

JOINT INSTITUTE FOR NUCLEAR RESEARCH

[Dzhelepov](https://web.archive.org/web/20031204011713/http%3A/dzhelepov.jinr.ru/) laboratory of Nuclear Problems

**FINAL REPORT ON THE**

**START PROGRAMME**

*SPD Hardware Database*

**Supervisor:**

Khramov Evgeny Vladimirovich

**Student:**

Magkaev Rolen Arsenovich, Russia,
North Ossetian State University

**Participation period:**

July 21 – September 14,

Summer Session 2024

Dubna, 2024

TABLE OF CONTENTS

Abstract.....................................................................................................................3

Introduction...............................................................................................................4

History of development.............................................................................................5

SPD information ecosystem......................................................................................6

Analyzing the angular framework and react library.................................................9

Web interface..........................................................................................................12

Implemented HTTP requests..................................................................................15

Conclusion..............................................................................................................17

Literatures...............................................................................................................18

Acknowledgments..................................................................................................19

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.

An information system is being developed to store parameters of the equipment components, and provide means to access it both for personnel and automatic systems. For each type of device, a set of parameters are defined that are common to all devices of this type. The values of these parameters can be specified for each device having unique Hardware ID, or default values of some parameters can be taken. Having in mind large number of equipment, a means for filling data for a groups of devices are being implemented. A web interface is being developed for creating records for a single device or a group of the devices and retrieving information based on several criteria.

INTRODUCTION

The SPD experiment is a scientific project aimed at investigating elementary particle physics and discovering new fundamental principles of particle interaction and structure. The main task of the project is to process data obtained from the SPD detector, which is installed at the NICA accelerator at the Joint Institute for Nuclear Research, located in Dubna, Moscow region.

The SPD detector is a high-tech system capable of recording and analyzing events that occur when polarized deuterons and protons collide at high energies.

Information systems are needed to store the parameters of the equipment and make them accessible to both staff and automated systems. One such system will be Hardware Database, a service for storing and managing data on the parameters of equipment components and their relationships with other elements.

HISTORY OF DEVELOPMENT

When developing the information system for the SPD experiment, it is planned to actively use the experience of creating similar systems for other experiments. For example, a system for storing parameters of hardware components was previously developed for the CMS detector at the LHC. Hardware Database borrows key concepts from that system, such as assigning a unique identifier (HWID) to each component and grouping components into categories. However, the new version of the system uses a different technical stack.

Initially, Hardware Database was written using FastAPI + Angular frameworks and included two tables: groups and components. The parameters of these objects were stored in JSON format, which made data validation and processing difficult. To improve efficiency, a different technology stack was chosen.

To improve performance, the decision was made to switch from Python to Golang, which appears to be faster and implemented new features.

Initially, the web interface was developed using the Angular framework. However, over time it became clear that React (library) is more flexible in terms of scalability, easier to integrate with modern libraries and offers more control over the state of the application, which is especially important when working with dynamic data. Under “Analyzing the angular framework and react library” will describe the benefits of switching to React.

SPD INFORMATION ECOSYSTEM

In addition to Hardware Database, various databases and information systems 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;

At this stage of the SPD experiment, the information and computing systems are in the initial stage of development. Due to the lack of real data, the Hardware Database is being developed on the basis of test generated components.

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

* Obtaining information about device parameters and their interrelationships;
* Processing and transferring information to a database;
* Access to information for data processing and analysis programmes through APIs and applications;
* User access to data retrieval through the implemented web interface;
* Authorization in the system by means of JINR SSO and granting access rights.



Figure 1: Preliminary scheme of the service architecture

The platform for data storage and management was selected based on the projected volumes and flows of information, the structure of equipment records, and anticipated usage scenarios. As a result, it was decided to use PostgreSQL DBMS.

As for a client access, the initial plan was to develop the application as a desktop solution. However, after analyzing the requirement needs and current user information trends, it was decided that web-based solution would more suitable. The following will present the key advantages of a web-based solution over a desktop application:

**Cross-Platform Accessibility**:

* Desktop applications are typically limited to specific operating systems (Windows, macOS, Linux). Developing a separate version for each platform increases the development and maintenance workload.
* Web applications run in browsers, meaning they can be accessed from any device with a browser, regardless of the operating system, providing broader reach to users on any platform—desktop or mobile.

**Ease of Deployment and Updates:**

* Desktop applications require users to download and install the software, and any updates require manual intervention or an update process.
* Web applications allow for instant updates and fixes. Users always access the latest version without needing to download or install anything, ensuring easier maintenance and faster feature rollouts.

**Scalability**:

* Scaling desktop applications across a large user base is more complex, especially when dealing with server-client architectures, data synchronization, or user management.
* Web applications are inherently more scalable, as they can leverage cloud infrastructure, load balancers, and caching techniques, enabling efficient handling of high user traffic and large datasets.

**Responsive and Dynamic Interfaces**:

* While desktop applications can provide rich and dynamic interfaces, they often require more resources to handle various screen sizes and resolutions, especially when extending to mobile or tablet devices.
* Web applications, thanks to modern web technologies like React, offer responsive design, ensuring that the interface adapts seamlessly to various screen sizes and devices, enhancing the user experience across platforms.

**Cost Efficiency**:

* Developing and maintaining separate versions for desktop and mobile can significantly increase costs.
* Web applications require development for only one platform (the web), reducing development costs while ensuring a broader user base.

**Real-Time Data Handling**:

* Many modern applications need to work with real-time data (e.g., notifications, live updates, streaming). Implementing these features can be more challenging in a desktop environment.
* Web applications, with tools like WebSockets and modern APIs, are better suited for real-time data handling, providing smoother and more interactive user experiences.

ANALYZING THE ANGULAR FRAMEWORK AND REACT LIBRARY

**1. Methodology:**

* The analysis was conducted by examining the architectural features of both tools, their performance, state management, scalability support, and ecosystem. We also looked at how each tool handles large amounts of data and dynamic interfaces.

**2. Analysis Results:**

**2.1 Architecture:**

* Angular uses an MVC approach (Model-View-Controller), providing a strict structure for application development. This makes it easier to build scalable projects, though it limits flexibility in choosing technologies.
* React implements the view approach in a MVVM (Model-View-View-Model) pattern where the view is state driven. React provides more freedom to choose tools like Redux, ModX for state management, and React Router for routing.

**2.3 Reactivity and State Management:**

* Angular uses two-way data binding, which can be convenient for simple applications but may become problematic when working with large datasets due to increasing complexity.
* React uses one-way data flow, simplifying state management and making the application's behavior more predictable. This is especially useful when working with large datasets, where minimizing side effects is crucial.

**2.4 Scalability and Performance:**

* Angular supports a modular structure, which simplifies application scaling. However, two-way data binding and the complexity of the framework itself can slow performance when data volume increases.
* React uses virtual DOM (Document Object Model), which significantly improves performance by minimizing updates to the real DOM. This makes React an efficient solution for handling large datasets and frequently changing interfaces.

**2.5 Ecosystem and Tools:**

* Angular comes with a complete set of built-in tools, which simplifies technology selection but limits flexibility. The framework provides

everything needed for development out of the box, which can be an advantage for teams requiring a unified standard.

* React requires selecting additional tools for routing, state management, and server requests. This ensures greater flexibility and adaptability to the specific needs of the project but requires more effort from the team during the architecture setup phase.

**2.6 Testing:**

* Angular supports modular testing out of the box, making it easy to test components, services, and routing.
* React provides convenient tools for testing components using libraries like Jest and React Testing Library, making it a well-testable framework for developing high-quality code.

**3. React Performance Analysis with Large Data:**

When working with large data sets and frequently updated interfaces, React offers several key advantages:

1. Virtual DOM: React uses a virtual DOM to minimize updates to the real DOM. This reduces the load on the browser, which is especially important when processing large amounts of data where a fast response to interface changes is required.
2. One-way Data Flow: React implements a one-way data flow, making state management more transparent. This is especially useful when dealing with large data sets since any data changes are tightly controlled, and interface updates occur predictably.
3. State Management: React provides built-in tools for managing component state, such as hooks (useState, useReducer). For large applications with significant amounts of data, external state management libraries such as Redux or MobX can be used, which scale well and support complex data structures.
4. Optimizing Updates: React includes built-in tools for optimizing component rendering, such as the shouldComponentUpdate method or hooks, allowing developers to minimize unnecessary re-renders and improve performance when working with large data.
5. SSR (Server-Side Rendering): React, with tools like Next.js, supports server-side rendering, which can significantly improve performance when loading large datasets and displaying them on the client side.

**4. Conclusion:**

Based on the analysis, we can conclude:

* Angular is a powerful framework with a full set of built-in tools, well-suited for large applications with a unified architecture. However, when dealing with large datasets, Angular may be less efficient than React due to its architectural complexity and performance issues.
* React offers a more flexible and high-performance solution, especially for tasks involving large data sets and dynamic interfaces. Thanks to the virtual DOM, one-way data flow, and flexibility in choosing tools, React ensures high performance and the ability to optimize interface updates.

WEB IINTERFACE

The following functions are implemented for data management in the Hardware database service:

On this page, the user will see general information about the groups (its name, creation and update time) as well as create a new one. A user with a certain level of access will also be able to manage this data: update, delete and add devices based on the data of a certain (already created) group. The function of full duplication of groups is also implemented. These functions are built into the table itself for convenience.

****

Figure 2: All groups table

Each group has a subtable with properties and additional functionality for ease of use and management of added parameters:



Figure 3: Group properties table

Here the user can manage individual properties and create additional ones. For ease of use, there is a function to save complete group data to a JSON file.

As for the creation of a new group, the following steps should be performed:

* The first step is to create a description for a group of components of the same type, which will include the names of the component properties and their data types (integer, float, longint, etc.). Optionally, the default values of these parameters and the allowed ranges of their values can be specified.



Figure 5: Group creation dialog box

* After creating a group, you can create components based on this group, and fill the values of the parameters. If allowed range of values has been specified for the group, user can enter values only within it. If default values were specified in the group description, they will be used for all components of this group if no value will be filled for specific component.



Figure 6: Device creation dialog box

Table “All devices” where the device created by us according to the group template is presented:



Figure 7: All devices table

IMPLEMENTED HTTP REQUESTS

Some information about the implementation of the requests:

* Each HTTP request is accompanied by a middleware that is responsible for retrieving the JWT token from the request and sending it to the authorization microservice to provide information about the user and their 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 is executed.

 

Figure 8: Group HTTP requests Figure 9: Device HTTP requests



Figure 10: AUTH HTTP requests

CONCLUSION

In the course of further development of the Hardware Database service, the following tasks are expected to be realized:

1. Test the service with real data.
2. Implement support for centralized distribution of user roles using IAM.
3. Further development of the functionality and appearance of the interface in accordance with the wishes of users.

The implementation of these tasks will be carried out in cooperation with the development of other IS installations and depending on their readiness

LITERATURES

[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)

[Angular docs](https://angular.dev/overview)

[React docs](https://devdocs.io/react/)

ACKNOWLEDGMENTS

I am grateful to Evgeny Khramov. Fedor Prokoshin and Inga Tvauri for their comprehensive guidance and technical support during the development process. I would like to thank the START team for the opportunity to go through the program and unforgettable impressions about the time spent.