of real data, the Mapping Database is being developed on the basis of test generated components. It can be considered as a part of a Hardware Database – a catalog of components of the SPD detector system.

The Hardware Database SPD is being developed as an integrated information system that should provide:

• Obtaining information about the parameters of hardware components and their relationship with other components;

• Transmission of this information and recording in databases;

• Access to information for data processing and analysis programmes through APIs and applications;

• Access to information for users through interactive and asynchronous interfaces.

• Authorization in the system by means of JINR SSO and granting access rights;

Figure 1 shows a preliminary diagram of the Hardware Database architecture.

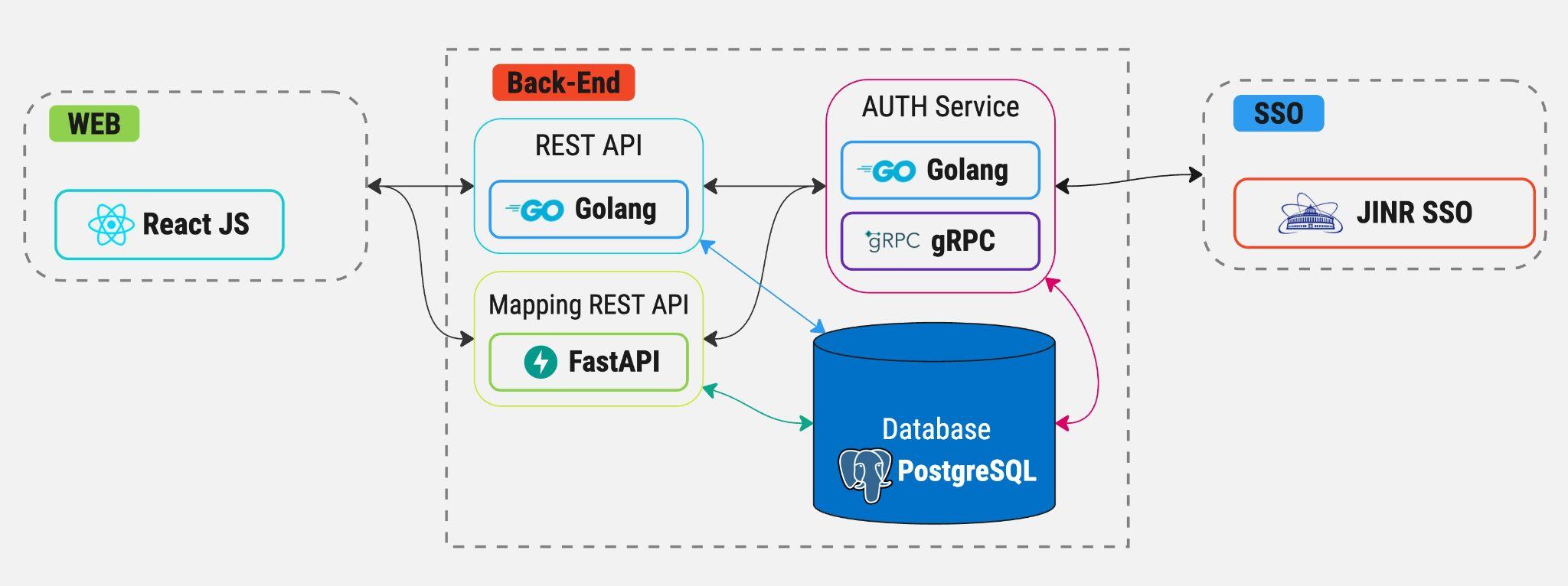


Figure 1: SPD Hardware Database architecture and schema

The Mapping Database shows the connection between the hardware components involved in data acquisition and processing, and the parameters of these components should be accessed from it. And from the other side, access for the mapping information for the specific device will be implemented for such components from their description in the Hardware database

The PostgreSQL database was chosen for reliable storage and processing of structured data. It is widely used for information management in various applications, from small websites to large corporate systems. PostgreSQL supports SQL standards and provides a rich set of functions for working with data, including support for complex queries, transactions, indexes, views and stored procedures. One of the key features of PostgreSQL is its ability to process large amounts of data and support multithreading, which makes it an excellent choice for applications with high performance and reliability requirements.

The frontend part of the client interface was developed using the React JS framework. It provides tools for creating modern dynamic user interfaces, as well as provides effective interaction with the server and data manipulation. An React JS application is built from a set of components, each one representing a specific part of the user interface. Components can be nested into each other, forming a hierarchy. Each component includes an HTML template for displaying the user interface, TypeScript code for logic and data structure, as well as CSS styles for appearance.

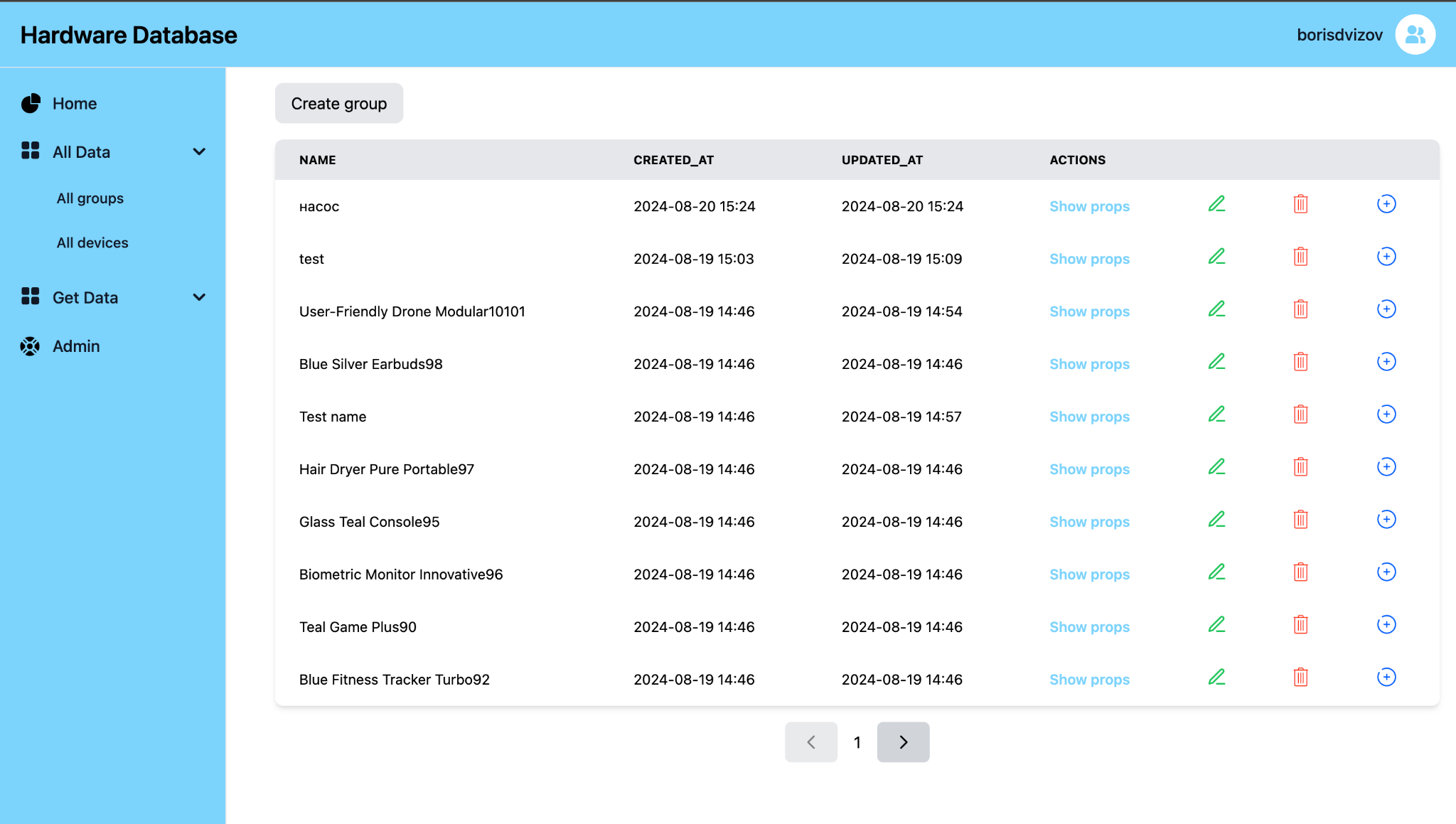


Fig. 2: Web interface

LANGUAGES AND FRAMEWORKS

The architecture of the project is built on the basis of microservices, which allows to divide the logic of different tasks into separate components that communicate with each other. This approach allows to increase the level of availability and fault-tolerance - in case of failure or breakdown of one microservice, the rest continue their work, as this approach allows to easily scale the system and increase the flexibility of development.

The FastAPI framework written in Python was used to develop the Mapping service. FastAPI is a lightweight asynchronous framework for Python, which is mainly used for developing API services. The main advantages of the framework are:

• Speed of operation. FastAPI beats all Python frameworks in terms of performance.

• Asynchrony. FastAPI uses ASGI servers by default, when in the same Django you have to deal with the configuration of the application from WSGI to ASGI, which takes enough time. Flask, unfortunately, does not support asynchrony and works only under WSGI

• Built-in data validation

REST API SERVICE

This service is designed to store and retrieve data on parameters of equipment components and to sort components by groups. The service operation is divided into several stages:

• Receiving HTTP requests via REST API and returning data requested by the user.

Each HTTP request is accompanied by middleware, which is responsible for receiving the JWT token from the request and sending it to the authorization microservice to provide information about the user and his access rights. If the user's permissions are not sufficient to access the requested resource, the request returns an error. If the request successfully passes the validation, it proceeds. Currently, we have implemented HTTP requests for working with groups,

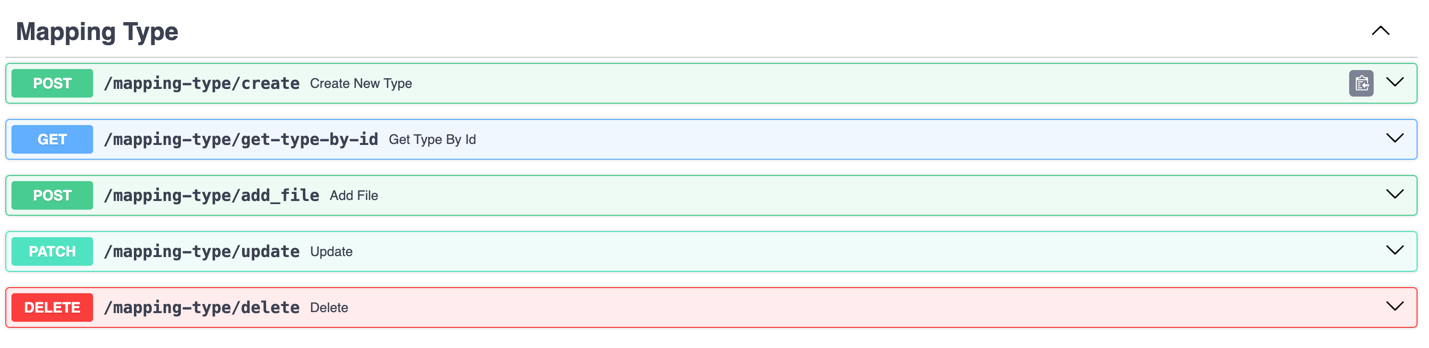


Fig: 3 Mapping type requests.

At the moment five requests related to groups of installation components are implemented in REST API:

• Create New Type - create a new group for components that includes the name of the group and the number of child components that can be connected to the component inherited from this group.

• Get Type by unique identifier (ID) - get the group by unique identifier that is assigned automatically when the group is created.

• Add File - method of creating several groups from JSON file, the method completely copies the implementation of the method create new type, except that in the method add file is created not one group, but several.

• Update - method to update the group, in this method you can update the name and number of child components that can be connected to the component inherited from this group.

• Delete - method for deleting a group



Fig: 4 Mapping component requests.

• Create new device - Creates a new device based on a previously created group. The device includes the following parameters:

• name – device name

• type - device typeа

• ID - unique identifier, which is automatically generated during device creation (the unique identifier is not specified in the request for device creation, as it is assigned automatically).

• ports – When creating, an array of strings that imply port names is specified, when creating a device, ports are written to a separate table in the database and have the following fields:

• device ID - unique identifier of the device to which this port belongs

• name – port name

• child ID – unique identifier of the device connected to this port

• ID – unique port identifier

• updateted at - port update time

• created at - port establishment time

• Get device by ID – obtaining a device by unique identifier (the unique identifier is generated automatically when creating a plant component)

• Get by hardware ID - retrieval by unique identifier of the installation component

• Adding intermediate device - adding an intermediate device component between two other components

• Add file - method of creating multiple devices from JSON file, the method completely copies the implementation of create new device method, except that in add file method several devices are created instead of one.

• Update - method for updating devices, in this method you can update the name that can be connected to the component inherited from this group.

• Delete - method for removing devices

• Get device parent – parent component retrieval

• Get device child – child acquisition

MAPPING

Mapping database architecture consists of 4 tables:

• Alembic version - A table responsible for the version of database migrations, which is generated using the Alembic (Python) library.

• Mapping type - Table storing information about component groups.

• Mapping device - Table storing information about components and their parent components.

• Mapping port - Table storing information about component ports and components connected to the ports.

Figure 5 shows a visual representation of the database architecture.

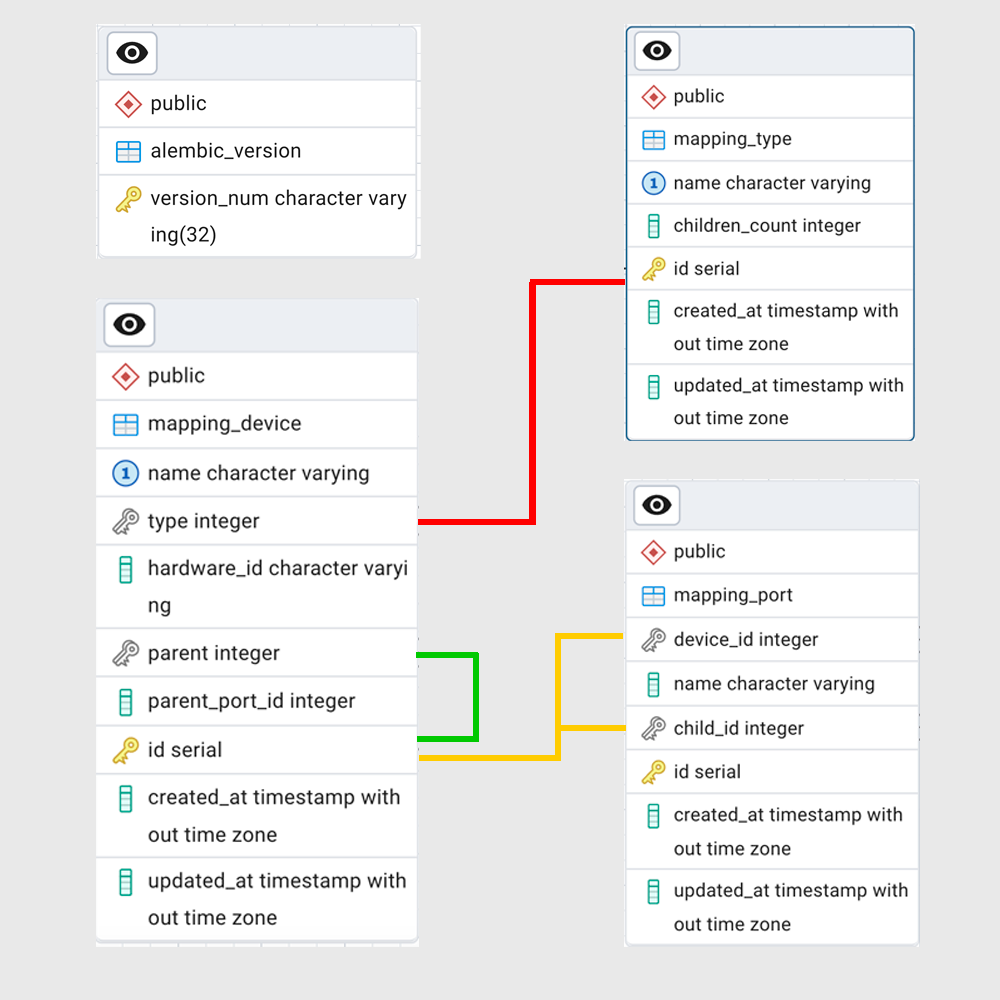


Fig: 5 Database Architecture.

SERVICE DESIGN

Unit of Work design pattern was used for service development. Within one endpoint it is often necessary to perform several queries to the database and there is a problem of data integrity in case one of the queries worked incorrectly. This problem is solved by this pattern, it provides an atomic transaction to the database - this means that in case of an error at the stage of one of the queries, all changes made to the other queries will be canceled. In addition, the logic of data processing and database queries was separated, all logical operations with data occur inside services, and all database queries are performed inside SQLAlchemy repository, this approach allows you to write all necessary database queries once and use them for all its models. Figure 6 shows a visual representation of the pattern Unit of Work.

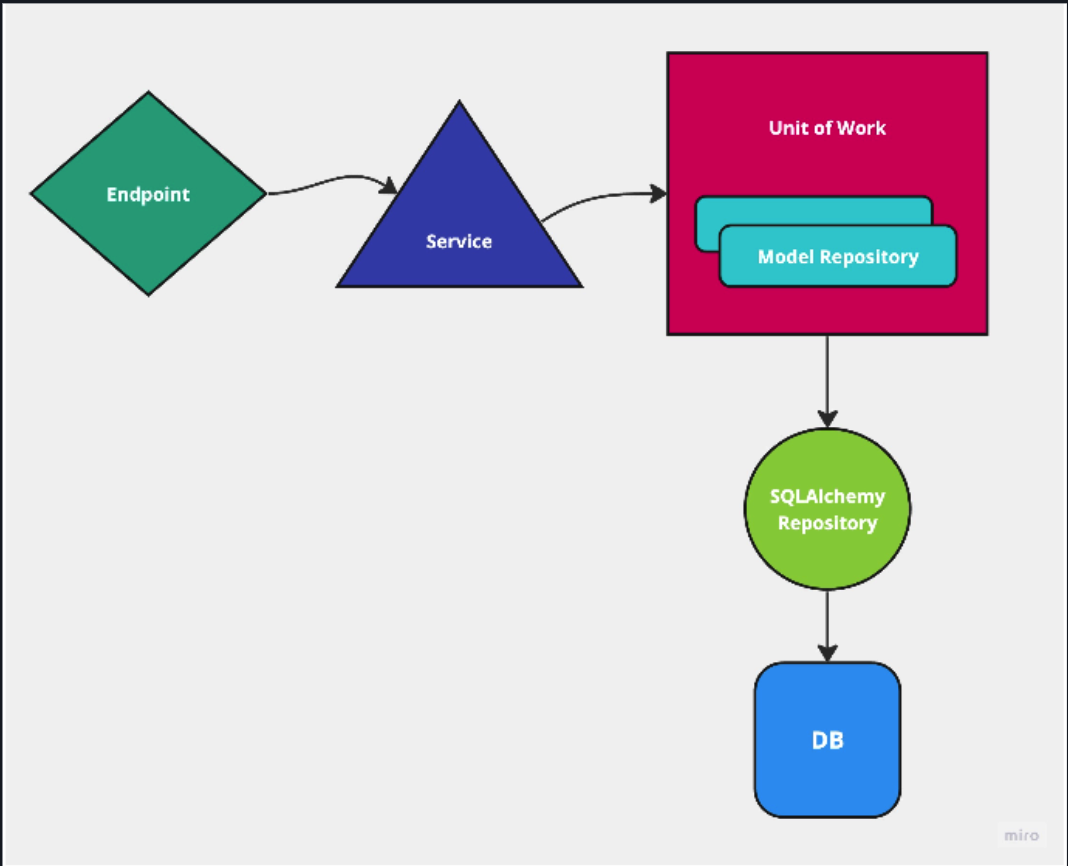


Fig: 6 Visual representation of the pattern Unit of Work.

VISUALISATION OF CONNECTIONS BETWEEN DEVICES

To visualize the connections between the components of the installation, a Web Interface in React JS language was developed, which receives data from the Mapping server via HTTP requests. The React Flow library is used to display components and their interconnections. To control the states of the Web Interface, the Jotai library is used, which is a state manager for React JS applications. Axios library is used to execute HTTP requests. Figure 7 shows the Web interface and an example of component mapping with their interrelationships.

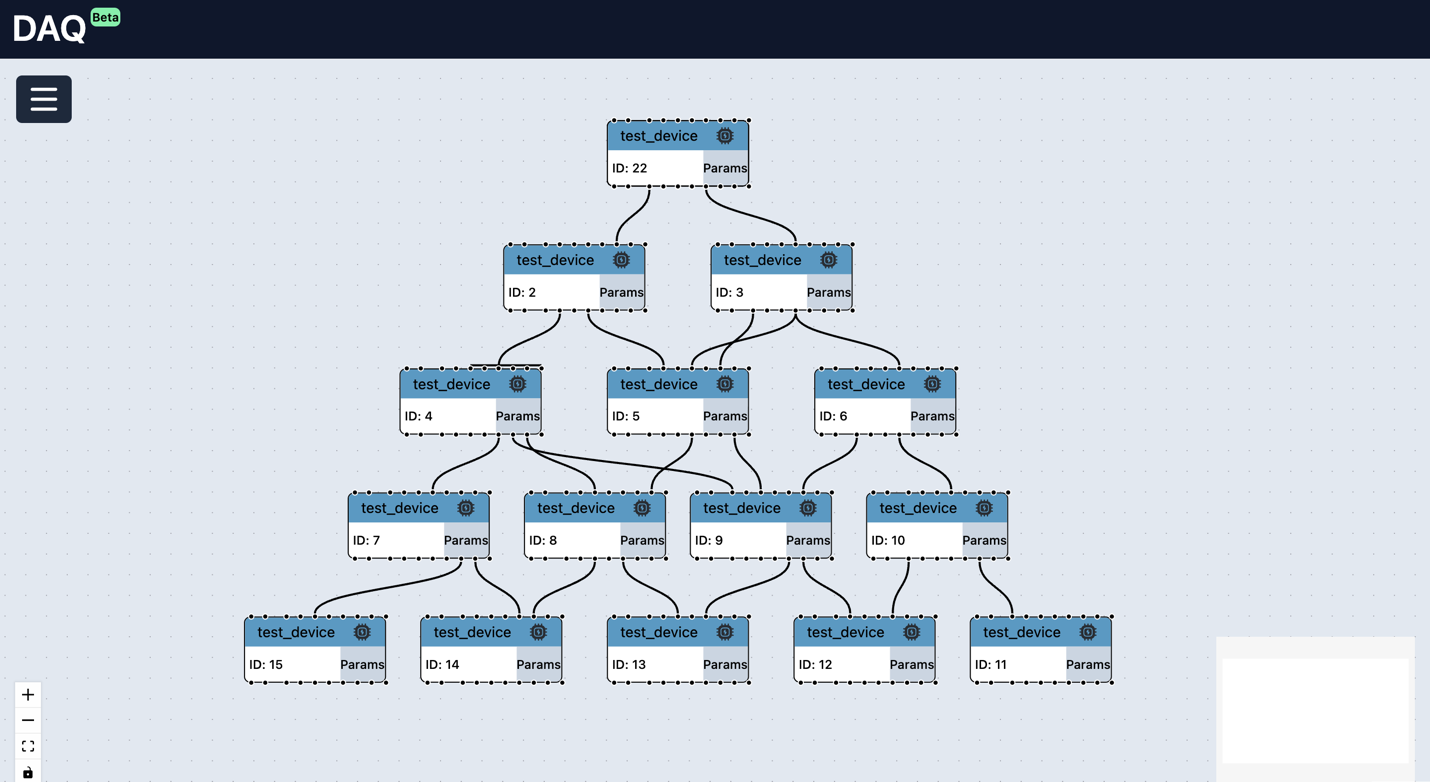


Fig 7 Web interface with an example of displaying components with their interrelationships.

CONCLUSION

In the course of further development of the Hardware Mapping project, the following tasks are expected to be performed:

1. Test the service with real data.
2. When the centralized PostgreSQL service is ready, deploy the production version of the service.
3. Evolve the service in response to user requests.
4. Improve visualisation, allowing customization in order to display parts of the connection scheme that user requires
5. Integrate web interface with the one for the component properties.

The implementation of these tasks will be carried out in cooperation with the developers of DAQ and other communication components of the SPD experiment

BIBLIOGRAPHY

[React Documentation](https://ru.react.js.org/docs/getting-started.html)

[ReactFlow Documentation](https://reactflow.dev/learn)

[PostgreSQL Documentation 16.4](https://www.postgresql.org/docs/16/index.html)

[FastAPI Documentation 0.112.1](https://fastapi.tiangolo.com/)

[Information systems for data taking and data processing, SPD collaboration meeting, 2023](https://indico.jinr.ru/event/3575/contributions/20669/attachments/15249/25742/2023-04-26.Information%20systems%20for%20data%20taking%20and%20data%20processing.pdf)

[TechnicalDesignReport\_SPD2023](http://spd.jinr.ru/wp-content/uploads/2023/03/TechnicalDesignReport_SPD2023.pdf)

ACKNOWLEDGMENTS

The author expresses gratitude to Fedor Prokoshin, Evgeny Khramov and Inga Tvauri for providing comprehensive guidance and moral support in the process of work, as well as for technical support and valuable recommendations. I am also grateful to the organizers of START for the opportunity to complete the program.