JOINT INSTITUTE FOR NUCLEAR RESEARCH
Meshcheryakov Laboratory of Information Technologies

FINAL REPORT ON THE START PROGRAMME

Development of tools for visualization of monitoring the state and usage of computing nodes of the Heterogeneous HybriLIT platform

Supervisors:
Dmitry Belyakov,
Maxim Zuev

Student:
Gennadiy Karpov, Russia,
Far Eastern Federal University

Participation period:
March 03 – April 13,
Winter Session 2024

Dubna, 2024

Abstract

This work describes the development of a toolkit for monitoring the status and usage of HybriLIT platform components. The heterogeneous hybrid platform is a part of the Multifunctional Information and Computing Complex (MICC), at the Laboratory of Information Technology, JINR, Dubna. When developing the solution, the disadvantages of existing solutions and the preservation of the design familiar to users of previous systems were taken into account. As a result, a graphical web application with the display of data from the monitoring server in real time was developed.

Table of contents

Introduction

The heterogeneous HybriLIT platform is part of the Multifunctional Information and Computing Complex (MICC), Laboratory of Information Technology, JINR, Dubna. The heterogeneous platform consists of the Govorun Supercomputer and the HybriLIT training and test site [17]. HybriLIT contains a large number of computing machines on which many calculations are performed. Such a large-scale system needs constant monitoring in order to be able to monitor the system status, system load and detect and fix anomalies in time.

The relevance of the work is due to the solution of the shortcomings of the currently existing solutions. The «Home-HLIT» monitoring system based on «Salsa» program code [1] has limited monitoring functionality and complex code, which makes it problematic to support it and add new functionality. Grafana software [2] requires the installation of Telegraf [3] and the development of a custom plug-in for specific monitoring tasks. Also, the expansion of the functionality of the Grafana+Telegraf combination is limited by the current architecture and capabilities of these products, which reduces the flexibility of configuration in comparison with a product developed for a specific platform.

Goals and targets

The purpose of the work

The purpose of the work is to create toolkit for visualizing the monitoring of the state and usage of computing nodes of the Heterogeneous HybriLIT platform.

Tasks

To achieve this goal, it is necessary to solve the following tasks:

1. Define the structure and functionality of the product;
2. Choose technologies for the product implementation;
3. Implement the product.

The structure and functionality of the product

Product requirements

The visualization toolkit for monitoring the status and usage of computing nodes should contain a user interface, a data exchange protocol with the server [16] and the possibility of separate configuration for each user.

The design should be developed based on the design of the previous solution to make it easier for the user to master the new platform.

Product structure

User interface

Taking into account the specified requirements, the product structure looks as follows (Figure 1).

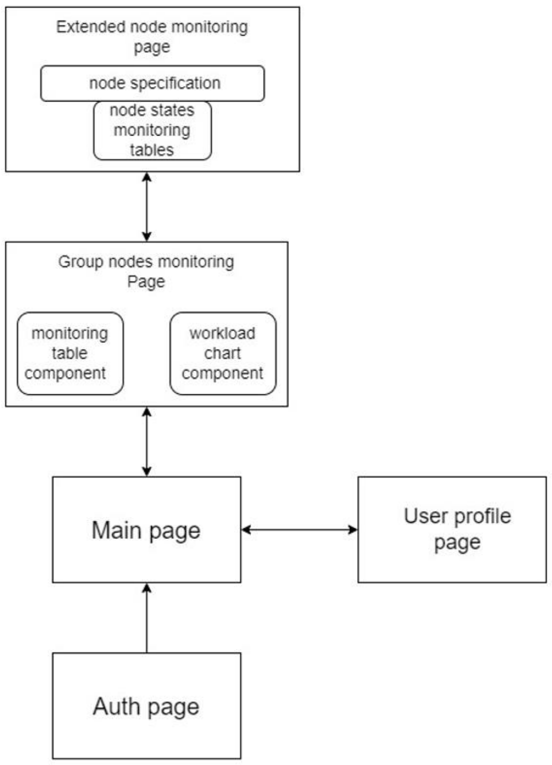

Extended node monitoring page
node specification
node states monitoring tables
Group nodes monitoring Page
monitoring table component
workload chart component
Main page
User profile page
Auth page

Figure 1. Product pages structure

The «User profile» page contains information about the user obtained from the FreeIPA system.

The «Group nodes monitoring» page provides a list of groups of computing nodes. When a group is selected, visual monitoring of the current state of its computing nodes is displayed. Monitoring is presented in the form of charts or tables to choose from.

The «Extended node monitoring» page contains a card about the components of the computing node, their specification and tables with the status and usage of components in real time. The tables are built on the same principle as the monitoring table of the computing nodes of the group.

Client-Server messaging protocol

Data exchange with the server consists of two parts: HTTP requests for authorization and static pages, and continuous data exchange via sockets to obtain the state of computing nodes in real time.

The following protocol has been developed as a protocol for exchanging requests between the server and the client via sockets: the client sends a message with the data type «string», which contains a command and parameters (if required), separated by the symbol «?». As an example, a request for data on the specification of an individual computing node looks like this: «spec?group?name». The server sends a JSON object in the following format:

```
{
    "header": "spec!group!name",
    "field1": {
        ...
    },
    "field2": {
        ...
    }
}
```

Using the key «header», the server transmits information about the response type and parameters (if required), separated by a «!» sign. The remaining keys are directly data on the required request from the client [16].

Authorization

At the initial login, the user enters his username and password, then the server sends a request to FreeIPA and receives a response. In case of successful authentication, the user receives a JWT token [14]. The scheme of initial authorization through the FreeIPA is shown in the diagram (Figure 2).

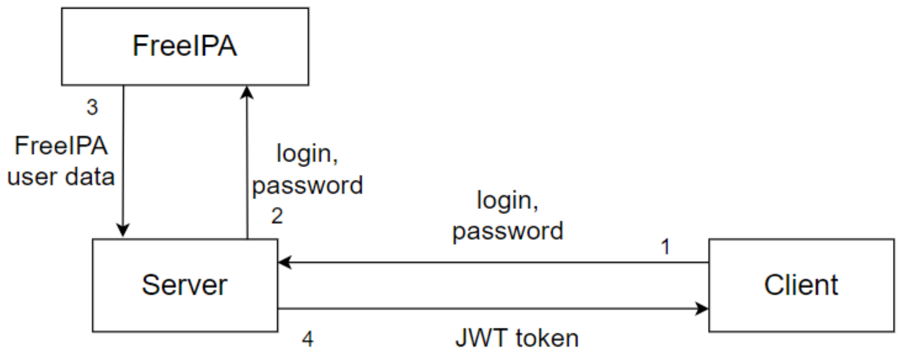


Figure 2. Initial login scheme.

Upon re-authorization, a JWT token is automatically sent from the client, and its correctness is checked on the server. In case of successful authentication, the server returns the user data associated with this JWT token to the client. The scheme of re-authorization is shown in the diagram (Figure 3).

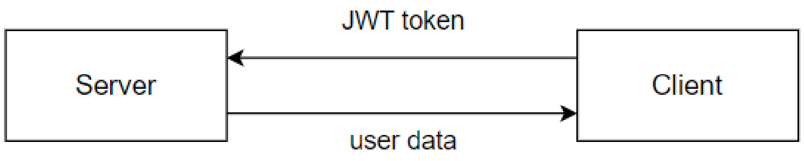


Figure 3. JWT token re-authorization.

Used technologies

To develop interactive components, the Vue.js 3 JavaScript framework was used. Vue.js 3 has support for reactivity, global storage, and the creation of reusable components [4]. Pinia is used as the global state management library, currently the official state management library for Vue.js 3 [5].

To create charts components, the vue-chartjs library was used, which is a library adapted to the Vue.js 3 component approach Chart.js [18, 19].

The design of web pages was created using the Bootstrap 5 library. Bootstrap is a set of ready-made templates and tools for creating a web application interface written in HTML, CSS and JavaScript languages [6].

HTTP requests to the server were made through the Axios library [11]. It allows to write compact code for sending, receiving and verifying HTTP requests.

To communicate with the server via sockets, the JavaScript built-in implementation of the web sockets protocol was used [7].

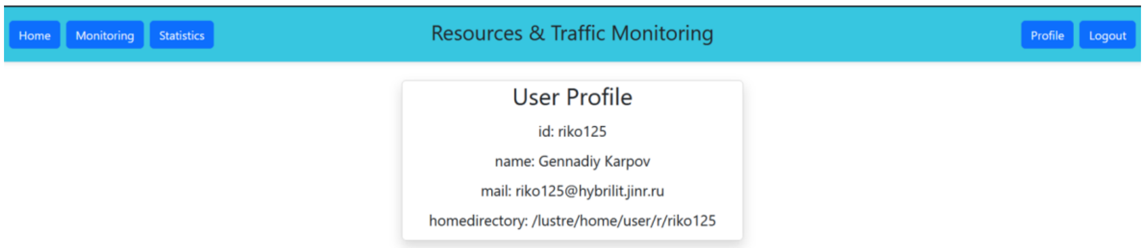
Authorization via the FreeIPA system was implemented on the server using the python-freeipa library [12].

Product implementation

User Interface

The user interface is a web application consisting of several pages and components.

The «User profile» page contains information about the user obtained from the FreeIPA system. The design of the page is shown in the Figure 4.



User Profile

id: riko125

name: Gennadiy Karpov

mail: riko125@hybrilit.jinr.ru

homedirectory: /lustre/home/user/r/riko125

Figure 4. User profile page design.

The monitoring page contains a list of buttons with the names of groups of computing nodes. When the group button is clicked, a visual monitoring of the current state of the computing nodes contained within the selected group is displayed. Monitoring is presented in the form of Pie-type charts and table.

Chart components display the average real-time load of a parameters of all computing nodes of the group. The design of charts is shown in the Figure 5.

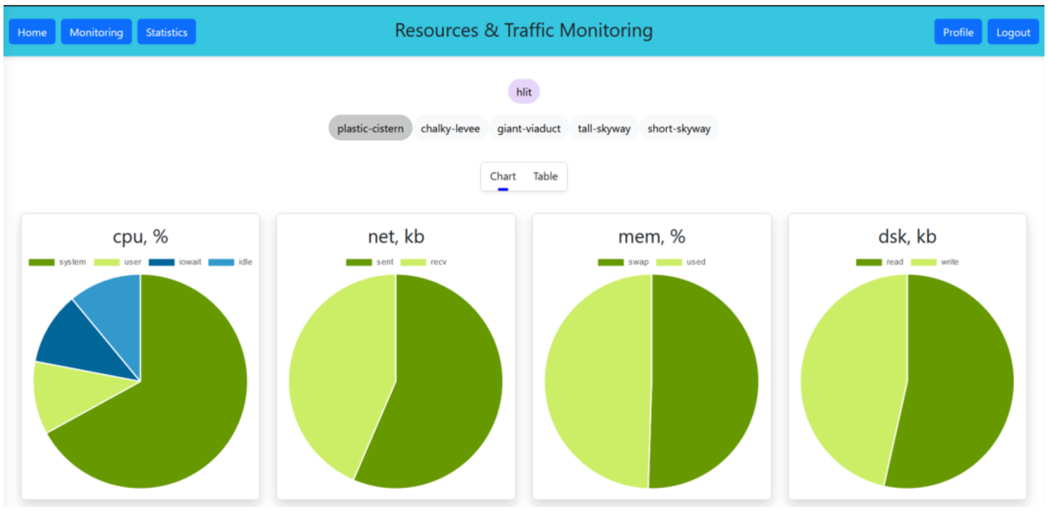
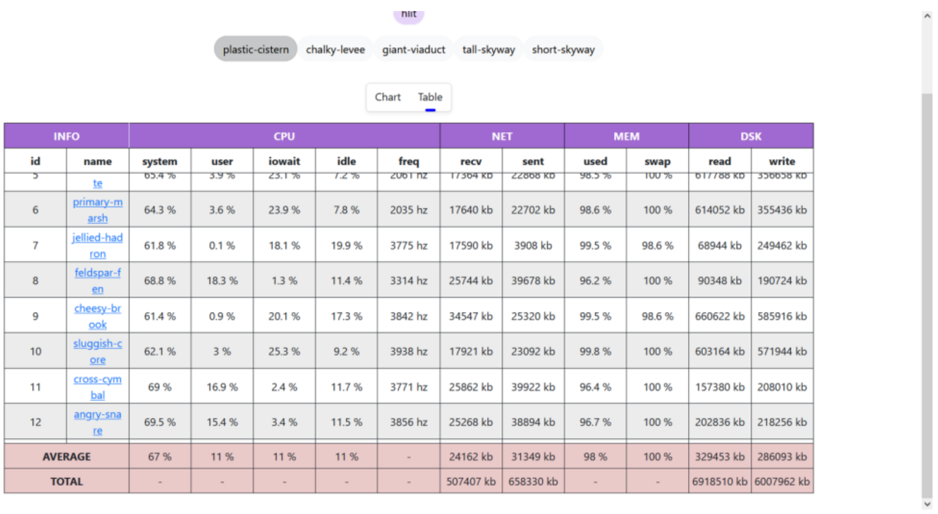


Figure 5. Chart components design.

The table component of the computing nodes of the group is a component consisting of three parts: header, body, footer. The header displays the categories and subcategories of computing node components. The footer displays the average and total usage by subcategory, if statistics are informative for a type of such data. The body directly displays information about each computing node in the group. By clicking on the name of the computing node, the view is switched to extended monitoring with more detailed information about its specification and loading. The table has a fixed height and always fits completely on the screen, which increases the convenience of viewing data relative to the table implemented in the previous «Home-HLIT» monitoring system [1], since the category names indicated in the header and data on the average and total load of the subcategory for all computing nodes are always visible. If the number of computing nodes does not fit into the table, it is possible to scroll through the «body» part of the table. The design of the table is shown in the Figure 6.

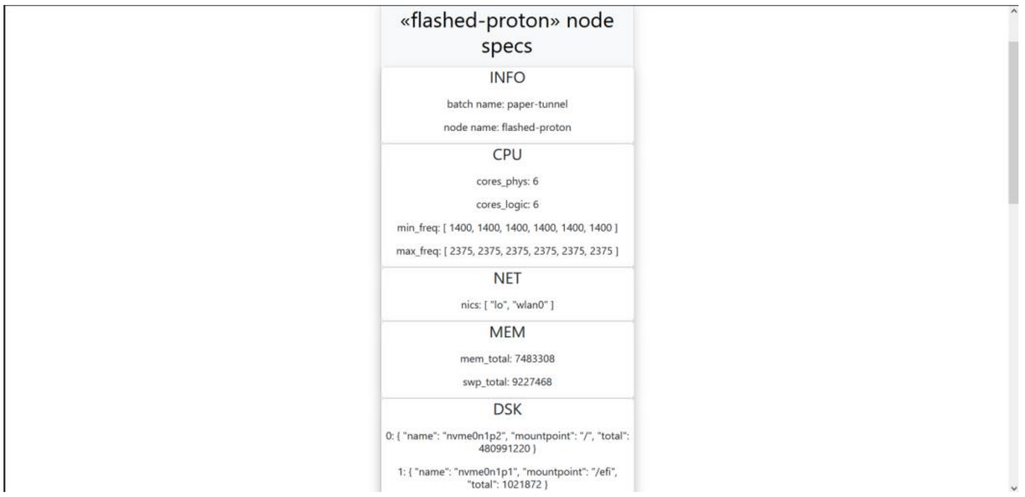


INFO		CPU					NET		MEM		DSK	
id	name	system	user	iowait	idle	freq	recv	sent	used	swap	read	write
5	te	65.4 %	3.9 %	23.1 %	7.2 %	2061 hz	17364 kb	22868 kb	98.5 %	100 %	617788 kb	356658 kb
6	primary-marsh	64.3 %	3.6 %	23.9 %	7.8 %	2035 hz	17640 kb	22702 kb	98.6 %	100 %	614052 kb	355436 kb
7	jellied-hadron	61.8 %	0.1 %	18.1 %	19.9 %	3775 hz	17590 kb	3908 kb	99.5 %	98.6 %	68944 kb	249462 kb
8	feldspar-fen	68.8 %	18.3 %	1.3 %	11.4 %	3314 hz	25744 kb	39678 kb	96.2 %	100 %	90348 kb	190724 kb
9	cheesy-brook	61.4 %	0.9 %	20.1 %	17.3 %	3842 hz	34547 kb	25320 kb	99.5 %	98.6 %	660622 kb	585916 kb
10	sluggish-core	62.1 %	3 %	25.3 %	9.2 %	3938 hz	17921 kb	23092 kb	99.8 %	100 %	603164 kb	571944 kb
11	cross-cymbal	69 %	16.9 %	2.4 %	11.7 %	3771 hz	25862 kb	39922 kb	96.4 %	100 %	157380 kb	208010 kb
12	angry-snare	69.5 %	15.4 %	3.4 %	11.5 %	3856 hz	25268 kb	38894 kb	96.7 %	100 %	202836 kb	218256 kb
AVERAGE		67 %	11 %	11 %	11 %	-	24162 kb	31349 kb	98 %	100 %	329453 kb	286093 kb
TOTAL		-	-	-	-	-	507407 kb	658330 kb	-	-	6918510 kb	6007962 kb

Figure 6. Table component design.

The «Extended node monitoring» page contains two components: a card with the specification of the computing node and a table with the status and usage of components in real time.

The specification card of the computing node contains information about the components of the computing node and their specification. The design of the specification card is shown in the Figure 7.



«flashed-proton» node specs

INFO

batch name: paper-tunnel

node name: flashed-proton

CPU

cores_phys: 6

cores_logic: 6

min_freq: [1400, 1400, 1400, 1400, 1400, 1400]

max_freq: [2375, 2375, 2375, 2375, 2375, 2375]

NET

nics: ["lo", "wlan0"]

MEM

mem_total: 7483308

swp_total: 9227468

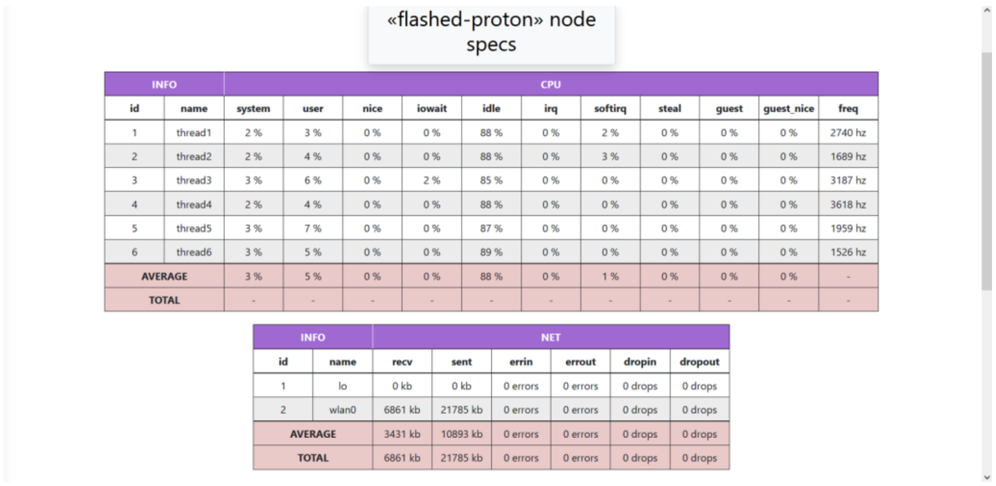
DSK

0: { "name": "nvme0n1p2", "mountpoint": "/", "total": 480991220 }

1: { "name": "nvme0n1p1", "mountpoint": "/efi", "total": 1021872 }

Figure 7. Specification card design.

The tables are built on the same principle as the monitoring table for computing nodes in the group. The design of the tables is shown in the Figure 8.


«flashed-proton» node specs

INFO		CPU										
id	name	system	user	nice	iowait	idle	irq	softirq	steal	guest	guest_nice	freq
1	thread1	2 %	3 %	0 %	0 %	88 %	0 %	2 %	0 %	0 %	0 %	2740 hz
2	thread2	2 %	4 %	0 %	0 %	88 %	0 %	3 %	0 %	0 %	0 %	1689 hz
3	thread3	3 %	6 %	0 %	2 %	85 %	0 %	0 %	0 %	0 %	0 %	3187 hz
4	thread4	2 %	4 %	0 %	0 %	88 %	0 %	0 %	0 %	0 %	0 %	3618 hz
5	thread5	3 %	7 %	0 %	0 %	87 %	0 %	0 %	0 %	0 %	0 %	1959 hz
6	thread6	3 %	5 %	0 %	0 %	89 %	0 %	0 %	0 %	0 %	0 %	1526 hz
AVERAGE		3 %	5 %	0 %	0 %	88 %	0 %	1 %	0 %	0 %	0 %	-
TOTAL		-	-	-	-	-	-	-	-	-	-	-

INFO		NET					
id	name	recv	sent	errin	errout	dropin	dropout
1	lo	0 kb	0 kb	0 errors	0 errors	0 drops	0 drops
2	wlan0	6861 kb	21785 kb	0 errors	0 errors	0 drops	0 drops
AVERAGE		3431 kb	10893 kb	0 errors	0 errors	0 drops	0 drops
TOTAL		6861 kb	21785 kb	0 errors	0 errors	0 drops	0 drops

Figure 8. Extended monitoring tables design.

Monitoring components are not fixed, the number of charts and categories in the tables change depending on the data received from the server.

To test the interface, virtual computing nodes and their groups were created.

Client-Server messaging protocol

The client to server messaging protocol has the following commands:

- «lsob» — get a list of groups of computing nodes;
- «head?group» — get a list of categories and subcategories of components of computing nodes in a group of computing nodes;
- «mstd?group» — subscribe to receive data for each computing node in the group;
- «spec?group?name» — get the specification of a selected computing node;
- «desc?group?name» — get categories and subcategories based on components of a selected computing node;
- «mext?group?label» — subscribe to receive data of components of a selected computing node.

The server to client messaging protocol has the following commands:

- «lsob» — send a list of groups of computing nodes;

- «head!group» — send a list of categories and subcategories of computing node components in a group of computing nodes;
- «mstd!group» — subscribe the client to receive data for each computing node in the group;
- «spec!group!name» — send the specification of a selected computing node;
- «desc!group!name» — send categories and subcategories based on components of a selected computing node;
- «mext!group!label» — subscribe the client to receive data of components of a selected computing node.

After receiving the data from the server, it is stored in the global state store [5]. After that, upon request from the page, the data is provided, preprocessed and transferred to the component. The scheme of data communication within the web application and with the server is shown in the Figure 9.

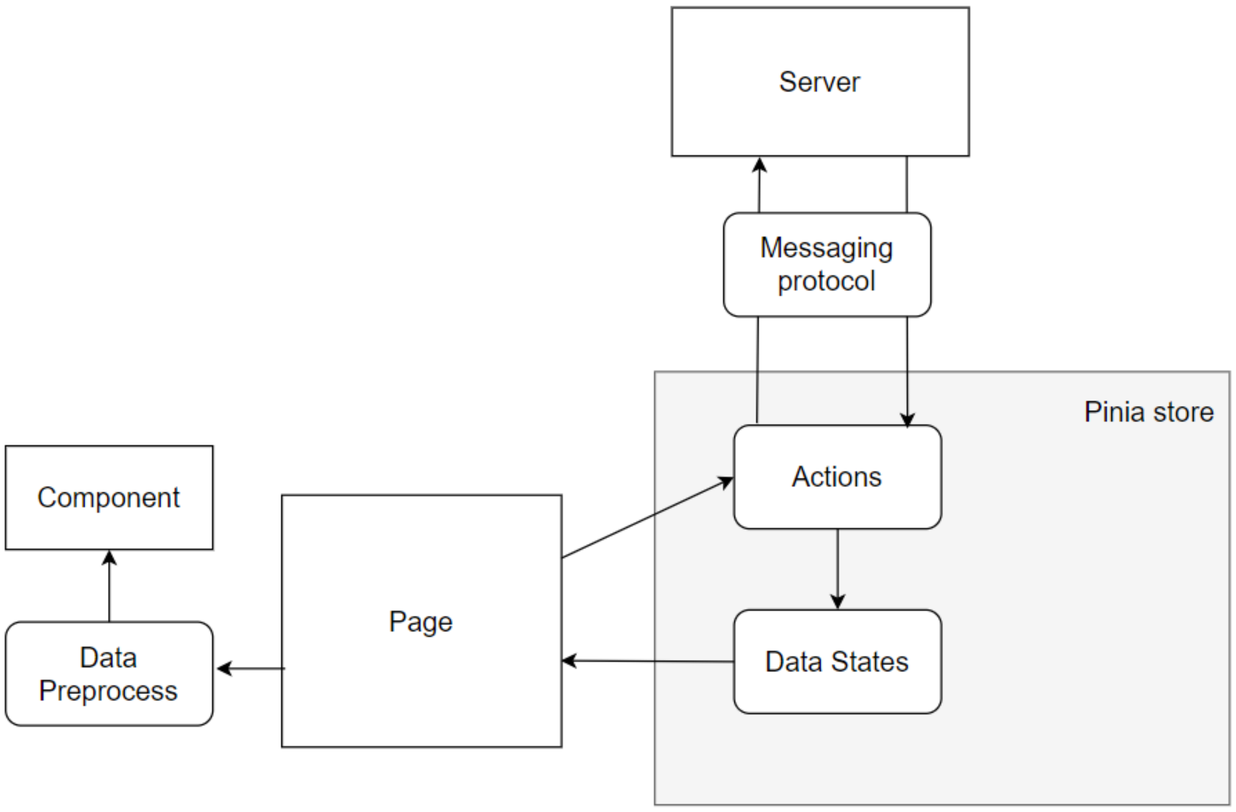


Figure 9. The scheme of data communication.

Authorization

Authorization in the web application is performed using a JWT token, which is generated when user logs in using an account registered in the FreeIPA and contains information about the user. This approach allows to synchronize access to the web application with employee account and configure access to various parts of the web application depending on the rights of a particular employee.

The JWT token is stored in HTTP-only cookies, which provides additional protection. There is no access to HTTP-only from JavaScript, it helps to avoid XSS hacks [13].

The «Authorization» page is shown in the Figure 10.

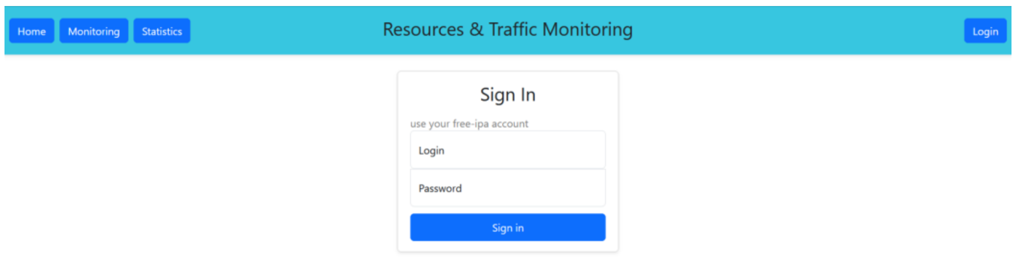


Figure 10. Authorization page design.

Conclusion

The developed toolkit provides the solution to the given task.

The developed solution has extended monitoring capabilities: it is possible to view both the status and usage of the whole computing group and separately of a specific computing node.

Web application design has retained the concept of the old solution used, but has received important improvements that will simplify the system usage.

The developed solution has the ability to scale in the future. In the future, it is possible to increase the number of different types of data representation: add more different charts, display data not only at the moment, but also for a period of time, add a display of network traffic sources and destinations. Also, by synchronizing with third-party authorization systems, it will be possible to customize the displayed data depending on the user's rights.

Acknowledgements

I would like to extend my heartfelt gratitude to the entire START and JINR teams for making my participation in this event possible. I am also thankful to our supervisors at JINR, Belyakov Dmitry Viktorovich and Zuev Maxim Igorevich, for their extensive support, and to Sushchenko Andrey Andreevich from FEFU for the opportunity to work outside my university walls and improve my skills. Additionally, Maxim Skazkin deserves a special mention for his productive work and collaboration on the project.

References

1. SALSA. — 2017. — URL: https://home-hlit.jinr.ru/#/ (visited on 03/05/2024).
2. Grafana. — 2024. — URL: https://grafana.com/ (visited on 03/05/2024).
3. Telegraf | InfluxData. — 2024. — URL: https://www.influxdata.com/time-series-platform/telegraf/ (visited on 03/05/2024).
4. Vue.js 3. — 2024. — URL: https://v3.ru.vuejs.org/ru/ (visited on 03/12/2024).
5. Pinia. — 2024. — URL: https://pinia.vuejs.org/ (visited on 03/13/2024).
6. Bootstrap. — 2024. — URL: https://getbootstrap.com/ (visited on 03/10/2024).
7. WebSockets docs. — 2024. — URL: https://developer.mozilla.org/ru/docs/Web/API/WebSockets_API (visited on 03/20/2024).
8. JavaScript docs. — 2024. — URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (visited on 03/11/2024).
9. HTML docs. — 2024. — URL: https://developer.mozilla.org/en-US/docs/Web/HTML (visited on 03/10/2024).
10. CSS docs. — 2024. — URL: https://developer.mozilla.org/en-US/docs/Web/CSS (visited on 03/10/2024).
11. Axios docs. — 2024. — URL: https://axios-http.com/ru/docs/intro (visited on 03/18/2024).
12. Python FreeIPA client docs. — 2024. — URL: https://python-freeipa.readthedocs.io/en/latest/ (visited on 03/21/2024).
13. Cookies docs. — 2024. — URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies (visited on 03/12/2024).
14. JWT-token. — 2024. — URL: https://jwt.io/ (visited on 03/23/2024).
15. *Фримен Эрик, Робсон Элизабет.* Изучаем программирование на JavaScript. — O‘Reilly Media, Inc, 2022.

16. *Maxim Skazkin.* Development of the server part of the system for monitoring the state of computing nodes of the Heterogeneous HybriLIT platform based on asyncronous data transfer technologies and the use of web sockets. — Dubna, 2024.
17. Heterogeneous platform “HybriLIT”. — 2024. — URL: http://hlit.jinr.ru/en/ (visited on 03/04/2024).
18. Vue-chartjs docs. — 2024. — URL: https://vue-chartjs.org/ (visited on 04/01/2024).
19. Chart.js docs. — 2024. — URL: https://www.chartjs.org/ (visited on 04/01/2024).
20. Fastapi. — 2024. — URL: https://fastapi.tiangolo.com/ (visited on 03/14/2024).