ДОДАТОК А Апробація результатів наукових досліджень

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки

VIII Міжнародна Конференція

ВИРОБНИЦТВО
&
МЕХАТРОННІ СИСТЕМИ 2024

VIII International Conference

MANUFACTURING
&
MECHATRONIC SYSTEMS 2024

M&MS
2024
VII International Conference
25-26 October
Kharkiv

УДК: 005:004.896:62-65:338.3

Виробництво & Мехатронні Системи 2024: матеріали VIII-ої Міжнародної конференції, Харків, 25-26 жовтня 2024 р.: тези доповідей / [редкол. І.Ш. Невлюдов (відповідальний редактор)].-Харків: [електронний друк], 2024. – 135 с.

У збірник включені тези доповідей, які присвячені сучасним тенденціям розвитку технологій та засобів виробництва та мехатронних систем, передовому досвіду та впровадженню їх в галузях систем промислової автоматизації та керування виробництвом; системній інженерії; CAD/CAM/CAE системах; мехатроніці (електро-механічних системах, електронних інструментах систем керування, механічних CAD системах); робототехніці та засобахм інтелектуалізації; MEMS (сучасних матеріалів та технологіях виготовлення MEMS) та компонентах і технологіях автоматизації видобутку, переробки та транспортування нафти та газу.

Редакційна колегія: І.Ш. Невлюдов, В.В. Євсєєв.

Manufacturing & Mechatronic Systems 2024: Proceedings of VIII st International Conference, Kharkiv, October 25-26, 2024: Thesises of Reports / [Ed. I.Sh. Nevlyudov (chief editor).] .- Kharkiv .: [electronic version], 2024. - 135 p.

The collection includes the thesises of reports on modern trends in the development of technologies and means of production and mechatronic systems, top experience and implementation of them in fields of: industrial automation and production management systems; systems engineering; CAD/CAM/CAE systems; mechatronics (electrical and mechanical systems, electronic control tools, mechanical CAD systems); robotics and intellectual toolls; MEMS (modern materials and manufacturing technologies MEMS) and components and technologies for the automation of oil, gas and oil extraction, processing and transportation.

Editorial board: Igor.Sh. Nevludov, Vladyslav.V. Yevsieiev

Міністерство освіти і науки України (МОНУ)
Харківський національний університет радіоелектроніки (ХНУРЕ)
Варшавський університет сільського господарства (WULS - SGGW)
Азербайджанський державний університет нафти і промисловості
Національний університет «Львівська політехніка»
Festo Didactic Україна
Jabil Circuit Ukraine Limited
ТОВ «Науково-виробниче підприємство «УКРІНТЕХ»»
Факультет автоматики і комп'ютеризованих технологій (АКТ)
Кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР),
Державне підприємство «Харківський науково-дослідний інститут технології машинобудування»
Державне підприємство «Південний державний проектно-конструкторський та науково-дослідний інститут авіаційної промисловості»

МАТЕРІАЛИ

VIII-ої Міжнародної Конференції

ВИРОБНИЦТВО
&
МЕХАТРОННІ СИСТЕМИ 2024

(25-26 жовтня 2024)
Харків, Україна

ОРГАНІЗАТОРИ

Міністерство освіти і науки України (МОНУ)

The Ministry of Education and Science of Ukraine

Харківський національний університет радіоелектроніки (ХНУРЕ)

Kharkiv National University of Radioelectronics

Варшавський університет сільського господарства (WULS - SGGW)

Warsaw University of Life Sciences WULS - SGGW

Азербайджанський державний університет нафти і промисловості

Azerbaijan State Oil and Industry University

Festo Didactic Україна

Festo Didactic Ukraine

ТОВ «Науково-виробниче підприємство «УКРІНТЕХ»»

Research and Production Enterprise "UKRINTECH" Ltd

Національний університет «Львівська політехніка»

National University Lviv Polytechnic

Державне підприємство «Харківський науково-дослідний інститут технології машинобудування», м . Харків , Україна

State Enterprise «Kharkiv Scientific-Research Institute of Mechanical Engineering Technology», Kharkiv, Ukraine

Державне підприємство «Південний державний проектно-конструкторський та науково-дослідний інститут авіаційної промисловості», м. Харків, Україна

State Enterprise «National Design & Research Institute of Aerospace Industries», Kharkiv, Ukraine

Jabil Circuit Ukraine Limited

КОМІТЕТ КОНФЕРЕНЦІЇ

МІЖНАРОДНИЙ ПРОГРАМНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Ігор Шакирович Невлюдов	голова комітету конференції, доктор технічних наук, професор, заслужений діяч науки і техніки України, лауреат Державної премії в галузі науки і техніки України; лауреат Державної премії України в галузі освіти, завідувач кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР), Харківського національного університету радіоелектроніки, Україна
Олександр Іванович Филипенко	заступник голови комітету конференції, доктор технічних наук, професор, лауреат Державної премії України в галузі освіти, декан факультету Автоматики і комп'ютеризованих технологій (АКТ), Харківського національного університету радіоелектроніки, Україна.
Мурад Анвер огли Омаров	доктор технічних наук, професор, заслужений діяч науки Азербайджанської Республіки проректор з міжнародного співробітництва, Харківський національний університет радіоелектроніки, Україна
Владислав В'ячеславович Євсєєв	секретар, доктор технічних наук, професор, професор кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР), Харківського національного університету радіоелектроніки, Україна.
Andrzej Chochowski	доктор технічних наук, професор Варшавського університету сільського господарства (WULS - SGGW), Польща
Pawel Obstawski	доктор технічних наук, професор Варшавського університету сільського господарства (WULS - SGGW), Польща.
Сергій Богомолов	лектор/доцент, доктор філософії (комп'ютерні науки), Дослідницька школа комп'ютерних наук, Коледж інженерії та комп'ютерних наук, Австралійський національний університет, Австралія.
Микола Васильович Замірець	доктор технічних наук, професор, заслужений діяч науки і техніки України, лауреат Державної премії України в галузі науки і техніки, директор Державного підприємства Науково-дослідного технологічного інституту приладобудування, Україна
Михайло Васильович Лобур	доктор технічних наук, професор, заслужений діяч науки і техніки України, відмінник народної освіти України, завідувач кафедри систем автоматизованого проектування Національного університету «Львівська політехніка», Україна.
Євген Сергійович Риженко	керівник відділу дидактики ДП «Фесто», Україна

Сергій Володимирович Демченко	директор ТОВ «Науково-виробничого підприємства «УКРІНТЕХ»», Україна.
Самед Імамалі огли Юсіфов	кандидат технічних наук, доцент, декан факультету інформаційних технологій та управління, Азербайджанський державний університет нафти і промисловості, Азербайджан.
Фарід Гаджі огли Агаєв	кандидат технічних наук, доцент, завідувач кафедри управління та системної інженерії, Азербайджанський державний університет нафти і промисловості, Азербайджан.
Віктор Васильович Косенко	доктор технічних наук, професор, професор кафедри автоматики, електроніки та телекомунікацій, Національний університет «Полтавська політехніка імені Юрія Кондратюка» Україна.
Володимир Вікторович Козирський	доктор технічних наук, професор, заслужений діяч науки і техніки України, директор Навчально-наукового інституту енергетики, автоматики та енергозбереження, Національний університет біоресурсів і природокористування України, Україна.
Віталій Пилипович Лисенко	доктор технічних наук, професор, заслужений працівник освіти України, завідувач кафедри автоматики та робототехнічних систем ім. акад. І.І. Мартиненка, Національний університет біоресурсів і природокористування України, Україна.
Юрій Францевич Зіньковський	доктор технічних наук, лауреат Державної премії України в галузі освіти, професор кафедри радіоконструювання і виробництва радіоапаратури, Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Україна.
Володимир Митрофанович Свищ	доктор технічних наук, професор, заслужений діяч науки і техніки України, професор кафедри систем управління літальними апаратами, Національний аерокосмічний університет ім. М. Є. Жуковського "Харківський авіаційний інститут", Україна.
Віталій Євгенович Овчаренко	доктор технічних наук, професор, заступник директора з наукової роботи Державного підприємства «Науково-дослідний технологічний інститут приладобудування», Україна.
Лариса Сергіївна Глоба	доктор технічних наук, професор, лауреат Державної премієї Кабінету Міністрів України, завідувач кафедри інформаційно-комунікаційних мереж, Інститут телекомунікаційних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», Україна.
Анатолій Олександрович Андрусевич	доктор технічних наук, професор, начальник Криворізького коледжу Національного авіаційного університету, Україна.

Роман Володимирович Артюх	кандидат технічних наук, директор Державного підриємтва «Південний державний проектно-конструкторський інститут авіаційної промисловості», Україна.
Kurtwitz	генеральний менеджер Titan Machinery Limited, Шотландія.
Liu Shan	генеральний менеджер Titan Machinery Limited, Китай.
Володимир Андрійович Павлиш	кандидат технічних наук, професор, заслужений діяч науки і техніки України», перший проректор Національного університету «Львівська політехніка», Україна
Сергій Іванович Осадчий	доктор технічних наук, професор, в.о. завідувача кафедри конструкції повітряних суден, авіадвигунів та підтримання льотної придатності, Льотна академія НАУ, м.Кропивницький, Україна.
Анатолій Афанасійович Єфіменко	доктор технічних наук, професор, завідувач кафедри електронних засобів та інформаційно-комп'ютерних технологій, Одеський національний політехнічний університет, Україна
Володимир Михайлович Решетюк	кандидат технічних наук, доцент кафедри автоматики та робототехнічних систем ім. акад. І.І. Мартиненка, Національний університет біоресурсів і природокористування України, Україна.

ОРГАНІЗАЦІЙНИЙ КОМІТЕТ КОНФЕРЕНЦІЇ

Олександр Михайлович Цимбал	заступник голови конференції з організаційних питань, доктор технічних наук, професор комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР), Харківський національний університет радіоелектроніки, Україна.
Сергій Павлович Новоселов	кандидат технічних наук, доцент, професор кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР), Харківський національний університет радіоелектроніки, Україна.
Євген Анатолійович Разумов-Фризюк	кандидат технічних наук, доцент, доцент кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР), Харківський національний університет радіоелектроніки, Україна.
Наталія Павлівна Демська	кандидат технічних наук, доцент, доцент кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР), Харківський національний університет радіоелектроніки, Україна.

ЗМІСТ

Automation of Power Grid Element Management to Enhance Energy Efficiency

Artem Hubar
KITPVm-23-2, **Kharkiv National University of Radioelectronics**
Ukraine, Kharkiv, 14 Nauky Ave, email: artem.hubar@nure.ua.

Abstract: This paper discusses innovative solutions in the automation of power grid management aimed at increasing energy efficiency in Ukraine, particularly in the context of energy shortages caused by the war. Specific technologies and their impact on energy security are proposed.

Keywords: automation, energy efficiency, power grids, Smart Grid, energy security.

I. Introduction

Energy is a valuable resource, limited in its availability. In the modern world, humanity is constantly seeking ways to improve energy consumption efficiency, as this issue is not only about savings but also about responsibility to future generations.

In the context of a global energy crisis, driven by rising prices and resource depletion, it is important to realize that our electricity consumption today has a direct impact on the lives of future generations. The quality of life of our descendants and their ability to enjoy comfort depend on the approaches we choose for utilizing energy resources.

This awareness drives the search for innovative technologies and solutions that can reduce energy consumption and lessen the negative impact on the environment. It is necessary not only to meet the needs of today but also to leave a sustainable and clean world for future generations. In this context, the automation of managing the elements of the power grid is critically important, as it allows for optimizing energy consumption and ensuring the effective use of resources, which, in turn, will contribute to the preservation of our planet

II. Overview of Relevance

With the onset of the war in Ukraine, the relevance of efficient electricity consumption has significantly increased. The conflict has led to energy shortages, resulting in numerous power outages. This has created new challenges for consumers, businesses, and the state as a whole, as it affects not only the comfort of citizens but also the economic stability of the country.

In conditions of electricity scarcity, it is essential not only to reduce consumption but also to implement new automation technologies capable of significantly enhancing energy efficiency. In this context, the automation of managing elements of the power grid gains particular importance, as it can become a key factor in ensuring reliable and efficient use of energy resources.

In the context of Ukraine's energy security, it is important to note that the Energy Security Strategy of Ukraine emphasizes the need to synchronize Ukrainian energy systems with European ones, which will allow for sustainable development of the energy sector, particularly through the automation of networks. In this regard, the automation of power grids is one of the key steps toward achieving a secure and efficient energy system [1].

Analysis shows that the relevance of research and publications related to the automation of managing elements of power grids to enhance their energy efficiency is very high. One of the leading technologies is Smart Grid. In 1886, the first alternating current power grid was launched in Great Barrington, Massachusetts. It was a centralized system managed on demand.

In the 20th century, local networks began to interconnect to improve efficiency and reliability. By the 1960s, they transformed into large systems supplying energy from thousands of power plants to industrial and residential consumers through high-voltage lines.

The rapid increase in demand since the 1970s required the construction of new power plants. During peak loads, this led to outages, prompting consumers to demand more reliable services. By the end of the 20th century, heating and air conditioning systems had become the main energy consumers, leading to peak loads during the day. This increased the costs of maintaining the grid, which were reflected in tariffs.

In the 21st century, countries like South Korea, China, India, and Brazil have become leaders in implementing modern smart energy systems. In the 2020s, the number of publications on this topic in Ukraine has increased nearly threefold; a pie chart comparing the number of publications is presented in Fig. 1.

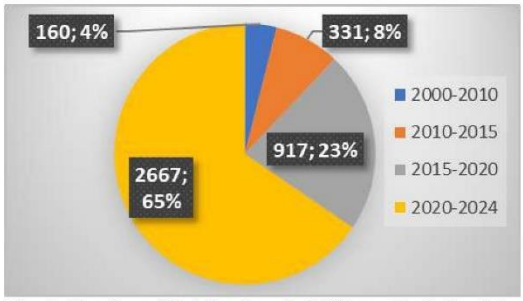


Fig. 1. Number of Publications in Different Periods of the 21st Century

The diagram shows the number of published articles on the topic of automation of managing elements of power grids to enhance their energy efficiency for four-time intervals. The percentages in the diagram reflect the share of each interval in the total number of publications released on this topic in the 21st century.

Figure 2 presents a diagram demonstrating the dynamics of the number of published articles over the period from 2020 to 2024.

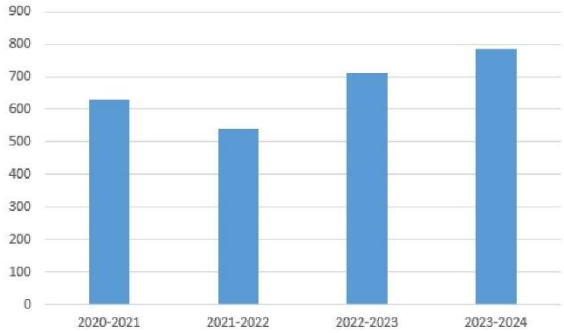


Fig. 2. Number of Publications in Recent Years

As can be seen from the figures, the number of publications continues to grow. According to the Energy Security Strategy of Ukraine, about 50% of Ukraine's power grid is in a state of disrepair, significantly complicating its operation. This situation necessitates the implementation of innovative solutions, such as monitoring and automation systems, which can help restore and optimize the functioning of the networks. The automation of networks will reduce the load on the aging infrastructure and enhance its efficiency [1].

Automation of managing elements of power grids to improve energy efficiency encompasses several key areas. One of these is smart grids (Smart Grid), such as ABB Ability and Siemens Spectrum Power. ABB Ability enables the automation, optimization, and resilience of operations across various sectors, increasing the reliability and efficiency of energy networks [2]. Siemens Spectrum Power specializes in reliable and efficient management of energy markets, allowing for the integration of various energy sources and real-time management [3].

Another area is substation automation. Schneider Electric's EcoStruxure solution allows for monitoring and remote control of key components of the power grid, significantly enhancing its operational efficiency [4]. GE Grid Solutions provides solutions for the transmission and distribution of energy, facilitating continuous energy supply and minimizing losses [5].

In the field of demand response management, notable solutions include Oracle Utilities Opower and Enel X Demand Response. Oracle Opower employs artificial intelligence and behavioral technologies to influence consumers, helping them reduce energy consumption through analysis and information [6]. Enel X Demand Response enables consumers to adjust their energy consumption based on the state of the grid, helping to reduce load and increase stability [7].

The integration of renewable energy sources is an essential component of modern power grids. Schneider Electric's Advanced Distribution Management System (ADMS) provides automatic balancing of energy flows and minimizes losses, allowing for the effective integration of solar and wind energy [8]. Siemens' Decentralized Energy Management integrates decentralized energy sources, optimizing their use and increasing grid flexibility [9].

Particular attention should be paid to real-time monitoring and control systems. With solutions like Honeywell Energy Control System and OSIsoft PI System, it is possible not only to continuously monitor the condition of the grid but also to automatically adjust energy flows to maximize efficiency [10]. These systems provide analysis, load management, and help reduce energy losses in networks [11].

The results of implementing automation in power grids and energy-efficient technologies in various countries demonstrate significant progress in improving energy consumption efficiency. In the U.S., for instance, Smart Grid technologies have become the foundation for modernizing the power grid following the Recovery and Reform Act of 2009, which allocated about $4.5 billion. As a result of these measures, electricity losses have been reduced by 10-15%, and the reliability of energy supply has improved, particularly in California, where the duration of outages has decreased by 25%. The implementation of renewable energy sources has also gained momentum, with wind and solar energy now accounting for about 20% of the overall electricity consumption balance in some states [12].

Germany is implementing the "Energy Transition" (Energiewende) policy, which involves gradually reducing dependence on fossil fuels and increasing the share of renewable sources. In 2020, the share of renewable sources in total electricity production exceeded 50%, and CO2 emissions decreased by 42% compared to 1990 levels. The deployment of smart meters allows consumers to monitor their consumption and optimize electricity costs [13].

In South Korea, the Smart Grid project has been implemented in the city of Gwangju, incorporating demand management systems and smart meters. This implementation has resulted in a 20% reduction in electricity consumption during peak hours, as well as significant savings on electricity costs through the integration of renewable energy [14].

These examples demonstrate how automation and modern technologies can significantly enhance the efficiency of power grids and energy security, which is an important direction for Ukraine in the context of its energy transformation.

Among the implemented automation projects, the introduction of automated electricity commercial accounting systems (ACCS) should be noted. This system optimizes electricity consumption, reduces losses in distribution networks, and ensures more accurate accounting of consumption. Thanks to ACCS, energy companies can manage their resources more effectively, improving the overall stability of the energy system [15].

Automation systems for managing electricity supply modes and lighting have also been implemented, allowing for the optimization of urban electrical networks [16].

Ukraine faces numerous threats in the energy sector, including market monopolization, infrastructure wear, and dependence on energy resource imports. These problems significantly limit opportunities for sustainable development of the energy system. The automation of

power grids is critically important for overcoming these challenges and ensuring reliable energy supply.

III. CONCLUSIONS

Automation of managing elements of power grids to enhance their energy efficiency is a crucial link that affects stability, economic, and human resources.

To address the current issues in Ukraine's energy system, it is advisable to implement several ready-made solutions that can be the most beneficial.

First, smart grids (Smart Grid) can significantly optimize energy consumption and reduce peak loads. The implementation of Smart Grid technologies, such as solutions from ABB Ability, will facilitate the integration of renewable energy sources, which, in turn, will reduce dependence on imported fossil fuels.

Second, substation automation using technologies from companies like Schneider Electric will increase the reliability of energy supply and reduce maintenance costs for infrastructure. This solution will also help lower the likelihood of accidents, which is critically important for the stability of the energy system.

Third, demand response management systems, such as solutions from Oracle Utilities Opower, will allow users to adapt their energy consumption. This will reduce the load on the grid during peak periods, improving the overall stability of the energy system.

The experience of other countries confirms the effectiveness of these technologies. For example, in the U.S., the implementation of Smart Grid has led to a 10-15% reduction in electricity losses and improved supply reliability. Germany, by implementing the "Energy Transition" policy, has achieved a share of renewable sources in electricity production exceeding 50%. In South Korea, the Smart Grid project in Gwangju has resulted in a 20% reduction in electricity consumption during peak loads.

In general, the implementation of these solutions will enhance Ukraine's energy security, improve the stability of energy supply, and reduce energy poverty, creating a favorable environment for investment in this sector.

Therefore, the successful implementation of power grid automation is a key condition for achieving the strategic goals of Ukraine's energy security, as outlined in the Strategy until 2025. This will reduce dependence on imported energy resources, decrease infrastructure wear, and improve energy resource management, which is critically important for the country's economic development.

REFERENCE LIST

[1] EUEA. "Overview of Ukraine's Energy Security Strategy by EUEA" – [Electronic resource] URL: https://euea-energyagency.org/uk/novyny-ta-podiyi/novyny-rynku/oglyad-strategiyi-energetychnoyi-bezpeky-ukrayiny-vid-yeuea/.

[2] ABB. "ABB Ability" – [Electronic resource] URL: https://global.abb/topic/ability/en.

[3] Siemens. "Grid Control" – [Electronic resource] URL: https://www.siemens.com/global/en/products/energy/grid-software/operation/grid-control.html.

[4] Schneider Electric. "EcoStruxure" – [Electronic resource] URL: https://www.se.com/ww/en/work/campaign/innovation/platform.jsp.

[5] GE Vernova. "Grid Solutions" – [Electronic resource] URL: https://www.gevernova.com/grid-solutions/.

[6] Oracle. "Opower Energy Efficiency" – [Electronic resource] URL: https://www.oracle.com/utilities/opower-energy-efficiency/.

[7] Enel X. "Demand Response" – [Electronic resource] URL: https://www.enelx.com/n-a/en/utilities/distributed-energy/demand-response.

[8] Schneider Electric. "Advanced Distribution Management System (ADMS)" – [Electronic resource] URL: https://www.se.com/ww/en/work/solutions/for-business/electric-utilities/advanced-distribution-management-system-adms/.

[9] Siemens Energy. "Omnivise Hybrid Control" – [Electronic resource] URL: https://www.siemens-energy.com/global/en/home/products-services/service/omnivise-hybrid-control.html.

[10] Honeywell. "Energy Control System" – [Electronic resource] URL: https://process.honeywell.com/content/dam/pmt/en/documents/gated/HPS%20BRO%20RDA%20-%20EECS%20A4%20V5.pdf.

[11] AVEVA. "PI System" – [Electronic resource] URL: https://www.aveva.com/en/products/aveva-pi-system/.

[12] "Digitalization of Energy: How Smart Grid Technologies Will Help Rebuild Ukrainian Power Grids," Mind.ua – [Electronic resource] URL: https://mind.ua/publications/20273149-cifrovizaciya-energetiki-yak-tehnologiyi-smart-grid-dopomozhut-vidbuduvati-ukrayinski-energomerezhi.

[13] "Building Smart Grid - A Path to Increasing the Resilience of Ukraine's Energy System," Ukrinform – [Electronic resource] URL: https://www.ukrinform.ua/rubric-economy/3863598-rozbudova-smart-grid-slah-do-pidvisenna-stijkosti-energosistemi-ukraini.html.

[14] "Implementation of Energy-Efficient Technologies in New Construction," Science.lpnu.ua – [Electronic resource] URL: https://science.lpnu.ua/sites/default/files/journal-paper/2020/dec/22823/02-bakhtin.pdf.

[15] "Automated Commercial Electricity Accounting System (ASKOE)" – [Electronic resource] URL: https://grandtesla.com.ua/service/askoe.

[16] DOI. "Analysis of the Prospects for the Development of Distributed Generation Systems in Ukraine" – [Electronic resource] URL: https://doi.org/10.32838/2663-5941/2021.3/3.

Наукове видання

Ігор НЕВЛЮДОВ,
Владислав ЄВСЄЄВ,

VIII Міжнародна Конференція
«Виробництво & Мехатронні Системи»
(укр., англ., пол.. мовою)

Відповідальний редактор – Невлюдов І.Ш.

Харківський національний університет радіоелектроніки
Кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР)
61166, Харків, проспект Науки, 14
корпус "А"
ауд. 162-1
тел .: +38 (057) 702-14-86
e-mail:m_ms@nure.ua

Підписано до друку 10.10.2024
Формат А4 (210х297мм). Папір 80г/м^2 .
[електронній друк]

SCI-CONF.COM.UA

PERSPECTIVES OF CONTEMPORARY SCIENCE: THEORY AND PRACTICE

PROCEEDINGS OF V INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE JUNE 24-26, 2024

LVIV
2024

PERSPECTIVES OF CONTEMPORARY SCIENCE: THEORY AND PRACTICE

Proceedings of V International Scientific and Practical Conference

Lviv, Ukraine

24-26 June 2024

Lviv, Ukraine

2024

UDC 001.1

The 5th International scientific and practical conference "Perspectives of contemporary science: theory and practice" (June 24-26, 2024) SPC "Sci-conf.com.ua", Lviv, Ukraine. 2024. 1310 p.

ISBN 978-966-8219-88-7

The recommended citation for this publication is:
Ivanov I. Analysis of the phaunistic composition of Ukraine // Perspectives of contemporary science: theory and practice. Proceedings of the 5th International scientific and practical conference. SPC "Sci-conf.com.ua". Lviv, Ukraine. 2024. Pp. 21-27. URL: https://sci-conf.com.ua/v-mizhnarodna-naukovo-praktichna-konferentsiya-perspectives-of-contemporary-science-theory-and-practice-24-26-06-2024-lviv-ukrayina-arhiv/.

Editor
Komarytskyy M.L.
Ph.D. in Economics, Associate Professor

Collection of scientific articles published is the scientific and practical publication, which contains scientific articles of students, graduate students, Candidates and Doctors of Sciences, research workers and practitioners from Europe, Ukraine and from neighbouring countries and beyond. The articles contain the study, reflecting the processes and changes in the structure of modern science. The collection of scientific articles is for students, postgraduate students, doctoral candidates, teachers, researchers, practitioners and people interested in the trends of modern science development.

e-mail: lviv@sci-conf.com.ua

homepage: *https://sci-conf.com.ua*

TABLE OF CONTENTS

AGRICULTURAL SCIENCES

VETERINARY SCIENCES

BIOLOGICAL SCIENCES

TECHNICAL SCIENCES

PSYCHOLOGICAL SCIENCES

SOCIOLOGICAL SCIENCES

IMPACT OF AUTOMATION AND CALS TECHNOLOGIES ON HUMAN FACTOR IN PRODUCTION

Sotnik Svitlana Viktorivna,
PhD, associate professor of CITAR department
Hubar Artem Yuriyovych,
student
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine

Introductions. Technological advances in manufacturing necessarily affect entire process chain, from product concept to manufacturing and delivery to consumer. Modern technologies are changing paradigm of manufacturing, affecting automation, data management, analytics, and labor utilization [1-4].

Technologies such as automation, Internet of Things (IoT), artificial intelligence (AI), 3D printing, and other innovations help speed up production, reduce time and costs, and improve product quality [5, 6].

In context of automation, it is important to consider how it affects workflows by reducing manual labor and increasing productivity through use of systems such as robots, CNC machines (Computer Numerical Control) and automatic assembly lines, and CALS (Continuous Acquisition and Life cycle Support) technologies.

Given impact of automation and other innovations in modern manufacturing, CALS technologies play key role in effective management of information at various stages of product life cycle. These technologies include standardized data exchange methods, electronic catalogs, and logistics support systems that allow for collection, updating, and transfer of necessary information quickly and efficiently between all participants in production process. They help to reduce time spent on data preparation, improve accuracy and reduce errors, which in turn contributes to overall efficiency and competitiveness of enterprise. So, topic is relevant.

Aim. The purpose of this article is to study impact of automation and CALS technologies on human factor in production. The work examines role of these technologies in changing work processes, employee skills, and social aspects. It is

planned to identify advantages, challenges and opportunities for improving efficiency and competitiveness of enterprises and to propose strategy to support employees in process of adaptation to new technologies.

The impact of automation and CALS technologies on human factor in production results in increased efficiency, reduced errors, improved product quality, and increased safety. Automation of routine tasks allows employees to focus on more complex and creative aspects of their work, which contributes to overall productivity. CALS technologies integrate information systems, providing access to up-to-date data in real time, which improves coordination and decision-making.

Materials and methods. CALS is used to ensure effective information management across entire product lifecycle, including development, production, and support. The benefits of CALS include standardization of data exchange, which simplifies integration of different systems and facilitates collaboration between departments. This helps to increase efficiency of production processes and reduce information management costs.

The disadvantages of CALS are complexity of implementation and high costs of staff training and infrastructure to support new technologies. In addition, it is necessary to constantly update software and ensure compatibility with existing systems. In terms of impact on people, implementation of CALS may change role of employees, forcing them to adapt to new software interfaces and data exchange standards. This may require additional training and competency development.

The human factor in context of CALS includes impact on employee motivation, readiness for change, and ability to adapt to new technologies. The integration of CALS can affect work processes and require review of organizational structures to make optimal use of these technologies.

Let's take look at impact of CALS technologies on human factor in production:

1. Cooperation and communication:

– CALS allow for uniform exchange of information between all participants in product lifecycle;

– improved communication between customers, suppliers, manufacturers and

service personnel.

2. Education and training of personnel:

– implementation of CALS technologies requires training of employees;

– skills in working with electronic systems and data exchange become key.

3. Reducing errors and improving quality:

– CALS systems help to avoid errors related to "human factor";

– ensure accuracy and standardization of data.

4. Adaptation to changes:

– CALS technologies allow for rapid implementation of new solutions and changes;

– employees must adapt to new processes and tools.

Results and discussion. The impact of automation and CALS technologies on human factor in production is becoming an extremely important aspect today, as these technologies can both increase and change requirements for employees' skills and their opportunities in labor market.

When studying economic and social aspects, it is important to understand how cost of implementing automation and CALS technologies affects society as whole, taking into account benefits and risks associated with these innovations.

Automation or digitalization of labor has given rise to new concept of "digital labor." New digital labor markets claim to provide more flexibility, cost-effectiveness and efficiency for both clients and independent contractors. However, this flexibility is often accompanied by precarious working conditions and violations of established legal and social standards of work quality".

Some workers do not have same access to training resources and retraining opportunities to keep up with rapid technological advances in production. Those with low levels of qualifications and those living in remote or underdeveloped regions may face limited opportunities to acquire new skills and knowledge.

These inequalities can widen digital divide between those who effectively use technology and those who lag behind. Unequal access to training and retraining can deepen social inequalities and lead to increased unemployment and economic

inequality. To reduce these negative effects, it is important to ensure broad access to training programs and retraining initiatives for all categories of workers and regions. In addition, improvements in educational attainment and technological skills should be incentivized to ensure fairer and more equitable access to benefits of technological progress across all sectors of production".

To summarize, technological progress in manufacturing supported by automation and CALS technologies has significant potential to improve efficiency and competitiveness, but requires careful management of social impacts and implementation costs. In course of this analysis, we will provide rough estimates of impact of CALS technologies on human factor in various industries (Fig. 1).

Influence of CALS-technologies on human factor in different branches of production:

1. Automotive industry (40 %) – increasing efficiency of production processes, reducing errors and improving labor safety.
2. Aviation industry (35 %) – improving coordination between different stages of production and maintenance, increasing accuracy and speed of tasks.
3. Electronics manufacturing (30 %) – optimization of complex production processes, reduction of manual labor and increased assembly accuracy.
4. Military industry (25 %) – increasing reliability and quality of products, improving logistics and service.
5. Pharmaceutical industry (20 %) – ensuring accuracy and reliability in production and quality control processes, reducing risks to employee health.

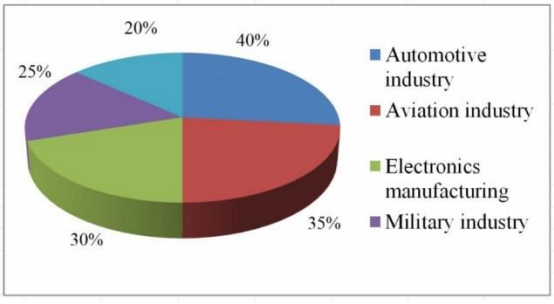


Fig. 1. Influence of CALS-technologies on human factor in different branches of production

The study highlighted impact of CALS-technologies on human factor in manufacturing. The impact of automation and CALS technologies on human factor in production can significantly improve product quality and avoid errors, but at same time requires restructuring. This means that employees must not only have relevant knowledge, but also be mentally prepared for change, motivated and willing to adapt and learn new things. To support such employees, it is important to provide access to variety of training resources and retraining opportunities. Here are some examples that can help employees reorient without significant moral discomfort:

1. Customized training programs, meaning that customized courses need to be developed that take into account employees' current skills and experience, which will then allow employees to learn at their own pace, reducing stress and increasing learning success.

2. Mentoring support, which means that mentoring programs should be created where experienced colleagues can help newcomers adapt to new technologies and workflows, and help them adapt to change more quickly and effectively.

3. Online courses and webinars will allow employees to learn new material at time and place that is convenient for them (as long as they have access to online resources), and this increases learning flexibility and reduces feeling of being locked in.

4. Hands-on training and workshops, this facilitates faster learning and professional competence.

5. Supporting social and psychological well-being can be achieved by creating specific support programs that include psychological support, professional development counseling with opportunities to discuss any career change issues.

These measures will help to make retraining process less stressful and more productive for employees, which, in turn, will help to improve production efficiency and overall competitiveness of enterprise.

Conclusions. The study summarized problem of influence of automation and CALS-technologies on human factor in production. The main aspects of this influence in different industries were considered. The work identified following key

points:

– impact of CALS technologies varies by industry, with greatest impact in automotive (40 %), aviation (35 %) and electronics (30 %) industries;

– CALS technologies help improve communication, reduce errors, and improve product quality;

– implementation of these technologies requires continuous training and development of workers.

A strategy to support workers in adapting to new technologies is proposed, including customized training programs, mentoring support, online courses, and hands-on training. This study contributes to better understanding of complex impact of CALS-technologies on human factor in production and emphasizes need for balanced approach to implementation of these technologies, taking into account both technological and social aspects. This study can serve as basis for further study of optimization of CALS-technology implementation processes in various branches of production.

REFERENCES:

1. Sotnik, S. V. Design features of control panels and consoles in automation systems / S. V. Sotnik et al. // 9th International scientific and practical conference “Science and innovation of modern world” (May 18-20, 2023) Cognum Publishing House, London, United Kingdom, 2023. – P. 201 – 205.

2. Sotnik, S. V. Analysis of design process of automated fire protection system / S.V. Sotnik et al. // V Форум “Автоматизація, електроніка та робототехніка” (AERT-2023). – 2023. – С. 59-62.

3. Зарубін, І. С. Ефективність використання роботизованих систем у виробництві / І. С. Зарубін та ін. // Комп’ютерно-інтегрованих технологій, автоматизації та робототехніки 2024: матеріали І-ої Всеукраїнської конференції, Харків, 16-17 травня 2024 (CITAR-2024). – 2024. – С. 150-153.

4. Nevludov, I. S. Cloud giants: AWS, Azure and GCP / I.S. Nevludov et al. // 2023 2nd International Conference on Innovative Solutions in Software Engineering Ivano-Frankivsk, Ukraine, November 29-30. – 2023. – С. 18-23.

5. Baker, J.H. Some interesting features of semantic model in Robotic Science / J.H. Baker, V. et al. // International Journal of Engineering Trends and Technologythis link is disabled. – 2021. – 69(7). – P. 38-44.

6. Mohammad, A.S.Y. Neural networks as a tool for pattern recognition of fasteners / A.S.Y. Mohammad et al. // International Journal of Engineering Trends and Technologythis link is disabled. – 2021. – 69(10). – C. 151-160.

EUROPEAN CONFERENCE

Conference Proceedings

XXIV International Science Conference
«Modern technologies among us in the environment»

June 17-19, 2024

Rome, Italy

MODERN TECHNOLOGIES AMONG US IN THE ENVIRONMENT

Abstracts of XXIV International Scientific and Practical Conference

Rome, Italy
(June 17-19, 2024)

UDC 01.1

ISBN – 9-789-40372-430-0

The XXIV International Scientific and Practical Conference «Modern technologies among us in the environment», June 17-19, 2024, Rome, Italy. 419 p.

The recommended citation for this publication is: Yakovenko R., Labunets V. Features of mineral nutrition of intensive apple plantations. Abstracts of XXIV International Scientific and Practical Conference. Rome, Italy. Pp. 14-15.

URL: https://eu-conf.com/en/events/modern-technologies-among-us-in-the-environment/

TABLE OF CONTENTS

ВИКОРИСТАННЯ ВЕБ-ДОДАТКІВ ДЛЯ МОНІТОРІНГУ ТА УПРАВЛІННЯ

Губарь Артем Юрійович
здобувач другого магістерського рівня
факультету автоматики і комп'ютеризованих технологій
Харківський національний університет радіоелектроніки

Митцева Ольга Сергіївна
к.пед.н., доцент кафедри філософії
Харківський національний університет радіоелектроніки

Веб-додатки – це унікальні програми, які працюють в браузері і відрізняються від звичайних сайтів. Вони вимагають більш складних налаштувань, оскільки користувач тут – активний учасник процесу, а не просто споживач контенту. Веб-додатки складаються з двох основних частин: клієнтської, яка відображається в браузері користувача, та серверної, яка обробляє запити і зберігає дані. Вони широко використовуються в різних галузях, включаючи моніторинг та управління, для підвищення продуктивності та ефективності.

Зараз автоматизація все більше проникає в робочі місця, і її вплив на робочі практики та ролі є далекосяжним. Робочі завдання, як правило, автоматизуються з урахуванням ефективності, результативності та безпеки, але менше уваги приділяється аспектам користувацького досвіду. Однак повністю автономні та безлюдні системи рідкісні, оскільки люди часто все ще повинні контролювати, втручатися, обслуговувати та контролювати автоматизовані середовища.[2]

Цифрова автоматизація проникла в багато сфер нашого повсякденного життя, маючи серйозні наслідки для соціальних, економічних та політичних систем. Здатність автоматизації постійно покращувати та трансформувати людське життя породила широке тіло наукових досліджень з внесками від соціальних та економічних наук, інженерії та технологій. [3]

Існує багато веб-додатків, які спрямовані на автоматизацію роботи та моніторинг задач або ресурсів на підприємстві. Jira та Trello – це популярні інструменти для управління проектами, які дозволяють командам ефективно відстежувати прогрес задач. Sales Creatio, Worksection, KeyCRM, SalesDrive та HugeProfit – це веб-додатки, які допомагають автоматизувати різні аспекти бізнесу, включаючи продажі, управління проектами та облік.[4 – 7]

Крім того, існують веб-додатки, які допомагають автоматизувати розрахунок вартості економічних рішень. Наприклад, EBITDA калькулятор дозволяють користувачам легко розраховувати вартість різних економічних рішень. Frontmen та IWIS пропонують розробку веб-додатків, які можуть автоматизувати такі розрахунки, що може бути особливо корисним для підприємств, які шукають способи оптимізації своїх процесів.

Неможна забувати що існують більш профільні завдання, що вирішуються окремими локалізованими додатками і це все збільшу ефективність робочої праці за менший проміжок часу.

Переваги:

- Ефективність – веб-додатки дозволяють автоматизувати рутинні задачі, що звільняє час співробітників для виконання більш складних завдань;
- Централізація – вони забезпечують централізований доступ до інформації, що полегшує управління проектами та ресурсами;
- Моніторинг – вони дозволяють відстежувати прогрес задач та стан ресурсів в реальному часі;
- Співпраця – вони полегшують співпрацю між командами, незалежно від їх географічного розташування.

Недоліки:

- Вартість – деякі веб-додатки можуть бути дорогими, особливо для малого та середнього бізнесу;
- Навчання – співробітникам може знадобитися час, щоб навчитися користуватися новими інструментами;
- Залежність від Інтернету – веб-додатки залежать від Інтернету, тому будь-які проблеми з підключенням можуть призвести до перерви в роботі.[8 – 9]

Кожен веб-додаток має свою специфіку та направленість, і він може бути оптимізований для вирішення певного кола завдань. Це може бути великою перевагою, коли вам потрібно вирішити дуже специфічну проблему, але це також може стати обмеженням, коли вам потрібно вирішити завдання, які виходять за рамки цієї специфіки.

Щодо похибок, які не враховуються вже існуючими веб-додатками, це є загальним викликом для всіх типів програмного забезпечення. Неможливо передбачити всі можливі сценарії, які можуть виникнути під час використання додатка, особливо коли мова йде про складні системи, такі як веб-додатки для моніторингу та управління. Однак, розробники постійно працюють над виявленням та виправленням похибок, а також над вдосконаленням своїх продуктів на основі відгуків користувачів.

Використання веб-додатків для моніторінгу та управління може значно підвищити продуктивність та ефективність роботи. Вони дозволяють автоматизувати рутинні задачі, централізувати доступ до інформації, відстежувати прогрес задач та стан ресурсів в реальному часі, а також полегшують співпрацю між командами. Однак, важливо пам’ятати про потенційні виклики, такі як вартість веб-додатків, необхідність навчання співробітників, ризик витоку або втрати даних, а також залежність від Інтернету. Незважаючи на ці виклики, веб-додатки продовжують революціонізувати сучасний бізнес, пропонуючи нові можливості для оптимізації та автоматизації процесів. Вони відкривають нові горизонти для підвищення продуктивності та ефективності, що робить їх незамінним інструментом в сучасному світі.

Отже, автоматизація може мати значний вплив на продуктивність та задоволеність персоналу. Тому, при впровадженні автоматизації, важливо не

тільки зосередитися на прагматичних аспектах, таких як ефективність та результативність, але також врахувати емоційні аспекти користувацького досвіду. Це може включати розробку більш інтуїтивних інтерфейсів, забезпечення зворотного зв'язку та створення більш приємного користувацького досвіду.

Список літератури

1. Engaging Automation at Work / 5th IFIP WG 13.6 Working Conference, HWID 2018, Espoo, Finland, August 20 – 21, 2018, Revised Selected Papers
2. Özkiziltan, Didem and Hassel, Anke, Humans versus Machines: An Overview of Research on the Effects of Automation of Work (August 8, 2020). 36 Pages papers.ssrn.com/sol3/papers.cfm?abstract_id=3789992
3. Чому вашій компаніїварто обрати Worksection. Електронний ресурс. Режим доступу: https://worksection.com/ua/why_worksection.html
4. Компонована платформа для управління продажами / Sales Creatio. Електронний ресурс. Режим доступу: https://www.creatio.com/ua/sales
5. KeyCRM – CRM система для бизнеса / Украинская CPM система. Електронний ресурс. Режим доступу: https://ua.keycrm.app/
6. CRM система для товарного бізнесу, простий облік фінансів та товарів / HugeProfit. Електронний ресурс. Режим доступу: https://h–profit.com/ua/
7. Pros and Cons of Jira Software for Project Management / By Lauren Good Published June 23, 2023. Електронний ресурс. Режим доступу: https://project–management.com/the–pros–and–cons–of–using–jira–software/
8. Main Advantages & Disadvantages of Web Apps in 2024 / Make IT Simple. February 29, 2024. Електронний ресурс. Режим доступу: https://www.makeitsimple.co.uk/blog/web–app–advantages–disadvantages
9. Mariia Kulyk, Olha Myttseva Role of Web Application Security in the Modern Educational Process at Higher Education Institutions V International Scientific and Practical Conference Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs (MC&FPGA-2023), Kharkiv, Ukraine, 2023, p. 52. DOI: 10.35598/mcfpga.2023.018

Scientific publications

MATERIALS

The XXIV International Scientific and Practical Conference
«Modern technologies among us in the environment»

Rome, Italy. 419 p.

(June 17-19, 2024)

Міністерство освіти і науки України

NURE

Харківський національний університет радіоелектроніки

ЗБІРНИК

студентських наукових статей

«Автоматизація та приладобудування»

«Automation and Development of Electronic Devices»

ADED-2023

(Випуск 2)

[електронне видання]

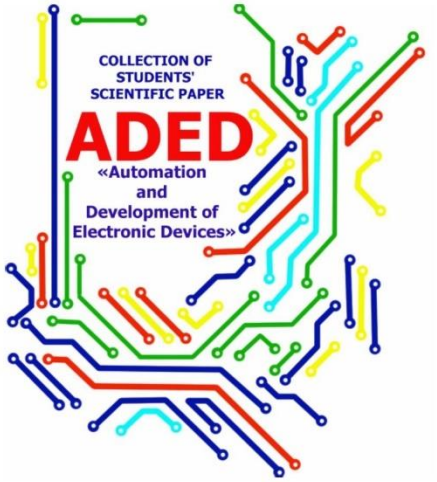
http://nure.ua/department/kafedra-komp-yuterno-integrovanih-tehnologiy-avtomatizatsiyi-ta-mehatroniki-kitam

http://itez.zntu.edu.ua/

http://kafea.kdu.edu.ua

Харків 2023

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки
кафедра комп'ютерно-інтегрованих технологій, автоматизації та робототехніки
(КІТАР)



ЗБІРНИК

студентських наукових статей

«Автоматизація та приладобудування»

«Automation and Development of Electronic Devices»

ADED-2023

(Випуск 2)

[електронне видання]

Харків 2023

Автоматизація та Приладобудування («Automation and Development of Electronic Devices» ADED-2023) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки ; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2023. – Вип. 2. – 408с.

Collection of Students' Scientific Paper «Automation and Development Of Electronic Devices» ADED-2023 Part 2 (Key infrastructure 2023) - Kharkiv/ The Editorial.: Nevlyudov I.Sh. (head), that all. Kharkiv: Kind of Kharkiv National Universitety of Radio Elektronics [electronic edition], 2023. – 408p with.

Рекомендовано рішенням
Науково-технічної ради
Харківського національного
університету радіоелектроніки
протокол №6 від 29.11.2018

Рекомендовано рішенням Вченої ради
факультету Автоматики і комп'ютеризованих технологій
Харківського національного
університету радіоелектроніки
протокол № 4 від 30.11.2023

Збірник містить наукові статті здобувачів першого (бакалаврського), другого (магістерського) рівнів вищої освіти кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР) Харківського національного університету радіоелектроніки, кафедри Інформаційних технологій електронних засобів (ІТЕД) Запорізького національного технічного університету та кафедри Електронних апаратів (ЕА) Кременчуцького національного університету ім. М. Остроградського які навчаються за спеціальностями: 151 Автоматизація та комп'ютерно-інтегровані технології, 172 Телекомунікації та радіотехніка, 171 Електроніка та 163 Біомедична інженерія. Статті надані в авторській редакції.

ЗМІСТ

УДК 519

ВЕБ-ДОДАТОК ДЛЯ МОНІТОРИНГУ ТА УПРАВЛІННЯ ЗАПАСАМИ В 3D-ДРУКАРНІ

А.Ю. Губарь
Харківський національний університет радіоелектроніки
Україна, 61166, Харків, пр. Науки 14
Email: artem.hubar@nure.ua.

Анотація: Стаття присвячена принципам та моделям автоматизації управління запасами на в прикладі 3D-друкарні. В рамках проекту виконано аналіз попиту, прогнозування, відстеження робочого часу працівників, аналіз витрат та вартості виробництва. Отримані результати сприятимуть оптимізації виробництва та забезпеченню ефективного управління ресурсами в умовах сучасних воєнних реалій.

Ключові слова: моніторинг, управління запасами, 3D-друкарня, прогнозування, аналіз витрат, оптимізація виробництва.

WEB APPLICATION FOR MONITORING AND INVENTORY MANAGEMENT IN A 3D PRINTER

A.Y. Hubar
Kharkiv National University of Radio Electronics
Ukraine, 61166, Kharkiv, Nauky av.,14
Email: artem.hubar@nure.ua.

Anotations: This article is devoted to the principles and models of automation of inventory management in the example of a 3D printing house. The project includes the implementation of demand analysis, forecasting, tracking employees' working hours, and analyzing costs and production expenses. The obtained results will contribute to optimizing production and ensuring efficient resource management in the conditions of contemporary wartime realities.

Key words: monitoring, inventory management, 3D printing, forecasting, cost analysis, production optimization.

Предметом даної роботи є розробка веб-додатка для моніторингу та управління запасами в 3D-друкарні. Запропонований додаток спрямований на створення зручного інтерфейсу для відстеження різноманітних матеріалів і деталей, що використовуються у виробництві 3D-виробів. Для досягнення цієї мети планується використання бази даних для зберігання інформації про наявні ресурси та їх використання.

Запропонована тема стосується не тільки інновацій у сфері виробництва та веб-технологій, але й знаходить свою актуальність в умовах повномасштабної війни в Україні. Зокрема, у звіті Deutsche Welle відзначено, що технології 3D-друку стають важливим інструментом для задоволення потреб у пластикових деталях для військових потреб [1].

Українські волонтери та друкарі активно використовують 3D-друкарні для виготовлення різноманітних деталей, таких як хвостовики для снарядів, кріплення для дронів, оболонки для снарядів, інструменти для розмінування протитанкових мін тощо. Ці деталі стають важливими компонентами на фронті, допомагаючи українським військовим забезпечувати свої потреби в умовах воєнного конфлікту.

Зазначається, що технології 3D-друку стають частиною креативної децентралізованої моделі виробництва. Це особливо важливо в умовах війни, де імпровізовані рішення та невеликі виробничі групи можуть швидко реагувати на потреби фронту [1].

Підсумково, в контексті війни в Україні, розробка веб-додатка для моніторингу та управління запасами в 3D-друкарні виявляється важливою ініціативою. Вона допомагає забезпечувати армію необхідними ресурсами, зокрема, пластиковими деталями, які можуть бути вироблені шляхом 3D-друку. Такий підхід є актуальним та ефективним у контексті сучасних воєнних реалій, забезпечуючи важливі матеріальні ресурси для підтримки оборонної справи.

База даних у розробленому веб-додатку є ключовим компонентом, оскільки вона забезпечує зберігання структурованої інформації про матеріали, деталі та інші ресурси. Використання бази даних дозволяє ефективно організувати дані, забезпечуючи швидкий доступ до необхідної інформації та спрощуючи процеси аналізу.

Основними сутностями бази даних є матеріали, деталі, працівники, витрати, замовлення.

Кожен запис у базі даних відображає конкретну одиницю матеріалу чи деталі, забезпечуючи повноту та точність обліку запасів. Використовуючи реляційні бази даних створюються різні зв’язки між сутностями такі як один до одного, багато до одного та багато до багатьох.

Додатково, використання бази даних дозволяє впровадження різноманітних функцій, таких як генерація звітів, аналіз використання ресурсів та автоматизація процесів оновлення інформації. Це сприяє ефективному використанню матеріальних ресурсів та максимізації продуктивності виробництва.

Використання бази даних в даному контексті є стратегічно важливим елементом, оскільки вона допомагає підприємствам не лише відстежувати запаси, але і приймати інформовані рішення з управління ресурсами. Надалі можливо розглядати можливість розширення функціоналу бази даних, включаючи інтеграцію з іншими системами, що дозволить подальше покращення управління запасами в 3D-друкарнях [2].

Важливо визначити конкретні технології баз даних, які застосовуються для забезпечення ефективного функціонування веб-додатку. Однією з важливих технологій є реляційні бази даних RDBMS (*Relational Database* Management System), які дозволяють структурувати дані у вигляді таблиць, зв'язаних за допомогою ключів. Використання RDBMS сприяє ефективному зберіганню та обробці інформації.

Крім того, мова запитів SQL (Structured Query Language) використовується для взаємодії з базою даних. SQL дозволяє виконувати різноманітні запити, забезпечуючи необхідний доступ до даних для користувачів та адміністраторів системи.

У контексті великого обсягу даних, використання технологій кластеризації баз даних може забезпечити високий рівень доступності та надійності системи. Такі кластери дозволяють розподілити навантаження між різними серверами, забезпечуючи оптимальну продуктивність.

Системи управління версіями баз даних, наприклад Git, можуть сприяти збереженню історії змін та відстеженню версій схем даних. Це важливо для забезпечення цілісності та відновлення даних у випадку необхідності.

У веб-додатку для моніторингу та управління запасами в 3D-друкарні використовуються передові методи шифрування, спрямовані на забезпечення максимального рівня конфіденційності та безпеки оброблюваної інформації. Це важливий аспект, особливо при роботі з конфіденційними даними щодо ресурсів, матеріалів та інших важливих показників виробництва.

Методи шифрування можуть бути використані для захисту конфіденційної інформації в базі даних, забезпечуючи безпеку даних користувачів та деталей про виробництво. Використання алгоритмів шифрування дозволяє захистити дані під час передачі та зберігання.

З основних методів – це TLS/SSL шифрування, AES (Advanced Encryption Standard), хешування паролів, методи аутентифікації та авторизації.

Інтеграція цих методів шифрування у веб-додатку забезпечує повний спектр заходів для захисту від потенційних загроз та надає користувачам впевненість у конфіденційності та безпеці їхніх даних.

Технології резервного копіювання та відновлення даних є невід'ємною частиною системи. Регулярне створення резервних копій бази даних дозволяє запобігти втраті інформації та забезпечити можливість відновлення в разі аварій.

Також слід розглянути використання систем моніторингу та збереження даних, які дозволяють вчасно виявляти та реагувати на будь-які проблеми у роботі бази даних.

У веб-додатку для моніторингу та управління запасами в 3D-друкарні використовується продумана система моніторингу та збереження даних, що дозволяє вчасно виявляти та ефективно реагувати на будь-які проблеми у роботі бази даних. Ці заходи спрямовані на забезпечення стабільності та безпеки функціонування системи.

З основних методів – це моніторинг ресурсів бази даних, класифікація запитів, система автоматичного сповіщення, моніторинг резервного копіювання, аналіз логів та виявлення несанкціонованого доступу, заходи захисту від SQL-ін'єкцій та інших атак.

Ці заходи формують інтегровану систему моніторингу та збереження даних, яка забезпечує вчасне виявлення та ефективну реакцію на будь-які проблеми у роботі бази даних, забезпечуючи надійність та безпеку функціонування веб-додатка.

Вищезгадані технології в сукупності створюють потужний інструментарій для забезпечення ефективного та безпечного ведення обліку ресурсів у 3D-друкарні.

Методологія, використана в даній роботі, включає детальний огляд існуючих систем управління запасами та використання сучасних технологій для створення веб-додатка. Детально проаналізовані переваги та недоліки, функціональні можливості та обмеження існуючих систем. Цей огляд слугує основою для вибору оптимального шляху в розробці веб-додатка.

Робота акцентує на використанні передових технологій для створення веб-додатка. Розглядається використання сучасних мов програмування, фреймворків та інструментів розробки, що гарантує ефективність та стабільність роботи додатка.

Створений веб сервіс включає в себе автоматизацію рутинних завдань, що сприяє ефективному використанню робочого часу.

Очікується, що результати роботи матимуть вагомий вклад у поліпшення управління запасами в 3D-друкарнях. Розроблений веб-додаток покликаний забезпечити ефективність та точність обліку ресурсів, що дозволить підприємствам, зайнятим у виробництві 3D-виробів, оптимізувати свою діяльність.

Одним із важливих аспектів розробки додатка є аналіз попиту та прогнозування. Використовуючи дані з бази даних, система здатна аналізувати історичні та поточні дані про попит на конкретні матеріали чи деталі, а також прогнозувати майбутні потреби. Це дозволяє підприємству більш точно планувати виробництво та управляти запасами.

Важливо також враховувати ефективне відстеження обліку працівників у виробничому процесі. Додаток має забезпечити можливість реєстрації працівників, їхніх функціональних обов'язків та робочого часу. Це дозволить ефективно впроваджувати аналіз витрат праці та визначати оптимальні стратегії використання робочого часу.

Аналіз витрат та аналіз вартості виробництва є невід'ємними складовими процесу управління запасами. Додаток має враховувати витрати на матеріали, працю та інші ресурси для аналізу загальних витрат та оптимізації вартості виробництва.

$$Ww = Wm + Wd. \quad (1)$$

де Ww − вартість виробництва; Wm – вартість матеріалів; Wd – вартість друку.

$$Wd = Hd * Kwd + Wn. \quad (2)$$

де Hd − час друку; Kwd − коефіцієнт друку;тWn − вартість налаштування.

Аналіз запасів є ключовим етапом у впровадженні ефективної системи управління запасами на підприємстві. Систематичний огляд наявних ресурсів дозволяє виявити оптимальні стратегії їх використання та забезпечення найвищої продуктивності. В ході аналізу слід враховувати фактори, такі як швидкість обороту запасів, попит на конкретні матеріали та деталі, а також можливі ризики пов'язані з недостачею або перевищенням запасів.

$$Pt = \frac{\sum M}{Km}. \quad (3)$$

де Pt – попит запасів за рік; M − одиниця матеріалу за останній рік; Km – кількість місяців роботи.

$$Pz = Pt + Pzw - Zo. \quad (4)$$

де Pz – потреба запасів; Pzw − потреба в утворенні нормативних залишків на кінець періоду; Zo − запаси на початок періоду.

Детальний огляд запасів також включає в себе визначення оптимальних рівнів запасів, щоб уникнути втрат від зайвих запасів або можливостей необхідних матеріалів. Крім того, важливо враховувати сезонні коливання, щоб забезпечити вчасне поповнення запасів перед періодами підвищеного попиту.

Таким чином, розроблений веб-додаток не лише спростить відстеження запасів у 3D-друкарні, але й забезпечить комплексний аналіз та управління різними аспектами виробничого процесу.

Узагальнюючи, аналіз запасів дозволяє підприємству приймати інформовані рішення щодо управління своїми ресурсами, що є важливим елементом стратегії оптимізації виробництва та забезпечення якісного обслуговування клієнтів.

Для подальшого вдосконалення системи можна розглянути можливість інтеграції інтелектуальних алгоритмів аналізу попиту та прогнозування запасів. Це дозволить більш точно адаптувати запаси до змінних вимог виробництва та покупців.

Також важливим етапом розвитку може стати дослідження можливостей забезпечення безпеки даних користувачів та конфіденційності інформації. Розробка та впровадження ефективних заходів забезпечення кібербезпеки буде ключовим чинником для успішного функціонування веб-додатка, особливо при роботі з конфіденційною інформацією та персональними даними.

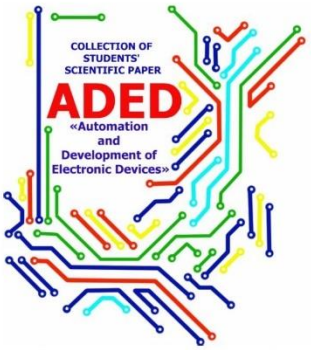
У зв'язку з ростом важливості електронної комерції в Україні, слід продовжити вивчення та впровадження новітніх технологій для підтримки онлайн-покупок. Можливість інтеграції

системи з платіжними платформами та сервісами доставки може значно полегшити процес замовлення та отримання 3D-виробів для клієнтів.

ЛІТЕРАТУРА

1. Армія друкарів: як Україна застосовує 3D-технології у війні. https://www.dw.com/uk/armia-drukariv-ak-ukraina-zastosovue-3dtehnologii-u-vijni/a-67165996
2. Основи системного підходу щодо створення інформаційних систем. https://buklib.net/books/23562/.

Науковий керівник: *Сезонова Ірина Костянтинівна, професор кафедри КІТАР*



[електронне видання]

Відповідальний редактор: д.т.н., проф. Євсєєв В.В., к.т.н., доц. Демська Н.П.

Рекомендовано рішенням Науково-технічної ради
Харківського національного університету радіоелектроніки
протокол №6 від 29.11.2018

Рекомендовано рішенням Вченої ради
факультету Автоматики і комп'ютеризованих технологій
Харківського національного
університету радіоелектроніки
протокол № 4 від 30.11.2023

АВТОМАТИЗАЦІЯ ТА ПРИЛАДОБУДУВАННЯ («Automation and Development of Electronic Devices» ADED-2023) [Електронний ресурс] : збірник студентських наукових статей / Харківський національний університет радіоелектроніки; [редкол.: І.Ш. Невлюдов та ін.]. – Харків : ХНУРЕ, 2023. – Вип. 2. – 408с.

COLLECTION OF STUDENTS' SCIENTIFIC PAPER «AUTOMATION AND DEVELOPMENT OF ELECTRONIC DEVICES» ADED-2023 Part 1 (Key infrastructure 2023) – Kharkiv/ The Editorial.: Nevlyudov I. (head), that all. Kharkiv: Kind of Kharkiv National Universitety of Radio Elektronics [electronic edition], 2023.– 408p. with.

Збірник містить наукові статті здобувачів першого (бакалаврського), другого (магістерського) рівнів вищої освіти кафедри комп'ютерно-інтегрованих технологій, автоматизації та робототехніки (КІТАР) Харківського національного університету радіоелектроніки, кафедри Інформаційних технологій електронних засобів (ІТЕД) Запорізького національного технічного університету та кафедри Електронних апаратів (ЕА) Кременчуцького національного університету ім. М. Остроградського які навчаються за спеціальностями: 151 Автоматизація та комп'ютерно-інтегровані технології, 172 Телекомунікації та радіотехніка, 171 Електроніка та 163 Біомедична інженерія. Статті надані в авторській редакції.

ДОДАТОК Б ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ

МІНІСТЕРСТВООСВІТИІ НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

РОБОТА НА ТЕМУ:

«РОЗРОБЛЕННЯ СИСТЕМИ УПРАВЛІННЯ ПРИСТРОЯМИ ЕЛЕКТРИЧНИХ МЕРЕЖ ПРОМИСЛОВОГО ПІДПРИЄМТСВА»

Виконав:
студент гр. КІТПВм-23-2
Губарь А.Ю.

Керівник роботи:
доцент
Сезонова І.К.

Мета та об'єкт дослідження

1

- Об'єкт дослідження – архітектура та пристрої електромереж.
- Мета роботи – автоматизація управління роботою елементів електромереж з метою підвищення їх енергоефективності.
- Предмет дослідження – процес управління елементами електромереж.
- Методи дослідження – аналіз та синтез, моделювання, математичні методи, програмне моделювання.

Постановка задачі

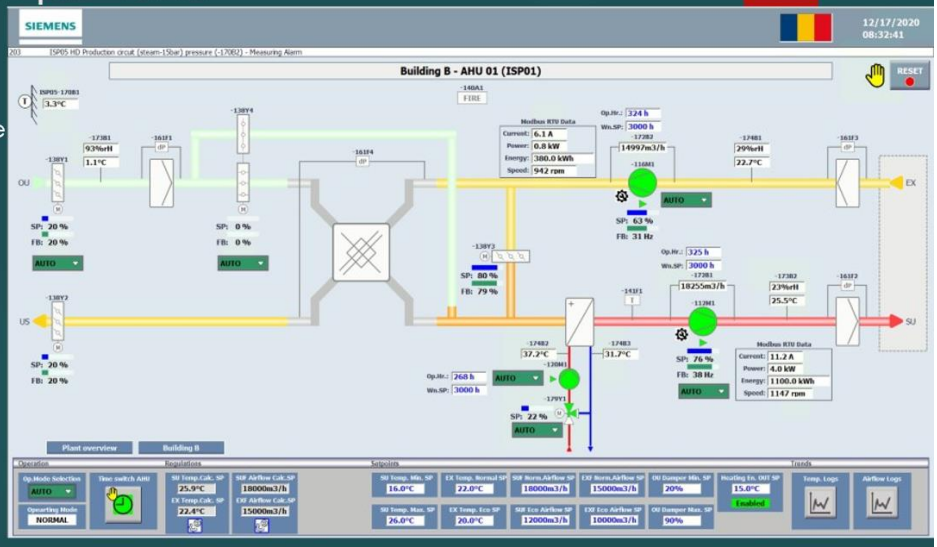
2

- провести аналіз сучасних технологій автоматизації елементів електромереж;
- розробити математичні моделі, що описують вплив параметрів автоматизації на енергоефективність електромереж;
- виконати моделювання системи автоматизації, яка керує показниками роботи електромережі промислового підприємства, включаючи надійність, стабільність та мінімізацію енергетичних витрат.

Аналіз готових рішень

3

- SIMATIC WinCC
- Schneider Electric EcoStruxure
- GE Digital iFIX



Стратегія розв'язання ключових проблем

4

Системи автоматичного керування потребують більш гнучкого масштабування а також узгодження елементів системи на різних рівнях. Цей попит змушує створювати більш гнучкі системи, а саме інтеграції різних smart девайсів з різними параметрами. Для цього пропонується інтегрувати систему тригерів, нижче представлено математичну модель:

$$if\left(D_{parm}\right), \text{то виконати } A$$

де - D_{parm} - параметр девайсу,
A – дія.

Мова програмування та середовище розробки

5

Перелік використаних програмних засобів:
- Docker
- Draw.io.

Середовище розробки:
- IntelliJ IDEA Community Edition 2024.1.4
- pgAdmin
- SceneBuilder
- Node-RED

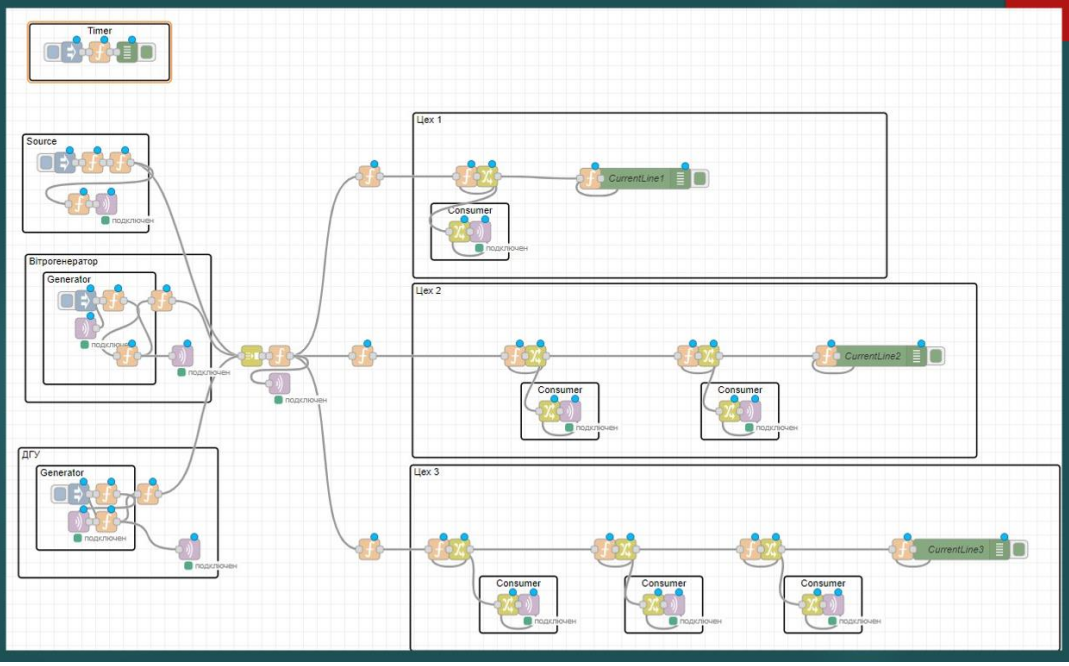
Мова програмування:
- Java 21+;
- JavaScript;
- SQL;

Додаткові технології:
- InfluxDB — база даних для обробки часових рядів.
- EMQX — брокер MQTT для роботи з повідомленнями в IoT.
- JavaFX — фреймворк для розробки графічного інтерфейсу Java-додатків.

Технічне забезпечення: ПК з процесором Core i3 і вище.

Прототип електросистеми

6



Організація області імен для MQTT-брокера

7

Для опису взаємодії з пристроїв, які керуватимуться системою, була обрана наступна структура:

$$client/\{device\_type\}/\{device\_name\}/data$$

де client - ім'я яке служить для об'єднання всіх повідомлень користувача, відокремлюючи їх від системних,

device_type – тип пристрою (наприклад, сенсор). Дозволяє логічно розділяти пристрої,

device_name – унікальне ім'я пристрою. Визначає пристрій серед решти того самого типу,

data – набір даних, містить повідомлення з пристроїв, повідомлення мають різну структуру залежно від типу пристрою.

Така структура дозволяє мати систему, що розширюється і масштабується.

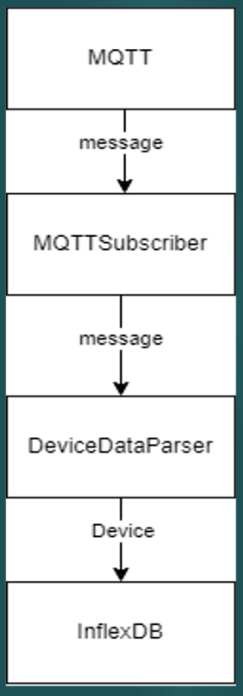
8

Структурна схема архітектури системи управління елементами електромереж промислового підприємства

9

Модуль моніторингу

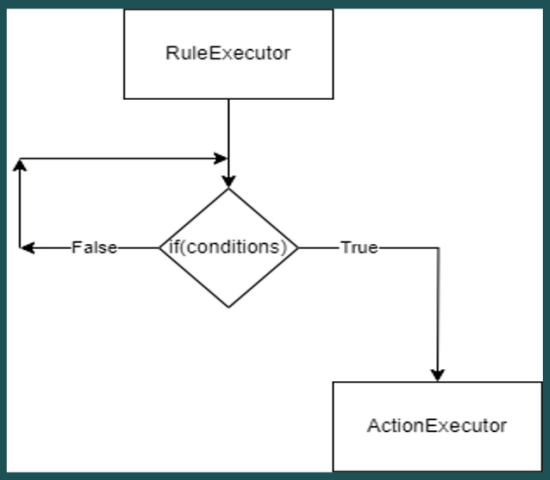
MQTT-брокер посилає повідомлення в MQTTSubscriber. Після повідомлення передається компонент DeviceDataParser, який аналізує дані з повідомлення для створення об'єкта Device, створений об'єкт зберігається в базі даних для подальшого аналізу.



Модуль керування

10

1) правило передається до компонента RuleExecutor;
2) правила містять умови (наприклад , перевірка значення окремого датчика у визначеному діапазоні) та дії, які необхідно виконати;
3) RuleExecutor постійно отримує з бази даних актуальні дані з датчиків, після чого здійснює перевірку умов, зазначених у правилах;
4) у разі виконання умов, компонент ActionExecutor виконує дії, визначені у правилах .



Висновки

11

- проведено огляд аналогічних готових рішень;
- проаналізовано переваги та недоліки таких систем ;
- представлена стратегія розв'язання ключових проблем на основі аналізу конкурентів;
- визначено системні вимоги до розробки системи управління електроелементами , обрано мову програмування, середовище та допоміжні технології;
- для тестування системи, створено прототип електросистеми ;
- розроблено структурну схему архітектури системи управління елементами електромереж промислового підприємства
- розроблено структурну схему модуля моніторингу.
- розроблено структурну схему модуля керування .
- розроблено програмне забезпечення для автоматичного управління електроелементами на підприємстві .
- підведено підсумки виконаної роботи.
- **було опубліковано статтю у «Виробництво &Мехатронні Системи 2024», яка стосується теми моєї кваліфікаційної роботи.**

Дякую за увагу

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦИОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

РОБОТА НА ТЕМУ:

«РОЗРОБЛЕННЯ СИСТЕМИ УПРАВЛІННЯ ПРИСТРОЯМИ ЕЛЕКТРИЧНИХ МЕРЕЖ ПРОМИСЛОВОГО ПІДПРИЄМТСВА»

Виконав:
студент гр. АКТСІу -20-1
Губарь А.Ю.

Керівник роботи:
доцент
Сезонова І.К.

ДОДАТОК В Лістинг програми

```
package org.example.client.controllers.expression.object;

import javafx.fxml.FXML;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.text.Text;
import lombok.Getter;
import org.example.client.factory.DeviceUIFactory;
import org.example.entity.Device;

import javafx.scene.control.Button;

@Getter
public class DeviceController extends ExpressionObjectController {
    private final Device device;
    @FXML
    private Label labelClass;
    @FXML
    private Label labelName;

    @FXML
    private Text deviceId;

    @FXML
    private Button deleteButton;
    @FXML
    private ComboBox<String> methodBox;

    public DeviceController(Object device) {
        this.device = (Device) device;
    }


    @Override
    public Object getObject() {
        return device;
    }

    @FXML
    public void initialize() {
        initLabelName();
        initLabelClass();
        deviceId.setText(device.getSensorId());
        DeviceUIFactory deviceUIFactory = new DeviceUIFactory();
        Device deviceUi = deviceUIFactory.getDeviceUi(device);
        for (var method : deviceUi.getClass().getDeclaredMethods()) {
            methodBox.getItems().add(method.getName());
        }
    }

    private void initLabelClass() {
        labelClass.setText(getDeviceClassName());
    }

    private void initLabelName() {
        if (isDeviceTitleNotEmpty()){
            labelName.setText(device.getTitle());
        }
    }
```

```
    private boolean isDeviceTitleNotEmpty() {
        return device.getTitle() != null && !device.getTitle().isEmpty();
    }

    private String getDeviceClassName() {
        String[] devicePath = device.getClass().getName().split("\\.");
        return devicePath[devicePath.length-1];
    }

    public void setMethod(String methodString) {
        this.methodBox.setValue(methodString);
    }

    @Override
    public String getMethodString() {
        return methodBox.getValue();
    }
}
```

```
package org.example.client.controllers.expression.object;

import javafx.fxml.FXML;
import javafx.scene.control.ComboBox;
import lombok.Data;
import lombok.Getter;

@Data
public abstract class ExpressionObjectController {
    protected Runnable actionOnDelete;
    public Object object;

    @FXML
    protected void OnDelete(){
        if (actionOnDelete != null) {
            actionOnDelete.run();
        }
    }
    public abstract String getMethodString();
}
```

```
package org.example.client.controllers.expression.object;

import javafx.fxml.FXML;
import javafx.scene.control.TextField;
import lombok.Setter;

public class TextFieldController extends ExpressionObjectController {
    @FXML
    private TextField textField;

    public void setInitValue(String initValue) {
        this.initValue = initValue;
    }

    private String initValue = "";

    @Override
    public Object getObject() {
```

```
        return textField;
    }

    @FXML
    public void initialize() {
        if (initValue.length() >0){
            System.out.println(initValue);
            textField.setText(initValue.toString());
        }
    }

    public double getValue() {
        return Double.parseDouble(textField.getText());
    }

    public String getMethodString() {
        return "getText";
    }
}
```

```
package org.example.client.controllers.parser;

import javafx.scene.control.ChoiceBox;
import org.example.client.controllers.ExpressionsContainerController;
import org.jetbrains.annotations.NotNull;

import java.util.Optional;

public class ConditionLogicOperatorParser {
    public @NotNull String parse(ExpressionsContainerController
expressionContainer) {
        Optional<String> logicalOperatorOptional =
getOperatorFromExpression(expressionContainer);
        return logicalOperatorOptional.orElse("AND");
    }

    private Optional<String>
getOperatorFromExpression(ExpressionsContainerController containerController) {
        ChoiceBox<String> choiceBox = containerController.getChoiceBox();
        if (choiceBox != null) {
            return Optional.of(choiceBox.getValue());
        }
        return Optional.empty();
    }
}
```

```
package org.example.client.controllers.parser;

import javafx.scene.control.TextField;
import org.example.client.controllers.ExpressionController;
import
org.example.client.controllers.expression.object.ExpressionObjectController;
import org.example.entity.Device;

public class ExpressionObjectListParser {
    public Object[] getExpression(Object targetObject, ExpressionController
expressionController){
        Object[] expression = new Object[3];
        String operand = expressionController.getChoiceBox().getValue();
        if (isDevice(targetObject)){
            String method = getMethod(expressionController);
            expression = new Object[]{targetObject, method, operand};
```

```
        }else if (isTextField(targetObject)){
            TextField textField = (TextField) targetObject;
            targetObject = textField.getText();
            expression = new Object[]{targetObject, null, operand};
        }
        return expression;
    }

    private boolean isTextField(Object targetObject) {
        return targetObject instanceof TextField;
    }

    private boolean isDevice(Object targetObject) {
        return targetObject instanceof Device;
    }

    private String getMethod(ExpressionController expressionController) {
        ExpressionObjectController expressionObjectController =
expressionController.getExpressionObjectController();
        return expressionObjectController.getMethodString();
    }

}
```

```
package org.example.client.controllers.parser;

import org.example.rule.entity.Expression;

import java.util.List;

public class ExpressionParser {
    public Expression parseExpressionList(List<Object[]> expressionList) {
        Expression[] expressions = new Expression[expressionList.size()];
        for (int i = 0; i < expressionList.size(); i++) {
            Object[] entry = expressionList.get(i);
            expressions[i] = new Expression(entry[0], (String) entry[1]);
        }

        Expression result = expressions[0];
        for (int i = 1; i < expressions.length; i++) {
            String operator = (String) expressionList.get(i - 1)[2];
            if (operator != null) {
                result = new Expression(result, operator, expressions[i]);
            }
        }
        return result;
    }
}
```

```
package org.example.client.controllers;

import javafx.fxml.FXML;
import javafx.scene.Parent;
import javafx.scene.control.ComboBox;
import javafx.scene.control.TextField;
import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import lombok.Data;
import lombok.Setter;
import org.example.client.service.ObjectService;
import org.example.client.view.ExpressionsContainerView;
```

```
import java.util.ArrayList;
import java.util.List;

@Data
public class ActionController {
    private final List<Object> objects;
    @FXML
    private ComboBox<Object> actionObjectSelector;
    @FXML
    private ComboBox<String> actionFieldSelector;
    @FXML
    private VBox expressionsContainer;

    ExpressionsContainerController expressionsContainerControllers;
    @Setter
    Object selectedObject;
    @Setter
    String selectedField;
    @Setter
    ExpressionsContainerView expressionsContainerView;
    private Runnable onDelete;

    public ActionController() {
        ObjectService objectService = ObjectService.getInstance();
        this.objects = objectService.getObjects();
    }

    @FXML
    public void initialize() {
        actionObjectSelector.getItems().addAll(objects);
        actionObjectSelector.setOnAction(e -> {
            Object selectedObject = actionObjectSelector.getValue();
            if (selectedObject != null) {
                actionFieldSelector.getItems().clear();
                for (var method : selectedObject.getClass().getDeclaredMethods())
{
                    actionFieldSelector.getItems().add(method.getName());
                }
            }
        });
        if (selectedObject != null) actionObjectSelector.setValue(selectedObject);
        if (selectedField != null) actionFieldSelector.setValue(selectedField);
        if (expressionsContainerView != null) {
            addExpressionsContainerView(expressionsContainerView);
        }else {
            ExpressionsContainerView expressionsContainerView = new
ExpressionsContainerView(objects);
            addExpressionsContainerView(expressionsContainerView);
        }
    }

    public void addExpressionsContainerView(ExpressionsContainerView
expressionsContainerView) {
        Parent expressionScene = expressionsContainerView.getView();
        ExpressionsContainerController expressionsContainerController =
expressionsContainerView.getExpressionsContainerController();
        expressionsContainer.getChildren().add(expressionScene);
        expressionsContainerControllers = expressionsContainerController;
    }


    @FXML
    private void handleDeleteButtonAction() {
        onDelete.run();
```

```
    }
}
```

```
package org.example.client.controllers;

import javafx.fxml.FXML;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.control.ChoiceBox;
import javafx.scene.layout.VBox;
import lombok.Getter;
import org.example.client.SceneManager;
import org.example.client.service.ObjectService;
import org.example.client.view.ExpressionsContainerView;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

@Getter
public class ConditionalController {
    private final List<Object> objects;
    List<ExpressionsContainerController> expressionsContainerControllers = new
ArrayList<>();
    SceneManager sceneManager;
    @FXML
    VBox vBox;
    @FXML
    VBox conditionalBox;

    Map<ExpressionsContainerView, String> expressionContainers = new HashMap<>();

    public ConditionalController() {
        this.sceneManager = new SceneManager();
        ObjectService objectService = new ObjectService();
        this.objects = objectService.getObjects();
    }

    @FXML
    private void initialize() {
        if (expressionContainers!= null){
            for(Map.Entry<ExpressionsContainerView, String> entry :
expressionContainers.entrySet()){
                addExpressionsContainer(entry.getKey(), entry.getValue());
            }
        }else {
            addConditional();
        }
    }

    @FXML
    public void onAddConditional() {
        addConditional();
    }

    private void addConditional() {
        ExpressionsContainerView expressionsContainerView = new
ExpressionsContainerView(objects);
        addExpressionsContainer(expressionsContainerView, null);
    }

    public void addExpressionsContainer(ExpressionsContainerView
```

```
expressionsContainerView, String selectedLogicOperator) {
        ExpressionsContainerController expressionsContainerController =
expressionsContainerView.getExpressionsContainerController();
        expressionsContainerController.addRemoveButton();
        Parent expressionScene = expressionsContainerView.getView();
        VBox conditionalItem = new VBox();
        conditionalItem.setId("itisVbox");
        if (!expressionsContainerControllers.isEmpty()) {
            ChoiceBox<String> choiceBox = getChoiceLogicOperand();
            conditionalItem.getChildren().add(choiceBox);
            expressionsContainerController.setChoiceLogic(choiceBox);
            if (selectedLogicOperator != null &&
!selectedLogicOperator.isEmpty()){
                choiceBox.setValue(selectedLogicOperator);
            }
        }
        conditionalItem.getChildren().add(expressionScene);
        conditionalBox.getChildren().add(conditionalItem);
        expressionsContainerController.setActionOnDelete(() ->
onDeleteConditional(conditionalItem, expressionsContainerController));
        expressionsContainerControllers.add(expressionsContainerController);
    }

    public void onDeleteConditional(Parent conditionalPane,
ExpressionsContainerController expressionsContainerController) {
        conditionalBox.getChildren().remove(conditionalPane);
        expressionsContainerControllers.remove(expressionsContainerController);
    }


    private ChoiceBox<String> getChoiceLogicOperand() {
        ChoiceBox<String> choiceBox = new ChoiceBox<>();
        choiceBox.getItems().addAll("AND", "OR");
        choiceBox.setValue("AND");
        return choiceBox;
    }


    public void addExpressionsContainer(ExpressionsContainerController
expressionsContainerController) {
        expressionsContainerControllers.add(expressionsContainerController);
        ExpressionsContainerView containerView = new
ExpressionsContainerView(expressionsContainerController);
        Parent parent = containerView.getView();
        expressionsContainerController.setActionOnDelete(()-
>this.onDeleteConditional(parent, expressionsContainerController));
        expressionsContainerController.addRemoveButton();
    }

    public void addExpressionView(ExpressionsContainerView
expressionsContainerView, String logicalOperator) {
        expressionContainers.put(expressionsContainerView, logicalOperator);
    }
}
```

```
package org.example.client.controllers;

import javafx.fxml.FXML;
import javafx.scene.control.ChoiceBox;
import javafx.scene.layout.HBox;
import lombok.Getter;
import lombok.Setter;
import org.example.client.controllers.expression.object.DeviceController;
```

```
import
org.example.client.controllers.expression.object.ExpressionObjectController;
import org.example.client.controllers.expression.object.TextFieldController;
import org.example.client.factory.ObjectViewFactory;
import org.example.client.view.expression.object.ObjectView;
import org.example.client.view.expression.object.TextFieldView;

@Getter
public class ExpressionController {

    private String lastOperator;
    @FXML
    private HBox expressionsBox;
    private ObjectView objectView;
    ChoiceBox<String> choiceBox = new ChoiceBox<>();
    ;
    private ExpressionObjectController expressionObjectController;
    @Setter
    private String selectedOperator;

    public Object getObject() {
        return expressionObjectController.getObject();
    }

    public ExpressionController(Object object) {
        System.out.println("obj "+object);

        if (object instanceof Number|| object instanceof  Boolean){
            System.out.println("number "+object);

            TextFieldController textFieldController = new TextFieldController();
            textFieldController.setInitValue(String.valueOf(object));
            TextFieldView textFieldView = new TextFieldView(textFieldController);
            this.expressionObjectController =
textFieldView.getExpressionObjectController();
            this.objectView = textFieldView;
        }else {
            objectInit(object);
        }
    }

    public ExpressionController(Object object, String lastOperator) {
        objectInit(object);
        this.lastOperator = lastOperator;
    }

    private void objectInit(Object object) {
        ObjectViewFactory objectViewFactory = new ObjectViewFactory();
        this.objectView = objectViewFactory.getObjectView(object);
        this.expressionObjectController =
objectView.getExpressionObjectController();
    }

    @FXML
    public void initialize() {
        expressionsBox.getChildren().add(objectView.getView());
        if (selectedOperator != null) {
            choiceBox.getItems().addAll("+", "-", "/", "*", "==", ">", "<", ">=",
"<=");
            choiceBox.setValue(selectedOperator);
            expressionsBox.getChildren().add(choiceBox);
        } else if(isNotEnd()) {
            choiceBox.getItems().addAll("+", "-", "/", "*", "==", ">", "<", ">=",
"<=");
```

```
            expressionsBox.getChildren().add(choiceBox);
        }
    }

    public void setDeviceMethod(String methodName) {
        ((DeviceController) expressionObjectController).setMethod(methodName);
    }

    private boolean isNotEnd() {
        if (lastOperator == null) return true;
        if (
                lastOperator.equals("==") ||
                        lastOperator.equals("<") ||
                        lastOperator.equals(">") ||
                        lastOperator.equals(">=") ||
                        lastOperator.equals("<=")) {
            return false;
        }
        return true;
    }
}
```

```
package org.example.client.controllers;


import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.fxml.FXML;
import javafx.scene.Parent;

import javafx.scene.control.Button;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.ComboBox;
import javafx.scene.layout.HBox;

import lombok.Getter;
import lombok.Setter;
import
org.example.client.controllers.expression.object.ExpressionObjectController;
import org.example.client.service.ObjectService;
import org.example.client.view.ExpressionView;
import org.example.entity.Device;

import java.util.ArrayList;
import java.util.List;

@Getter
public class ExpressionsContainerController {
    private final List<Object> objects;


    @Setter
    private List<ExpressionController> expressionControllers = new ArrayList<>();
    @FXML
    private HBox deviceContainer;
    @FXML
    private ComboBox<Object> deviceChoiceBox = new ComboBox<>();
    @Setter
    private Runnable actionOnDelete;
    private ChoiceBox<String> choiceBox;
    private final ObservableList<Object> objectsObservable =
FXCollections.observableArrayList();
    @FXML
```

```
    HBox controlElement;



    public ExpressionsContainerController() {
        ObjectService objectService = ObjectService.getInstance();
        this.objects = objectService.getObjects();
    }

    @FXML
    public void initialize() {
        deviceChoiceBox.setItems(objectsObservable);
        objectsObservable.addAll(objects);
        if (expressionControllers != null){
            for (ExpressionController expressionController :
expressionControllers) {
                ExpressionView expressionView = new
ExpressionView(expressionController);
                Parent expressionPane = expressionView.getView();
                ExpressionObjectController deviceViewController =
expressionController.getExpressionObjectController();
                deviceViewController.setActionOnDelete(() ->
onDeleteChildExpression(expressionPane, expressionController));
                deviceContainer.getChildren().add(expressionPane);
            }
        }
    }

    @FXML
    public void onAddValue(){
        String textField= "textField";
        addObject(textField);
        Device device = (Device) objects.getFirst();
        device.updateDevice();
    }

    @FXML
    public void onAddObject() {
        Object object = deviceChoiceBox.getValue();
        addObject(object);
    }


    private void addObject(Object object) {
        String selectedOperator = null;
        if (!this.expressionControllers.isEmpty()) {
            selectedOperator =
this.expressionControllers.getLast().getChoiceBox().getValue();
        }
        ExpressionView expressionView = new ExpressionView(object,
selectedOperator);
        fillContainer(expressionView);
    }

    private void fillContainer(ExpressionView expressionView) {
        Parent expressionPane = expressionView.getView();
        ExpressionController expressionController =
expressionView.getExpressionModelView();
        ExpressionObjectController deviceViewController =
expressionController.getExpressionObjectController();
        deviceViewController.setActionOnDelete(() ->
onDeleteChildExpression(expressionPane, expressionController));
        deviceContainer.getChildren().add(expressionPane);
        expressionControllers.add(expressionController);
```

```
    }

    public void onDeleteChildExpression(Parent expressionParent,
ExpressionController expressionController) {
        deviceContainer.getChildren().remove(expressionParent);
        expressionControllers.remove(expressionController);
    }


    @FXML
    public void onDeleteThisExpressionsContainer() {
        if (actionOnDelete != null) {
            actionOnDelete.run();
        }
    }

    public void addRemoveButton(){
        Button onDeleteThisExpressionsContainerButton = new Button("Delete");
        onDeleteThisExpressionsContainerButton.setMnemonicParsing(false);
        onDeleteThisExpressionsContainerButton.setStyle("-fx-background-color:
#F44336; -fx-text-fill: white; -fx-font-size: 14px;");
onDeleteThisExpressionsContainerButton.setOnAction(_ -
>onDeleteThisExpressionsContainer());
        controlElement.getChildren().add(onDeleteThisExpressionsContainerButton);
    }

    public void setChoiceLogic(ChoiceBox<String> choiceBox) {
        this.choiceBox = choiceBox;
    }

}
```

```
package org.example.client.controllers;

import javafx.beans.property.DoubleProperty;
import javafx.beans.property.Property;
import javafx.beans.property.StringProperty;
import javafx.fxml.FXML;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.VBox;
import org.example.client.service.DoubleStringConverter;
import org.example.entity.Device;

import java.lang.reflect.Field;

public class FullDeviceController {
    @FXML
    public Label deviceId;
    @FXML
    private VBox deviceDetailsBox;

    public void setDevice(Object device) {
        this.device = (Device) device;
        deviceId.setText(String.format("%s\nDeviceId: %s",
this.device.getDeviceType(),this.device.getSensorId()));
        populateDeviceDetails();
    }

    private Device device;
```

```
    public FullDeviceController() {

    }

    private void populateDeviceDetails() {
        try {
            if (device != null) {
                deviceDetailsBox.getChildren().clear();
                Device deviceEnitity = device;

                Field[] fields = deviceEnitity.getClass().getDeclaredFields();
                for (Field field : fields) {
                    field.setAccessible(true);
                    try {

                        Property<?> property = (Property<?>)
field.get(deviceEnitity);
                        if (property instanceof StringProperty) {

                            Label label = new Label(field.getName() + ": ");
                            TextField textField = new TextField();


textField.textProperty().bindBidirectional(((StringProperty) property));
                            deviceDetailsBox.getChildren().addAll(label,
textField);
                        } else if (property instanceof DoubleProperty) {

                            Label label = new Label(field.getName() + ": ");
                            TextField textField = new TextField();

textField.textProperty().bindBidirectional(((DoubleProperty) property), new
DoubleStringConverter());
                            deviceDetailsBox.getChildren().addAll(label,
textField);
                        }
                    } catch (IllegalAccessException e) {
                        e.printStackTrace();
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


}
```

```
package org.example.client.controllers;

import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import javafx.fxml.FXML;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import org.example.entity.Transformer;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisher;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisherFactory;


public class MainSceneController {
```

```
    @FXML
    private Button mainButton;

    @FXML
    private TextField turnsRation;

    @FXML
    private String baseTurnsRation;

    @FXML
    private StringProperty textContent = new SimpleStringProperty("1");

    @FXML
    public void initialize() {
        turnsRation.setText("1");
    }
    @FXML
    private void buttonClicked() {
        MQTTPublisher mqttPublisher =
MQTTPublisherFactory.getPublisher("GUIClientPusher");
        Transformer transporter = new Transformer();
        transporter.setSensorId("device1");
        transporter.setTurnsRation(Double.parseDouble(turnsRation.getText()));
        mqttPublisher.writeData(transporter);
    }
}
```

```
package org.example.client.controllers;

import javafx.animation.Interpolator;
import javafx.animation.TranslateTransition;
import javafx.fxml.FXML;
import javafx.scene.Node;
import javafx.scene.Parent;
import javafx.scene.control.ScrollPane;
import javafx.scene.input.*;
import javafx.scene.layout.AnchorPane;

import javafx.scene.layout.Pane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Text;
import javafx.scene.transform.Scale;
import javafx.scene.shape.Rectangle;
import org.example.client.view.FullDeviceView;
import org.example.entity.Device;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.logging.Logger;

public class MapController {

    private static final Logger logger =
Logger.getLogger(MapController.class.getName());

    @FXML
    private VBox listBox;

    @FXML
    private AnchorPane dropPane;
    @FXML
```

```
    private ScrollPane scrollPane;

    private final Map<VBox, Device> vBoxDeviceMap = new HashMap<>();
    private VBox draggedVBox = null;

    private final List<FullDeviceController> fullDeviceControllers = new
ArrayList<>();

    public MapController() {}

    @FXML
    public void initialize() {}

    public void setDevices(List<Object> devices) {
        if (devices == null) {
            throw new IllegalArgumentException("Devices list cannot be null");
        }
        initialize(devices);
    }

    private String getDeviceString(Device device) {
        return String.format("%s %s", device.getDeviceType(),
device.getSensorId());
    }

    public void initialize(List<Object> objects) {
        if (listBox == null || dropPane == null) {
            throw new IllegalStateException("FXML elements are not initialized");
        }

        // Получаем список устройств
        List<Device> devices = getDevicesFromObjects(objects);

        initializeDeviceList(devices);
        setupDragAndDropParsers(devices);
    }

    private List<Device> getDevicesFromObjects(List<Object> objects) {
        List<Device> devices = new ArrayList<>();
        for (Object object : objects) {
            if (object instanceof Device device) {
                devices.add(device);
            }
        }
        return devices;
    }

    private void initializeDeviceList(List<Device> devices) {
        for (Device device : devices) {
            String deviceInfo = getDeviceString(device);
            logger.info("Device info: " + deviceInfo);
            Text text = new Text(deviceInfo);

            // Обработчик начала перетаскивания
            text.setOnDragDetected(event -> handleDragDetected(event,
deviceInfo));

            listBox.getChildren().add(text);
        }
    }

    private void handleDragDetected(MouseEvent event, String deviceInfo) {
        Text sourceText = (Text) event.getSource();
        Dragboard db = sourceText.startDragAndDrop(TransferMode.MOVE);
```

```
        ClipboardContent content = new ClipboardContent();
        content.putString(deviceInfo);
        db.setContent(content);
        event.consume();
    }

    private void setupDragAndDropParsers(List<Device> devices) {
        dropPane.setOnDragOver(event -> handleDragOver(event));
        dropPane.setOnDragDropped(event -> handleDragDropped(event, devices));
        dropPane.setOnKeyPressed(e->{
            if (e.getCode() == KeyCode.DELETE && draggedVBox != null) {
                dropPane.getChildren().remove(draggedVBox);
            }
        });
    }

    private void handleDragOver(DragEvent event) {
        if (event.getGestureSource() != dropPane &&
event.getDragboard().hasString()) {
            event.acceptTransferModes(TransferMode.MOVE);
        }
        event.consume();
    }

    private void handleDragDropped(DragEvent event, List<Device> devices) {
        Dragboard db = event.getDragboard();
        if (db.hasString()) {
            String deviceInfo = db.getString();
            logger.info("Drag Dropped: " + deviceInfo);

            if (event.getGestureSource() instanceof Parent) {
                if (draggedVBox != null) {
                    dropPane.getChildren().remove(draggedVBox);
                    vBoxDeviceMap.remove(draggedVBox);
                }
            }

            for (Device device : devices) {
                if (getDeviceString(device).equals(deviceInfo)) {
                    addDeviceToDropPane(device, event);
                    event.setDropCompleted(true);
                    break;
                }
            }
        } else {
            event.setDropCompleted(false);
        }
        event.consume();
    }

    private void addDeviceToDropPane(Device device, DragEvent event) {
        FullDeviceView fullDeviceView = new FullDeviceView(device);
        FullDeviceController fullDeviceController =
fullDeviceView.getController();
        Parent deviceView = fullDeviceView.getView();

        VBox vBox = new VBox();
        vBox.getChildren().add(deviceView);
        dropPane.getChildren().add(vBox);

        vBoxDeviceMap.put(vBox, device);
        draggedVBox = vBox;

        double x = event.getX();
```

```
        double y = event.getY();
        vBox.setLayoutX(x - vBox.getWidth() / 2);
        vBox.setLayoutY(y - vBox.getHeight() / 2);

        setupDragAndDropForVBox(vBox, device);
    }

    private void setupDragAndDropForVBox(VBox vBox, Device device) {

        vBox.setOnMousePressed(event -> {
            draggedVBox = vBox;
            mouseDragX = event.getSceneX();
            mouseDragY = event.getSceneY();
            startDragX = vBox.getLayoutX();
            startDragY = vBox.getLayoutY();
            vBox.requestFocus();
        });

        vBox.setOnMouseDragged(event -> {
            if (draggedVBox != null) {

                double deltaX = event.getSceneX() - mouseDragX;
                double deltaY = event.getSceneY() - mouseDragY;


                double newX = startDragX + deltaX;
                double newY = startDragY + deltaY;


                draggedVBox.setLayoutX(newX);
                draggedVBox.setLayoutY(newY);
            }
        });

        vBox.setOnMouseReleased(event -> {
            draggedVBox = null;
        });


        vBox.setOnKeyPressed(event -> {

            if (event.getCode() == KeyCode.DELETE) {
                dropPane.getChildren().remove(vBox);

            }
        });


        vBox.setFocusTraversable(true);
    }


    private double mouseDragX, mouseDragY, startDragX, startDragY;



}
```

```
package org.example.client.controllers;

import javafx.fxml.FXML;
import javafx.scene.Parent;
import javafx.scene.control.Alert;
```

```
import javafx.scene.layout.VBox;
import org.example.client.view.RuleContainerSceneView;
import org.example.rule.RuleBuilder;
import org.example.service.RuleService;

import java.util.List;


public class RuleBuilderSceneController {
    @FXML
    private VBox ruleContainer;

    RuleContainerController ruleContainerController;
    RuleContainerSceneView ruleContainerSceneView;

    public RuleBuilderSceneController() {

    }

    @FXML
    public void initialize(){
        this.ruleContainerSceneView =  new RuleContainerSceneView();
        this.ruleContainerController = ruleContainerSceneView.getController();
        Parent parent = ruleContainerSceneView.getView();
        this.ruleContainer.getChildren().add(parent);
    }

    @FXML
    public void onSaveRule() {
        ConditionalController conditionControllers=
ruleContainerController.getConditionControllers();
        List<ActionController> actionControllers =
ruleContainerController.getActionControllers();
        RuleBuilder ruleBuilder = new RuleBuilder()
                .fromUi(conditionControllers, actionControllers);
        RuleService ruleService = new RuleService();
        ruleService.saveRule(ruleBuilder.build());
        System.out.println("Rule successfully added!");

        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Success");
        alert.setHeaderText(null);
        alert.setContentText("Rule successfully added!");
        alert.showAndWait();

    }
}
```

```
package org.example.client.controllers;

import javafx.fxml.FXML;
import javafx.scene.Parent;
import javafx.scene.control.Alert;
import javafx.scene.layout.VBox;
import lombok.Getter;
import lombok.Setter;
import org.example.client.SceneManager;
import org.example.client.view.ActionView;
import org.example.client.view.ConditionalView;
import org.example.rule.RuleBuilder;
import org.example.service.RuleService;
```

```
import java.util.ArrayList;
import java.util.List;

public class RuleContainerController {
    @FXML
    private VBox conditionsContainer;
    @FXML
    private VBox actionsContainer;

    @Getter
    private ConditionalController conditionControllers;
    @Getter
    private final List<ActionController> actionControllers = new ArrayList<>();
    @Setter
    private ConditionalView conditionalView;
    private final List<ActionView> actionViews = new ArrayList<>();

    public RuleContainerController() {

    }

    @FXML
    private void initialize() {
        if (conditionalView != null){
            addConditionalView(conditionalView);
        }else {
            addCondition();
        }
        if (actionViews !=null){
            for (ActionView actionView : actionViews){
                addAction(actionView);
            }
        }
    }

    @FXML
    public void addCondition() {
        SceneManager sceneManager = new SceneManager();
        ConditionalView conditionalView = new ConditionalView();
        addConditionalView(conditionalView);
    }

    private void addConditionalView(ConditionalView conditionalView) {
        Parent conditionalScene = conditionalView.getView();
        ConditionalController conditionalModelView =
conditionalView.getConditionalView();
        conditionsContainer.getChildren().add(conditionalScene);
        conditionControllers = conditionalModelView ;
    }


    @FXML
    public void onAddAction() {
        ActionView actionView = new ActionView();
        addAction(actionView);
    }

    private void addAction(ActionView actionView) {
        Parent actionScene = actionView.getView();
        ActionController actionController = actionView.getController();
        actionsContainer.getChildren().add(actionScene);
        actionControllers.add(actionController);
        actionController.setOnDelete(()->onDeleteAction(actionScene,
actionController));
```

```
    }

    @FXML
    public void onDeleteAction(Parent expressionParent, ActionController
actionController) {
        actionsContainer.getChildren().remove(expressionParent);
        actionControllers.remove(actionController);
    }

    private void clearConditional() {

conditionControllers.getExpressionsContainerControllers().forEach(ExpressionsConta
inerController::onDeleteThisExpressionsContainer);
    }


    public void onAddActionController(ActionController actionController) {
        ActionView actionView = new ActionView(actionController);
        actionViews.add(actionView);
    }
}
```

```
package org.example.client.controllers;

import javafx.fxml.FXML;
import javafx.geometry.HPos;
import javafx.scene.Node;
import javafx.scene.control.Label;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Priority;
import javafx.scene.layout.VBox;

import org.example.client.ruleparser.RuleSceneControllerParser;
import org.example.client.service.ObjectService;
import org.example.client.view.RuleContainerSceneView;
import org.example.rule.entity.Rule;
import org.example.service.RuleService;

import java.util.ArrayList;
import java.util.List;

public class RulesSceneController {

    List<Rule> rules;
    List<RuleContainerController> rulesSceneControllers = new ArrayList<>();

    public RulesSceneController() {
        ObjectService objectService = ObjectService.getInstance();
        rules = objectService.getRules();
    }

    @FXML
    private GridPane ruleGrid;

    @FXML
    private void initialize() {
        RuleSceneControllerParser ruleControllerParser = new
RuleSceneControllerParser();
        int rowIndex = 0;
        for (Rule rule : rules) {
            GridPane hBox = getRuleNodeItem(rule, ruleControllerParser);
            ruleGrid.add(hBox, 0, rowIndex);
```

```
            GridPane.setHgrow(hBox, Priority.ALWAYS);
            rowIndex++;
        }
    }

    private GridPane getRuleNodeItem(Rule rule, RuleSceneControllerParser
ruleControllerParser) {
        GridPane gridPane = new GridPane();
        gridPane.setHgap(10);
        gridPane.setVgap(5);

        Label idRule = new Label(String.valueOf(rule.getId()));
        Label descriptionRule = new Label(rule.getDescription());

        RuleContainerController controller =
ruleControllerParser.getRuleSceneController(rule);
        rulesSceneControllers.add(controller);
        Node ruleSceneView = new RuleContainerSceneView(controller).getView();

        gridPane.add(idRule, 0, 0);
        gridPane.add(descriptionRule, 1, 0);
        gridPane.add(ruleSceneView, 2, 0);

        gridPane.setStyle("-fx-border-color: white; -fx-border-width: 2; -fx-
border-style: solid;");

        idRule.setStyle("-fx-border-bottom-color: white; -fx-border-bottom-width:
1px;");
        descriptionRule.setStyle("-fx-border-bottom-color: white; -fx-border-
bottom-width: 1px;");
        ruleSceneView.setStyle("-fx-border-bottom-color: white; -fx-border-bottom-
width: 1px;");

        GridPane.setHalignment(idRule, HPos.LEFT);
        GridPane.setHalignment(descriptionRule, HPos.LEFT);
        GridPane.setHgrow(ruleSceneView, Priority.ALWAYS);



        return gridPane;
    }


    private VBox getRuleLabelNode(Rule rule) {
        VBox vBox = new VBox();
        vBox.setSpacing(5);
        Label idRule = new Label();
        Label descriptionRule = new Label();
        idRule.setText(String.valueOf(rule.getId()));
        descriptionRule.setText(rule.getDescription());
        vBox.getChildren().addAll(idRule, descriptionRule);
        return vBox;
    }



}
```

```
package org.example.client.entity;

import javafx.geometry.Insets;
import javafx.scene.control.*;
import javafx.scene.layout.HBox;
```

```
import java.util.List;

public class ActionUI {
    private final List<Object> objects;
    private Runnable onDelete;

    public ActionUI(List<Object> objects, Runnable onDelete) {
        this.objects = objects;
        this.onDelete = onDelete;
    }

    public HBox createActionRow() {
        ComboBox<Object> actionObjectSelector = new ComboBox<>();
        actionObjectSelector.getItems().addAll(objects);
        actionObjectSelector.setPromptText("Choice object");

        ComboBox<String> actionFieldSelector = new ComboBox<>();
        actionFieldSelector.setPromptText("Choice method");

        actionObjectSelector.setOnAction(e -> {
            Object selectedObject = actionObjectSelector.getValue();
            if (selectedObject != null) {
                actionFieldSelector.getItems().clear();
                for (var field : selectedObject.getClass().getDeclaredMethods()) {
                    actionFieldSelector.getItems().add(field.getName());
                }
            }
        });

        TextField actionValue = new TextField();
        actionValue.setPromptText("Input new value");

        Button deleteButton = new Button("Delete");
        deleteButton.setOnAction(e -> onDelete.run());

        HBox actionRow = new HBox(5, actionObjectSelector, actionFieldSelector,
actionValue, deleteButton);
        actionRow.setPadding(new Insets(5));
        return actionRow;
    }
}
```

```
package org.example.client.entity;

import org.example.entity.CurrentLineSensor;

public class CurrentSensorUI extends CurrentLineSensor{
    public CurrentSensorUI(CurrentLineSensor currentLineSensor) {
        super(currentLineSensor.getSensorId(), currentLineSensor.getVoltage(),
currentLineSensor.getCurrent());
    }

    @Override
    public void setCurrent(double current) {
        super.setCurrent(current);
    }

    @Override
    public void setVoltage(double voltage) {
        super.setVoltage(voltage);
    }
```

```
    @Override
    public double getVoltage() {
        return super.getVoltage();
    }

    @Override
    public double getCurrent() {
        return super.getCurrent();
    }
}
```

```
package org.example.client.entity;

import org.example.entity.Device;
import org.example.entity.Generator;

public class GeneratorUI extends Generator {
    public GeneratorUI(Generator generator) {
        super(generator.getSensorId(), generator.getVoltage(),
generator.getCurrent(), generator.getIsWorking());
    }

    @Override
    public double getCurrent() {
        return super.getCurrent();
    }

    @Override
    public double getVoltage() {
        return super.getVoltage();
    }

    @Override
    public boolean getIsWorking() {
        return super.getIsWorking();
    }

    @Override
    public void setVoltage(double voltage) {
        super.setVoltage(voltage);
    }

    @Override
    public void setCurrent(double current) {
        super.setCurrent(current);
    }

    @Override
    public void setIsWorking(Boolean isWorking) {
        super.setIsWorking(isWorking);
    }
}
```

```
package org.example.client.factory;

import org.example.client.entity.CurrentSensorUI;
import org.example.client.entity.GeneratorUI;
import org.example.entity.CurrentLineSensor;
import org.example.entity.Device;
```

```
import org.example.entity.Generator;

public class DeviceUIFactory {
    public Device getDeviceUi(Device device){
        if (device instanceof CurrentLineSensor currentLineSensor){
            return new CurrentSensorUI(currentLineSensor);
        }if (device instanceof Generator generator){
            return new GeneratorUI(generator);
        }
        else {
            return device;
        }
    }
}
```

```
package org.example.client.factory;

import javafx.scene.control.TextField;
import org.example.client.view.expression.object.DeviceView;
import org.example.client.view.expression.object.ObjectView;
import org.example.client.view.expression.object.TextFieldView;
import org.example.entity.Device;

public class ObjectViewFactory {
    public ObjectView getObjectView(Object object){
        if (object instanceof Device device){
            return new DeviceView(device);
        }else {
            return new TextFieldView();
        }

    }
}
```

```
package org.example.client.ruleparser;

import org.example.client.controllers.ActionController;
import org.example.client.controllers.ExpressionsContainerController;
import org.example.client.view.ActionView;
import org.example.client.view.ExpressionsContainerView;
import org.example.rule.entity.Action;
import org.example.rule.entity.Expression;

public class ActionControllerParser {
    public ActionController parse(Action action) {
        ActionController actionController = new ActionController();
        actionController.setSelectedObject(action.getTargetObject());
        actionController.setSelectedField(action.getFieldName());
        Expression expression = action.getExpression();
        ExpressionContainerControllerParser expressionParser = new
ExpressionContainerControllerParser();
        ExpressionsContainerController expressionsContainerController =
expressionParser.getExpressionContainer(expression);
        ExpressionsContainerView expressionsContainerView = new
ExpressionsContainerView(expressionsContainerController);
        actionController.setExpressionsContainerView(expressionsContainerView);
        return actionController;
    }

}
```

```
package org.example.client.ruleparser;

import org.example.client.controllers.ConditionalController;
import org.example.client.controllers.ExpressionsContainerController;
import org.example.client.view.ExpressionsContainerView;
import org.example.rule.entity.ConditionWithOperator;
import org.example.rule.entity.Expression;
import org.jetbrains.annotations.NotNull;

import java.util.List;

public class ConditionControllerParser {
    public @NotNull ConditionalController extract(List<ConditionWithOperator>
conditionWithOperators) {
        ConditionalController controller = new ConditionalController();
        ExpressionContainerControllerParser expressionParser = new
ExpressionContainerControllerParser();
        for (ConditionWithOperator conditionWithOperator : conditionWithOperators)
{
            Expression expression = conditionWithOperator.getCondition();
            ExpressionsContainerController expressionsContainerController =
expressionParser.getExpressionContainer(expression);
            ExpressionsContainerView expressionsContainerView = new
ExpressionsContainerView(expressionsContainerController);
            controller.addExpressionView(expressionsContainerView,
conditionWithOperator.getLogicalOperator());
        }
        return controller;
    }
}
```

```
package org.example.client.ruleparser;

import org.example.client.controllers.ExpressionsContainerController;
import org.example.client.controllers.parser.ConditionLogicOperatorParser;
import org.example.rule.entity.ConditionWithOperator;
import org.example.rule.entity.Expression;
import org.jetbrains.annotations.NotNull;

public class ConditionWithOperatorExtractor {

    public @NotNull ConditionWithOperator extract(ExpressionsContainerController
expressionContainer) {
        String logicalOperator = getLogicalOperator(expressionContainer);
        Expression expression = getExpression(expressionContainer);
        return new ConditionWithOperator(expression, logicalOperator);
    }

    private Expression getExpression(ExpressionsContainerController
expressionContainer) {
        ExpressionExtractor expressionExtractor = new ExpressionExtractor();
        return expressionExtractor.extract(expressionContainer);
    }

    private @NotNull String getLogicalOperator(ExpressionsContainerController
expressionContainer) {
        ConditionLogicOperatorParser logicOperatorParser = new
ConditionLogicOperatorParser();
        return logicOperatorParser.parse(expressionContainer);
    }

}
```

```
package org.example.client.ruleparser;

import org.example.client.controllers.ExpressionController;
import org.example.client.controllers.ExpressionsContainerController;
import org.example.rule.entity.Expression;

import java.util.ArrayList;
import java.util.List;

public class ExpressionContainerControllerParser {
    public ExpressionsContainerController getExpressionContainer(Expression
expression){
        ExpressionReverseParser expressionReverseParser = new
ExpressionReverseParser();
        List<Object[]> expressions =
expressionReverseParser.reverseParseExpression(expression);
        ArrayList<ExpressionController> expressionControllers = new ArrayList<>();
        for(Object[] expressionObject : expressions){
            ObjectExpressionControllerParser objectExpressionControllerParser =
new ObjectExpressionControllerParser();
            ExpressionController expressionController =
objectExpressionControllerParser.getExpressionController(expressionObject);
            expressionControllers.add(expressionController);
        }
        ExpressionsContainerController expressionsContainerController = new
ExpressionsContainerController();

expressionsContainerController.setExpressionControllers(expressionControllers);
        return expressionsContainerController;
    }
}
```

```
package org.example.client.ruleparser;

import org.example.client.controllers.ExpressionController;
import org.example.client.controllers.ExpressionsContainerController;
import org.example.client.controllers.parser.ExpressionObjectListParser;
import org.example.client.controllers.parser.ExpressionParser;
import org.example.rule.entity.Expression;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class ExpressionExtractor {
    public Expression extract(ExpressionsContainerController expressionContainer){
        List<Object[]> expressionList = getExpressionList(expressionContainer);


        return getExpression(expressionList);
    }

    private List<Object[]> getExpressionList(ExpressionsContainerController
expressionContainer) {
        List<Object[]> expressions = new ArrayList<>();
        List<ExpressionController> expressionControllers =
expressionContainer.getExpressionControllers();
        for (ExpressionController expressionController : expressionControllers) {
            Object targetObject = expressionController.getObject();
            ExpressionObjectListParser expressionObjectListParser = new
ExpressionObjectListParser();
```

```
            Object[] expression =
expressionObjectListParser.getExpression(targetObject, expressionController);
            expressions.add(expression);
        }
        return expressions;
    }

    private Expression getExpression(List<Object[]> expressionList) {
        ExpressionParser parser = new ExpressionParser();
        return parser.parseExpressionList(expressionList);
    }
}
```

```
package org.example.client.ruleparser;

import org.example.rule.entity.Expression;

import java.util.ArrayList;
import java.util.List;

public class ExpressionReverseParser {
    public List<Object[]> reverseParseExpression(Expression expression) {
        List<Object[]> expressionList = new ArrayList<>();
        parseExpression(expression, null, expressionList);
        return expressionList;
    }

    private void parseExpression(Expression expression, String operator,
List<Object[]> expressionList) {
        if (expression.getLeftOperand() == null && expression.getRightOperand() ==
null) {

            Object targetObject = expression.getTargetObject();
            String method = expression.getMethodName();
            expressionList.add(new Object[]{targetObject, method, operator});
        } else {
            if (expression.getLeftOperand() != null) {
                parseExpression(expression.getLeftOperand(),
expression.getOperator(), expressionList);
            }

            if (expression.getRightOperand() != null) {
                parseExpression(expression.getRightOperand(), operator,
expressionList);
            }
        }
    }


}
```

```
package org.example.client.ruleparser;

import org.example.client.controllers.ExpressionController;
import org.example.client.service.PrimitiveParser;
import org.example.entity.Device;

public class ObjectExpressionControllerParser {
    public ExpressionController getExpressionController(Object[] expressionData) {
        Object targetObject = expressionData[0];
        String method = (String) expressionData[1];
        String operand = (String) expressionData[2];
```

```
        if (targetObject instanceof Device device) {
            ExpressionController deviceController = new
ExpressionController(device);
            deviceController.setSelectedOperator(operand);
            deviceController.setDeviceMethod(method);
            return deviceController;
        } else  {
            PrimitiveParser parser = new PrimitiveParser();
            targetObject = parser.parse(targetObject);
            ExpressionController deviceController = new
ExpressionController(targetObject);
            deviceController.setSelectedOperator(operand);
            return deviceController;
        }
    }
}
```

```
package org.example.client.ruleparser;

import org.example.client.controllers.ActionController;
import org.example.client.controllers.ConditionalController;
import org.example.client.controllers.RuleContainerController;
import org.example.client.view.ConditionalView;
import org.example.rule.entity.Action;
import org.example.rule.entity.ConditionWithOperator;
import org.example.rule.entity.Rule;

import java.util.List;

public class RuleSceneControllerParser {
    public RuleContainerController getRuleSceneController(Rule rule){
        RuleContainerController ruleBuilderSceneController = new
RuleContainerController();
        ConditionalController controller = getConditionController(rule);
        ConditionalView conditionalView = new ConditionalView(controller);
        ActionControllerParser actionControllerParser = new
ActionControllerParser();
        List<Action> actions = rule.getActions();
        for (Action action : actions){
            ActionController actionController =
actionControllerParser.parse(action);
            ruleBuilderSceneController.onAddActionController(actionController);
        }
        ruleBuilderSceneController.setConditionalView(conditionalView);

        return ruleBuilderSceneController;
    }

    private ConditionalController getConditionController(Rule rule) {
        List<ConditionWithOperator> conditions =
rule.getConditionsWithOperators();
        ConditionControllerParser conditionControllerParser = new
ConditionControllerParser();
        return conditionControllerParser.extract(conditions);
    }
}
```

```
package org.example.client.ruleparser;

import org.example.client.controllers.ActionController;
import org.example.client.controllers.ExpressionsContainerController;
```

```
import org.example.rule.entity.Action;
import org.example.rule.entity.Expression;

import java.util.ArrayList;
import java.util.List;

public class UiActionsExtractor {
    private final List<Action> actions = new ArrayList<>();
    public List<Action> extractActions(List<ActionController> actionControllers) {
        for (ActionController actionController : actionControllers){
            Object targetObject =
actionController.getActionObjectSelector().getValue();
            String method = actionController.getActionFieldSelector().getValue();
            ExpressionsContainerController expressionsContainerControllers =
actionController.getExpressionsContainerControllers();
            ExpressionExtractor expressionExtractor = new ExpressionExtractor();
            Expression expression =
expressionExtractor.extract(expressionsContainerControllers);
            actions.add(new Action(targetObject, method, expression));
        }
        return actions;
    }

}
```

```
package org.example.client.ruleparser;

import org.example.client.controllers.ConditionalController;
import org.example.client.controllers.ExpressionsContainerController;
import org.example.rule.entity.ConditionWithOperator;

import java.util.ArrayList;
import java.util.Collection;
import java.util.List;

public class UiConditionsExtractor {
    private final List<ConditionWithOperator> conditionsWithOperators = new
ArrayList<>();

    public Collection<? extends ConditionWithOperator>
extractConditions(ConditionalController conditionsContainer) {
        List<ExpressionsContainerController> expressionsContainerControllers =
conditionsContainer.getExpressionsContainerControllers();
        return extracted(expressionsContainerControllers);
    }

    public List<ConditionWithOperator>
extracted(List<ExpressionsContainerController>  expressionsContainerController) {
        List<ConditionWithOperator> condition = new ArrayList<>();
        for (ExpressionsContainerController expressionContainer :
expressionsContainerController) {
            ConditionWithOperatorExtractor conditionExtractor = new
ConditionWithOperatorExtractor();
            ConditionWithOperator conditionWithOperator =
conditionExtractor.extract(expressionContainer);
            condition.add(conditionWithOperator);
        }
        return condition;
    }
}
```

```java
package org.example.client.service;

import javafx.util.StringConverter;

public class DoubleStringConverter extends StringConverter<Number> {

    @Override
    public String toString(Number object) {
        return object == null ? "" : object.toString();
    }

    @Override
    public Number fromString(String string) {
        try {
            return Double.parseDouble(string);
        } catch (NumberFormatException e) {
            return 0.0;
        }
    }
}
```

```java
package org.example.client.service;

import org.example.entity.Device;
import org.example.entity.WindSensor;
import org.example.rule.entity.Rule;
import org.example.service.InflexDBService;
import org.example.service.RuleService;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisherFactory;

import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class ObjectService {

    private static ObjectService instance;
    private final List<Object> objects = new ArrayList<>();
    private List<Rule> rules = new ArrayList<>();

    public ObjectService() {
        InflexDBService inflexDBService = new InflexDBService();
        objects.addAll(inflexDBService.getAllDevices());
        updateDevice();
        updateRules();
        objects.add(getWindSensor());
        objects.add(MQTTPublisherFactory.getPublisher("testClient"));
    }

    private void updateRules() {
        Runnable runnable = () -> {
            while (true) {
                RuleService ruleService = new RuleService();
                rules = ruleService.getAllRule();
                try {
                    Thread.sleep(15000);
                } catch (InterruptedException e) {
                    throw new RuntimeException(e);
                }
            }
        };
```

```
        Thread thread = new Thread(runnable);
        thread.start();
    }

    private void updateDevice() {
        ExecutorService executorService = Executors.newFixedThreadPool(10); //
Ограничиваем количество потоков

        Runnable runnable = () -> {
            while (true) {
                List<Object> objectsCopy = new ArrayList<>(objects);
                objectsCopy.forEach(object -> {
                    executorService.submit(() -> {
                        try {
                            if (object instanceof Device device) {
                                device.updateDevice();
                            }
                        } catch (Exception e) {
                            System.err.println("Error updating device: " +
e.getMessage());
                        }
                    });
                });
                try {
                    Thread.sleep(5000);
                } catch (InterruptedException e) {
                    Thread.currentThread().interrupt(); // Правильное завершение
потока
                    break;
                }
            }
        };

        Thread thread = new Thread(runnable);
        thread.start();
    }

    private WindSensor getWindSensor(){
        WindSensor sensor = new WindSensor();
        WindSimulator simulator = new WindSimulator(sensor);

        System.out.println("Начало симуляції:");


        Thread simulationThread = new Thread(new Runnable() {
            @Override
            public void run() {
                simulator.simulateWindChanges();
            }
        });

        simulationThread.start();
        return sensor;
    }

    public static synchronized ObjectService getInstance() {
        if (instance == null) {
            instance = new ObjectService();
        }
        return instance;
    }

    public List<Object> getObjects() {
        return new ArrayList<>(objects);
```

```
    }


    public List<Rule> getRules(){
        return new ArrayList<>(rules);
    }
}
```

```
package org.example.client.service;

public class PrimitiveParser {

    public Boolean parseBoolean(Object value) {
        if (value instanceof Boolean) {
            return (Boolean) value;
        } else if (value instanceof String) {
            String strValue = ((String) value).trim().toLowerCase();
            if ("true".equals(strValue)) {
                return true;
            } else if ("false".equals(strValue)) {
                return false;
            }
        } else if (value instanceof Number) {
            return ((Number) value).doubleValue() != 0;
        }

        return null;
    }

    public  Number parseNumber(Object value) {
        if (value instanceof Number) {
            return (Number) value;
        } else if (value instanceof String) {
            String strValue = ((String) value).trim();
            try {
                    return Double.parseDouble(strValue);
            } catch (NumberFormatException e) {

                return null;
            }
        }

        return null;
    }

    public  String parseString(Object value) {
        if (value == null) {
            return null;
        }
        return value.toString();
    }

    public static boolean isPrimitive(Object value) {
        return value instanceof Boolean || value instanceof Number || value
instanceof String;
    }


    public Object parse(Object value) {
        Number numberValue = parseNumber(value);
        if (numberValue != null) {
            return numberValue;
        }
```

```
        Boolean booleanValue = parseBoolean(value);
        if (booleanValue != null) {
            return booleanValue;
        }

        return parseString(value);
    }
}
```

```
package org.example.client.view.expression.object;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import lombok.Getter;
import org.example.client.controllers.expression.object.DeviceController;

import org.example.entity.Device;

@Getter
public class DeviceView extends ObjectView{
    private final Device deviceModel;

    public DeviceView(Device deviceModel) {
        this.deviceModel = deviceModel;
        this.setExpressionObjectController( new DeviceController(deviceModel));
        this.setView(loadView());
    }

    public Parent loadView() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/device.fxml"));
            loader.setController(getExpressionObjectController());
            return loader.load();
        } catch (Exception e) {
            throw new RuntimeException("Failed to load device view", e);
        }
    }

}
```

```
package org.example.client.view.expression.object;

import javafx.scene.Parent;
import lombok.Data;
import lombok.Getter;
import
org.example.client.controllers.expression.object.ExpressionObjectController;

@Data
public abstract class ObjectView {
    private Parent view;
    private ExpressionObjectController expressionObjectController;

    public abstract Parent loadView();
}
```

```
package org.example.client.view.expression.object;
```

```
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;

import org.example.client.controllers.expression.object.TextFieldController;

public class TextFieldView extends ObjectView {

    public TextFieldView(TextFieldController textFieldController){
        this.setExpressionObjectController(textFieldController);
        this.setView(loadView());
    }

    public TextFieldView() {
        this.setExpressionObjectController(new TextFieldController());
        this.setView(loadView());
    }

    public Parent loadView() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/value.fxml"));
            loader.setController(getExpressionObjectController());
            return loader.load();
        } catch (Exception e) {
            throw new RuntimeException("Failed to load device view", e);
        }
    }
}
```

```
package org.example.client.view;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import lombok.Data;
import org.example.client.controllers.ActionController;

import java.util.List;

@Data
public class ActionView {
    private final Parent view;
    private final ActionController controller;

    public ActionView() {
        controller = new ActionController();
        this.view = loadView();
    }

    public ActionView(ActionController actionController) {
        controller = actionController;
        this.view = loadView();
    }

    private Parent loadView() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/actionView.fxml"));
            loader.setController(controller);
            return loader.load();
        } catch (Exception e) {
            throw new RuntimeException("Failed to load device view", e);
        }
```

```
    }
}
```

```
package org.example.client.view;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import lombok.Getter;
import org.example.client.controllers.ConditionalController;

import java.util.List;
@Getter
public class ConditionalView {

    private final Parent view;
    private final ConditionalController conditionalView;

    public ConditionalView() {

        conditionalView = new ConditionalController();
        this.view = loadView();
    }

    public ConditionalView(ConditionalController controller) {
        conditionalView = controller;
        this.view = loadView();
    }

    private Parent loadView() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/ConditionalScene.fxml"));
            loader.setController(conditionalView);
            return loader.load();
        } catch (Exception e) {
            throw new RuntimeException("Failed to load device view", e);
        }
    }
}
```

```
package org.example.client.view;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import lombok.Getter;
import org.example.client.controllers.ExpressionsContainerController;

import java.util.List;

@Getter
public class ExpressionsContainerView {
    private  List<Object> objects;
    private final Parent view;
    private final ExpressionsContainerController expressionsContainerController;

    public ExpressionsContainerView(List<Object> objects) {
        this.objects = objects;
        expressionsContainerController = new ExpressionsContainerController();
        this.view = loadView();
    }

    public ExpressionsContainerView(ExpressionsContainerController
```

```
expressionsContainerController) {
        this. expressionsContainerController = expressionsContainerController;
        this.view = loadView();
    }

    private Parent loadView() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/expressionsContainer.fxml"));
            loader.setController(expressionsContainerController);
            return loader.load();
        } catch (Exception e) {
            throw new RuntimeException("Failed to load device view", e);
        }
    }

}
```

```
package org.example.client.view;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import lombok.Getter;
import org.example.client.controllers.ExpressionController;

@Getter
public class ExpressionView {

    private Object objectModel;
    private final Parent view;
    private final ExpressionController expressionModelView;

    public ExpressionView(Object objectModel, String lastOperator) {
        this.objectModel = objectModel;
        expressionModelView = new ExpressionController(objectModel, lastOperator);
        this.view = loadView();
    }

    public ExpressionView(ExpressionController expressionModelView) {
        this.expressionModelView = expressionModelView;
        this.view = loadView();
    }

    private Parent loadView() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/Expressions.fxml"));
            loader.setController(expressionModelView);
            return loader.load();
        } catch (Exception e) {
            throw new RuntimeException("Failed to load device view", e);
        }
    }

}

package org.example.client.view;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import lombok.Data;
import org.example.client.controllers.FullDeviceController;
```

```
@Data
public class FullDeviceView {
    private Parent view;
    private FullDeviceController controller;

    public FullDeviceView(Object object) {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/fullDevice.fxml"));
            this.view = loader.load();
            this.controller = loader.getController();
            controller.setDevice(object);
        } catch (Exception e) {
            throw new RuntimeException("Failed to load view", e);
        }

    }
}
```

```
package org.example.client.view;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import lombok.Data;
import org.example.client.controllers.MapController;

import java.io.IOException;
import java.util.List;

@Data
public class MapView {

    private Parent view;
    private MapController controller;

    public MapView(List<Object> objects) {
        FXMLLoader loader = new FXMLLoader(getClass().getResource("/map.fxml"));
        try {
            this.view = loader.load();
            this.controller = loader.getController();
            controller.setDevices(objects);
        } catch (IOException e) {
            throw new RuntimeException("Failed to load view", e);
        }
    }
}
```

```
package org.example.client.view;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import lombok.Data;
import org.example.client.controllers.RuleContainerController;

@Data
public class RuleContainerSceneView {
    private Parent view;
    private RuleContainerController controller;
```

```
    public RuleContainerSceneView() {
        this.controller = new RuleContainerController();
        this.view = loadView();
    }

    public RuleContainerSceneView(RuleContainerController controller) {
        this.controller = controller;
        this.view = loadView();
    }

    private Parent loadView() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/RuleContainer.fxml"));
            loader.setController(controller);
            return loader.load();
        } catch (Exception e) {
            throw new RuntimeException("Failed to load device view", e);
        }
    }
}
```

```
package org.example.client.view;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import lombok.Data;
import org.example.client.controllers.RuleBuilderSceneController;

@Data
public class RuleSceneView {
    private Parent view;
    private RuleBuilderSceneController controller;

    public RuleSceneView(RuleBuilderSceneController controller){
        this.controller = controller;
        this.view = loadView();
    }

    public RuleSceneView(){
        this.controller = new RuleBuilderSceneController();
        this.view = loadView();

    }

    private Parent loadView() {
        try {
            FXMLLoader loader = new
FXMLLoader(getClass().getResource("/RuleBuilderScene.fxml"));
            loader.setController(controller);
            return loader.load();
        } catch (Exception e) {
            throw new RuntimeException("Failed to load device view", e);
        }
    }
}
```

```java
package org.example.client.view;

import javafx.scene.Parent;
import lombok.Data;
import lombok.EqualsAndHashCode;
import org.example.client.controllers.RulesSceneController;

@EqualsAndHashCode(callSuper = true)
@Data
public class RulesSceneView extends View {
    private Parent view;
    private RulesSceneController controller;

    public RulesSceneView() {
        controller = new RulesSceneController();
        view = loadView(controller, "/RulesScene.fxml");
    }
}
```

```java
package org.example.client.view;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;

public abstract class View {
    protected Parent loadView(Object controller, String uriVIew) {
        try {
            FXMLLoader loader = new FXMLLoader(getClass().getResource(uriVIew));
            loader.setController(controller);
            return loader.load();
        } catch (Exception e) {
            throw new RuntimeException("Failed to load device view", e);
        }
    }
}
```

```java
package org.example.client;

import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import org.example.client.controllers.ConditionalController;
import org.example.client.controllers.RuleBuilderSceneController;
import org.example.client.controllers.ExpressionsContainerController;

import java.io.IOException;
import java.util.List;

public class SceneManager {
    public Parent getThirdScene(List<Object> objects) {
        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/expressionsContainer.fxml"));
        ExpressionsContainerController controller = new
ExpressionsContainerController();
        loader.setController(controller);
        return getLoad(loader);
    }

    public Parent getConditionalScene(List<Object> objects){
        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/ConditionalScene.fxml"));
        ConditionalController controller = new ConditionalController();
        loader.setController(controller);
```

```
        return getLoad(loader);
    }

    Parent loadFXML(String resource) {
        return getLoad(new FXMLLoader(getClass().getResource(resource)));
    }


    private Parent getLoad(FXMLLoader loader) {
        try {
            return loader.load();
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    public Parent getRuleBuilder(List<Object> objects) {
        FXMLLoader loader = new
FXMLLoader(getClass().getResource("/RuleBuilderScene.fxml"));
        RuleBuilderSceneController controller = new RuleBuilderSceneController();
        loader.setController(controller);
        return getLoad(loader);
    }
}
```

```
package org.example.client;

import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Menu;
import javafx.scene.control.MenuBar;
import javafx.scene.control.MenuItem;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import org.example.App;
import org.example.client.view.MapView;
import org.example.client.view.RuleSceneView;
import org.example.client.view.RulesSceneView;


import java.util.List;


public class UIManager {
    private final Stage primaryStage;
    private VBox mainLayout;
    SceneManager sceneManager;

    public UIManager(Stage primaryStage) {
        this.primaryStage = primaryStage;
        sceneManager = new SceneManager();
    }

    public void init() {
        mainLayout = new VBox();
        MenuBar menuBar = createMenuBar();
        mainLayout.getChildren().add(menuBar);

        Scene scene = new Scene(mainLayout);
        primaryStage.setScene(scene);
        primaryStage.setTitle("EECSystem");
        primaryStage.show();
    }
```

```
    public void showMainScene() {
        Parent mainContent = sceneManager.loadFXML("/mainScene.fxml");
        updateContent(mainContent);
        primaryStage.setTitle("Main window");
    }

    public void showRuleBuilder(List<Object> objects) {
        Parent secondContent = sceneManager.getRuleBuilder(objects);
        updateContent(secondContent);
    }

    public void  showRules(){
        RulesSceneView rulesSceneView= new RulesSceneView();
        Parent rulesScene = rulesSceneView.getView();
        updateContent(rulesScene);
    }

    public void showMap(List<Object> objects){
        MapView MapView = new MapView(objects);
        Parent MapContent = MapView.getView();
        updateContent(MapContent);
    }


    private void updateContent(Parent newContent) {
        if (mainLayout.getChildren().size() > 1) {
            mainLayout.getChildren().remove(1);
        }
        mainLayout.getChildren().add(newContent);
    }

    private MenuBar createMenuBar() {
        MenuBar menuBar = new MenuBar();
        Menu menuWindows = new Menu("Windows");

        MenuItem mainSceneItem = new MenuItem("Main window");
        mainSceneItem.setOnAction(e -> showMainScene());

        MenuItem ruleConstructorItem = new MenuItem("Rule builder");
        ruleConstructorItem.setOnAction(e -> showRuleBuilder(App.getObjects()));

        MenuItem mapItem = new MenuItem("Device Map");
        mapItem.setOnAction(e -> showMap(App.getObjects()));

        MenuItem rulesItem = new MenuItem("Rules");
        rulesItem.setOnAction(e->showRules());

        menuWindows.getItems().addAll(mapItem, rulesItem, ruleConstructorItem);
        menuBar.getMenus().add(menuWindows);

        return menuBar;
    }
}
```

```
package org.example.entity;

import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;


import java.time.Instant;
import java.util.Objects;
```

```
public class CurrentLineSensor extends Device {
    private final DoubleProperty voltage = new SimpleDoubleProperty();
    private final DoubleProperty current = new SimpleDoubleProperty();

    public CurrentLineSensor() {
        deviceType = DeviceType.CURRENT_LINE_SENSOR;
    }

    public CurrentLineSensor(String deviceId, Double voltage, Double current) {
        setSensorId(deviceId);
        setCurrent(current);
        this.voltage.set(voltage);
        deviceType = DeviceType.CURRENT_LINE_SENSOR;
    }

    public double getCurrent() {
        return current.get();
    }

    public void setCurrent(double current) {
        this.current.set(current);
    }

    public double getVoltage() {
        return voltage.get();
    }

    public void setVoltage(double voltage) {
        this.voltage.set(voltage);
    }

    @Override
    public String toString() {
        return "CurrentLineSensor{" +
                "idDevice=" + getSensorId() +
                "current=" + getCurrent() +
                ", voltage=" + getVoltage() +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        CurrentLineSensor that = (CurrentLineSensor) o;
        return Objects.equals(getVoltage(), that.getVoltage()) &&
                Objects.equals(getCurrent(), that.getCurrent()) &&
                Objects.equals(getSensorId(), that.getSensorId());
    }

    @Override
    public int hashCode() {
        return Objects.hash(voltage, current);
    }
}
```

```
package org.example.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import javafx.beans.property.SimpleStringProperty;
import javafx.beans.property.StringProperty;
import lombok.Getter;
```

```
import lombok.Setter;
import org.example.service.DeviceUpdater;
import org.example.service.InflexDBService;
import org.example.subscriber.MQTTPayloadBuilder;
import org.example.utlity.ClassMethodService;

import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.time.Instant;
import java.util.Optional;


public abstract class Device {
    private final StringProperty sensorId = new SimpleStringProperty();
    private final StringProperty title = new SimpleStringProperty();
    @Getter
    @Setter
    protected DeviceType deviceType;

    public void updateDevice(){
        DeviceUpdater deviceUpdater = new DeviceUpdater();
        deviceUpdater.updateDevice(this,deviceType, getSensorId());
    }

    @JsonIgnore
    public String toInfluxDBLineProtocol(){
        String measurement = deviceType.getMeasurement();
        String tags = "device=" + getSensorId();
        String fields = getInfluxDBFields(this);
        String timestamp = String.valueOf(Instant.now().getEpochSecond());
        return measurement + "," + tags + " " + fields + " " + timestamp;
    }

    private String getInfluxDBFields(Device device) {
        Field[] fields = this.getClass().getDeclaredFields();
        StringBuilder stringBuilder = new StringBuilder();
        for (int posF = 0 ; posF<fields.length; posF++){
            Field field = fields[posF];
            ClassMethodService classMethodService = new ClassMethodService();
            Method method =
classMethodService.getGetterMethodByFieldName(field.getName(),this.getClass());
            Object value;
            try {
                value = method.invoke(device);
            } catch (IllegalAccessException | InvocationTargetException e) {
                throw new RuntimeException(e);
            }
            String mqttFieldString = String.format("%s=%s",field.getName(),
value);
            stringBuilder.append(mqttFieldString);
            if (posF < fields.length-1){
                stringBuilder.append(",");
            }
        }
        return stringBuilder.toString();
    }



    @JsonIgnore
    public String getMqttPayload() throws RuntimeException {
        MQTTPayloadBuilder mqttPayloadBuilder = new MQTTPayloadBuilder();
        return mqttPayloadBuilder.getPayloadBuilder(this);
```

```
    }

    @JsonIgnore
    public String getMqttTopic() {
        return String.format("client/%s/%s/data", deviceType.getMeasurement(),
getSensorId());
    }

    public String getSensorId() {
        return sensorId.get();
    }

    public void setSensorId(String sensorId) {
        this.sensorId.set(sensorId);
    }

    public String getTitle() {
        return title.get();
    }

    public void setTitle(String title) {
        this.title.set(title);
    }
}
```

```
package org.example.entity;

import lombok.Getter;

import java.util.function.Supplier;

public enum DeviceType {
    TRANSPORTER("transformer", Transformer::new),
    CURRENT_LINE_SENSOR("currentsensor", CurrentLineSensor::new),
    GENERATOR("generator", Generator::new),
    SWITCH_BOARD("switchboard", SwitchBoard::new),
    WIND_SENSOR("windsensor", WindSensor::new);
    @Getter
    private final String measurement;
    private final Supplier<Device> deviceCreator;

    DeviceType(String measurement, Supplier<Device> deviceCreator) {
        this.measurement = measurement;
        this.deviceCreator = deviceCreator;
    }

    public Device createDevice() {
        return deviceCreator.get();
    }


    public static DeviceType fromType(String typeString) {
        for (DeviceType type : values()) {
            if (type.measurement.equals(typeString)) {
                return type;
            }
        }
        return null;
    }
}
```

```java
package org.example.entity;

import javafx.beans.property.BooleanProperty;
import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleBooleanProperty;
import javafx.beans.property.SimpleDoubleProperty;

import java.util.Objects;

public class Generator extends Device {
    BooleanProperty isWorking = new SimpleBooleanProperty();
    DoubleProperty voltage = new SimpleDoubleProperty();
    DoubleProperty current = new SimpleDoubleProperty();

    public Generator() {
        super.deviceType = DeviceType.GENERATOR;
    }

    public Generator(String id, double voltage, double current, boolean isWorking)
{
        this.setSensorId(id);
        this.isWorking.set(isWorking);
        this.voltage.set(voltage);
        this.current.set(current);
        super.deviceType = DeviceType.GENERATOR;
    }
    public double getVoltage() {
        return voltage.get();
    }

    public void setVoltage(double voltage) {
        this.voltage.set(voltage);
    }

    public double getCurrent() {
        return current.get();
    }

    public void setCurrent(double current) {
        this.current.set(current);
    }

    public boolean getIsWorking() {
        return isWorking.get();
    }

    public void setIsWorking(Boolean isWorking) {
        this.isWorking.set(isWorking);
    }

    @Override
    public String toString() {
        return "Generator{" +
                "id=" + super.getSensorId() +
                " voltage=" + getVoltage() +
                ", current=" + getCurrent() +
                ", isWorking=" + getIsWorking() +
                ", measurement=" + deviceType +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
```

```
        if (o == null || getClass() != o.getClass()) return false;
        Generator generator = (Generator) o;
        return isWorking == generator.isWorking &&
                Double.compare(getVoltage(), generator.getVoltage()) == 0 &&
                Double.compare(getCurrent(), generator.getCurrent()) == 0;
    }

    @Override
    public int hashCode() {
        return Objects.hash(isWorking, voltage, current);
    }
}
```

```
package org.example.entity;

import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;

import java.util.Objects;

public class SwitchBoard extends Device {
    DoubleProperty current = new SimpleDoubleProperty();

    public SwitchBoard() {
        deviceType = DeviceType.SWITCH_BOARD;

    }

    public SwitchBoard(String id, double current) {
        super.setSensorId(id);
        this.current.set(current);
        deviceType = DeviceType.SWITCH_BOARD;
    }

    public double getCurrent() {
        return current.get();
    }

    public DoubleProperty currentProperty() {
        return current;
    }

    public void setCurrent(double current) {
        this.current.set(current);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SwitchBoard that = (SwitchBoard) o;
        return Objects.equals(current, that.current);
    }

    @Override
    public int hashCode() {
        return Objects.hashCode(current);
    }

    @Override
    public String toString() {
        return "SwitchBoard{" +
                "id=" + super.getSensorId() +
```

```
                "current=" + current +
                '}';
    }
}
```

```
package org.example.entity;

import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;

import java.util.Objects;

public class SwitchBoard extends Device {
    DoubleProperty current = new SimpleDoubleProperty();

    public SwitchBoard() {
        deviceType = DeviceType.SWITCH_BOARD;

    }

    public SwitchBoard(String id, double current) {
        super.setSensorId(id);
        this.current.set(current);
        deviceType = DeviceType.SWITCH_BOARD;
    }

    public double getCurrent() {
        return current.get();
    }

    public DoubleProperty currentProperty() {
        return current;
    }

    public void setCurrent(double current) {
        this.current.set(current);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        SwitchBoard that = (SwitchBoard) o;
        return Objects.equals(current, that.current);
    }

    @Override
    public int hashCode() {
        return Objects.hashCode(current);
    }

    @Override
    public String toString() {
        return "SwitchBoard{" +
                "id=" + super.getSensorId() +
                "current=" + current +
                '}';
    }
}
```

```
package org.example.entity;
```

```
import javax.persistence.*;
import java.time.Instant;

@Entity
@Table(name = "Topic", schema = "public")
public class Topic {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE, generator =
"Topic_id_topic_seq")
    @SequenceGenerator(name = "Topic_id_topic_seq", sequenceName =
"Topic_id_topic_seq", allocationSize = 1)
    @Column(name = "id_topic")
    private Integer idTopic;

    @Column(name = "title", nullable = false)
    private String title;

    @Column(name = "data", nullable = false)
    private Instant data;

    // Constructors
    public Topic() {}

    public Topic(String title, Instant data) {
        this.title = title;
        this.data = data;
    }

    // Getters and Setters
    public Integer getIdTopic() {
        return idTopic;
    }

    public void setIdTopic(Integer idTopic) {
        this.idTopic = idTopic;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public Instant getData() {
        return data;
    }

    public void setData(Instant data) {
        this.data = data;
    }

    // toString, equals, hashCode
    @Override
    public String toString() {
        return "Topic{" +
                "idTopic=" + idTopic +
                ", title='" + title + '\'' +
                ", data=" + data +
                '}';
    }
}
```

```java
package org.example.entity;

import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;

import lombok.EqualsAndHashCode;


import java.time.Instant;
import java.util.Objects;

@EqualsAndHashCode(callSuper = true)
public class Transformer extends Device {


    private final DoubleProperty turnsRatio = new SimpleDoubleProperty();

    public Transformer() {
        super.deviceType = DeviceType.TRANSPORTER;
    }

    public Transformer(String sensorId, Double turnsRatio) {
        super.setSensorId(sensorId);
        this.turnsRatio.set(turnsRatio);
        super.deviceType = DeviceType.TRANSPORTER;
    }


    public double getTurnsRation() {
        return turnsRatio.get();
    }

    public void setTurnsRation(Double turnsRatio) {
        this.turnsRatio.set(turnsRatio);
    }

    @Override
    public String toString() {
        return "Transformer{" +
                "id='" + super.getSensorId() +
                "' turnsRation='" + turnsRatio.get() + '\'' +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Transformer that = (Transformer) o;
        return Objects.equals(getTurnsRation(), that.getTurnsRation()) &&
Objects.equals(getSensorId(), that.getSensorId());
    }

    @Override
    public int hashCode() {
        return Objects.hashCode(turnsRatio);
    }


}
```

```java
package org.example.entity;
```

```
import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;
import lombok.ToString;


public class WindSensor extends Device {


    private final DoubleProperty windSpeed = new SimpleDoubleProperty();

    public WindSensor() {
        super.deviceType = DeviceType.WIND_SENSOR;
        System.out.println(1);
        setSensorId("1");
        this.setWindSpeed(0.0);
    }

    public WindSensor(double windSpeed) {
        this.setWindSpeed(windSpeed);
        super.deviceType = DeviceType.WIND_SENSOR;
    }

    public double getWindSpeed() {
        return windSpeed.get();
    }


    public void setWindSpeed(double windSpeed) {
        try {
            if (windSpeed < 0) {
                throw new IllegalArgumentException("Скорость вітру не може бути
негативною.");
            }
            this.windSpeed.set(windSpeed);
        }catch (Exception e){
            System.out.println(e);
        }

    }

    @Override
    public String toString() {
        return "WindSensor{" +
                "deviceId=" + getSensorId() +
                ", windSpeed=" + getWindSpeed() +
                ", deviceType=" + deviceType +
                '}';
    }

    public void updateSensorData(double newWindSpeed) {
        setWindSpeed(newWindSpeed);
    }

}
```

```
package org.example.factory;

import org.example.entity.*;


import java.util.Optional;
import java.util.logging.Level;
import java.util.logging.Logger;
```

```
public class DeviceFactory {
    public static Optional<Device> getDevice(DeviceType measurement){
        Device device = null;
        switch (measurement){
            case DeviceType.CURRENT_LINE_SENSOR:
                device = new CurrentLineSensor();
                break;
            case DeviceType.TRANSPORTER:
                device = new Transformer();
                break;
            case GENERATOR:
                device = new Generator();
                break;
            case SWITCH_BOARD:
                device = new SwitchBoard();
                break;
            default:
                Logger.getLogger(DeviceFactory.class.getName()).log(Level.SEVERE,
"error sensors factory");
        }
        return Optional.ofNullable(device);
    }
}
```

```
package org.example.factory;

import org.example.service.postgres.RuleDataRepository;

public class RuleDataRepositoryFactory {
    public static RuleDataRepository getInstance() {
        return new RuleDataRepository();
    }
}
```

```
package org.example.factory;

import org.example.service.postgres.TopicRepository;

public class TopicRepositoryFactory {
    public static TopicRepository getInstance() {
        return new TopicRepository();
    }
}
```

```
package org.example.rule.entity;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import lombok.Data;
import lombok.ToString;
import org.example.rule.serializer.TargetObjectDeserializer;

import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.util.NoSuchElementException;


@ToString
```

```
@Data
public class Action {
    @JsonDeserialize(using = TargetObjectDeserializer.class)
    private Object targetObject;
    private String fieldName;
    private Expression expression;


    public Action() {
    }

    public Action(Object targetObject, String fieldName, Expression expression) {
        this.targetObject = targetObject;
        this.fieldName = fieldName;
        this.expression = expression;
    }

}
```

```
package org.example.rule.entity;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import lombok.Data;
import lombok.ToString;
import org.example.rule.serializer.TargetObjectDeserializer;

import java.lang.reflect.Method;
import java.util.Objects;

@ToString
@Data
public class Condition {
    @JsonDeserialize(using = TargetObjectDeserializer.class)
    private Object targetObject;
    private String fieldName;
    private String operator;
    private Object value;


    public Condition() {
    }

    public Condition(Object targetObject, String fieldName, String operator,
Object value) {
        this.targetObject = targetObject;
        this.fieldName = fieldName;
        this.operator = operator;
        this.value = value;
    }

    public boolean evaluate() throws Exception {
        Object fieldValue = getFieldValue();
        return compareValues(fieldValue);
    }

    private Object getFieldValue() throws Exception {
        Method field = targetObject.getClass().getDeclaredMethod(fieldName);
        field.setAccessible(true);
        return field.invoke(targetObject);
    }

    private boolean compareValues(Object fieldValue) {
        value = convertValueToFieldType(fieldValue, value);
```

```
        switch (operator) {
            case "==":
                return isEqual(fieldValue, value);
            case "!=":
                return isNotEqual(fieldValue, value);
            case ">":
                return isGreaterThan(fieldValue, value);
            case "<":
                return isLessThan(fieldValue, value);
            default:
                throw new IllegalArgumentException("Unsupported operator: " +
operator);
        }
    }

    private Object convertValueToFieldType(Object fieldValue, Object rawValue) {
        if (fieldValue == null || rawValue == null) {
            return rawValue;
        }
        Class<?> fieldClass = fieldValue.getClass();
        if (fieldClass.equals(Double.class)) {
            return Double.parseDouble(rawValue.toString());
        } else if (fieldClass.equals(Integer.class)) {
            return Integer.parseInt(rawValue.toString());
        } else if (fieldClass.equals(Boolean.class)) {
            return Boolean.parseBoolean(rawValue.toString());
        } else if (fieldClass.equals(String.class)) {
            return rawValue.toString();
        } else {
            throw new IllegalArgumentException("Unsupported field type: " +
fieldClass);
        }
    }

    private boolean isEqual(Object fieldValue, Object comparisonValue) {
        return fieldValue.equals(comparisonValue);
    }

    private boolean isNotEqual(Object fieldValue, Object comparisonValue) {
        return !fieldValue.equals(comparisonValue);
    }

    private boolean isGreaterThan(Object fieldValue, Object comparisonValue) {
        if (!(fieldValue instanceof Comparable)) {
            throw new IllegalArgumentException("Field value is not comparable: " +
fieldValue);
        }
        return ((Comparable<Object>) fieldValue).compareTo(comparisonValue) > 0;
    }

    private boolean isLessThan(Object fieldValue, Object comparisonValue) {
        if (!(fieldValue instanceof Comparable)) {
            throw new IllegalArgumentException("Field value is not comparable: " +
fieldValue);
        }
        return ((Comparable<Object>) fieldValue).compareTo(comparisonValue) < 0;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Condition condition = (Condition) o;
        return Objects.equals(targetObject, condition.targetObject) &&
```

```
Objects.equals(fieldName, condition.fieldName) && Objects.equals(operator,
condition.operator) && Objects.equals(value, condition.value);
    }

    @Override
    public int hashCode() {
        return Objects.hash(targetObject, fieldName, operator, value);
    }


}
```

```
package org.example.rule.entity;

import lombok.Data;
import lombok.Getter;
import lombok.ToString;

import java.util.Objects;

@Data
@ToString
public class ConditionWithOperator {
    private Expression condition;
    private String logicalOperator;

    public ConditionWithOperator() {
    }

    public ConditionWithOperator(Expression condition, String logicalOperator) {
        this.condition = condition;
        this.logicalOperator = logicalOperator;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        ConditionWithOperator that = (ConditionWithOperator) o;
        return Objects.equals(condition, that.condition) &&
Objects.equals(logicalOperator, that.logicalOperator);
    }

    @Override
    public int hashCode() {
        return Objects.hash(condition, logicalOperator);
    }
}
```

```
package org.example.rule.entity;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import lombok.Data;

import lombok.ToString;
import org.example.client.service.PrimitiveParser;
import org.example.entity.Device;
import org.example.rule.serializer.TargetObjectDeserializer;
import org.example.service.inflexdb.InflexDBRepository;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
```

```
import java.util.Optional;


@Data
@ToString
public class Expression {

    @JsonDeserialize(using = TargetObjectDeserializer.class)
    private Object targetObject;
    private String methodName;
    private String operator;
    private Expression leftOperand;
    private Expression rightOperand;
    private boolean isGrouped;

    public Expression() {
    }

    public Expression(Object targetObject) {
        this.targetObject = targetObject;
        this.isGrouped = false;
        if (targetObject instanceof  Device) updateDevice(targetObject);
    }

    public Expression(Object targetObject, String methodName) {
        this.targetObject = targetObject;
        this.methodName = methodName;
        this.isGrouped = false;
        if (targetObject instanceof  Device) updateDevice(targetObject);
    }

    public void updateDevice(Object targetObject){
        Device device = (Device) targetObject;
        InflexDBRepository inflexDBRepository = new InflexDBRepository();
        Optional<Device> newDevice =
inflexDBRepository.getDeviceById(device.getDeviceType(), device.getSensorId());
        newDevice.ifPresent(value -> this.targetObject = value);
    }

    public Expression(Expression groupedExpression) {
        this.leftOperand = groupedExpression;
        this.isGrouped = true;
    }

    public Expression(Expression leftOperand, String operator, Expression
rightOperand) {
        this.leftOperand = leftOperand;
        this.operator = operator;
        this.rightOperand = rightOperand;
        this.isGrouped = false;
    }

    public Object getTargetObject() {
        if (targetObject instanceof  Device) updateDevice(targetObject);
        return targetObject;
    }

    public void setTargetObject(Object targetObject) {
        this.targetObject = targetObject;
        if (targetObject instanceof  Device) updateDevice(targetObject);
    }

    public Object evaluate() {
        if (isGrouped) {
```

```
            return evaluateGroupedExpression();
        }

        if (isExpressionWithOperand()) {
            Object leftValue = leftOperand.evaluate();
            Object rightValue = rightOperand.evaluate();
            return performOperation(leftValue, rightValue);
        }

        return getFieldValue();
    }

    private boolean isExpressionWithOperand() {
        return leftOperand != null && rightOperand != null;
    }

    private Object evaluateGroupedExpression()  {
        if (isExpressionWithOperand()) {
            Object leftValue = leftOperand.evaluate();
            Object rightValue = rightOperand.evaluate();
            return performOperation(leftValue, rightValue);
        }
        return getFieldValue();
    }

    private Object getFieldValue(){
        if (targetObject == null) {
            throw new IllegalArgumentException("Target object or method name is
missing");
        }
        if (methodName == null) {
            return targetObject;
        }
        if (targetObject instanceof Device device){
            InflexDBRepository inflexDBRepository = new InflexDBRepository();
            Optional<Device> newData =
inflexDBRepository.getDeviceById(device.getDeviceType(),device.getSensorId());
            newData.ifPresent(value -> targetObject = value);
        }
        Method field = null;
        try {
            field = targetObject.getClass().getDeclaredMethod(methodName);
            field.setAccessible(true);
            return field.invoke(targetObject);
        }

        catch (NoSuchMethodException | IllegalAccessException |
InvocationTargetException e) {
            throw new RuntimeException(e);
        }
    }
    private Object performOperation(Object leftValue, Object rightValue) {
        PrimitiveParser parser = new PrimitiveParser();
        leftValue = parser.parse(leftValue);
        rightValue = parser.parse(rightValue);

        if (leftValue instanceof Boolean && rightValue instanceof Boolean) {
            boolean left = (Boolean) leftValue;
            boolean right = (Boolean) rightValue;

            switch (operator) {
                case "&&":
                    return left && right;
                case "||":
```

```
                    return left || right;
                case "==":
                    return left == right;
                case "!=":
                    return left != right;
                default:
                    throw new IllegalArgumentException("Unsupported operator for
boolean values: " + operator);
            }
        } else if (leftValue instanceof Number && rightValue instanceof Number) {
            switch (operator) {
                case "+":
                    return ((Number) leftValue).doubleValue() + ((Number)
rightValue).doubleValue();
                case "-":
                    return ((Number) leftValue).doubleValue() - ((Number)
rightValue).doubleValue();
                case "*":
                    return ((Number) leftValue).doubleValue() * ((Number)
rightValue).doubleValue();
                case "/":
                    if (((Number) rightValue).doubleValue() == 0) {
                        throw new ArithmeticException("Cannot divide by zero");
                    }
                    return ((Number) leftValue).doubleValue() / ((Number)
rightValue).doubleValue();
                case "==":
                    return leftValue.equals(rightValue);
                case "!=":
                    return !leftValue.equals(rightValue);
                case ">":
                    return ((Comparable<Object>) leftValue).compareTo(rightValue)
> 0;
                case "<":
                    return ((Comparable<Object>) leftValue).compareTo(rightValue)
< 0;
                default:
                    throw new IllegalArgumentException("Unsupported operator: " +
operator);
            }
        } else {
            throw new IllegalArgumentException("Operands must be numbers,
booleans, or objects with comparable fields");
        }
    }

    private Object parseToBoolean(Object value) {
        if (value instanceof Boolean) {
            return value;
        } else if (value instanceof String) {
            String strValue = ((String) value).trim().toLowerCase();
            if ("true".equals(strValue)) {
                return true;
            } else if ("false".equals(strValue)) {
                return false;
            }
        } else if (value instanceof Number) {
            return ((Number) value).doubleValue() != 0;
        }
        return value;
    }
```

```
}
```

```
package org.example.rule.entity;

import com.fasterxml.jackson.annotation.JsonIgnore;
import lombok.Data;

import java.time.LocalDateTime;
import java.util.List;

@Data
public class Rule {
    @JsonIgnore
    private long id;
    private String description;
    private List<ConditionWithOperator> conditionsWithOperators;
    private List<Action> actions;
    @JsonIgnore
    private LocalDateTime timeStamp;

    public Rule() {
    }

    public Rule(List<ConditionWithOperator> conditionsWithOperators, List<Action>
actions) {
        this.conditionsWithOperators = conditionsWithOperators;
        this.actions = actions;
    }
}
```

```
package org.example.rule.executor;

import org.example.rule.entity.Action;

public interface ActionExecutor {
    void execute(Action action);
}
```

```
package org.example.rule.executor;

import org.example.entity.Device;
import org.example.rule.entity.Action;

public class ActionExecutorFactory {
    public static ActionExecutor getInstance(Action action){
        return new DefaultActionExecutor();
    }
}
```

```
package org.example.rule.executor;

import java.util.HashMap;
import java.util.Map;

public class ActionExecutorRegistry {
    private static final Map<Class<?>, ActionExecutor> registry = new HashMap<>();

    public static void registerExecutor(Class<?> clazz, ActionExecutor executor) {
```

```
        registry.put(clazz, executor);
    }

    public static ActionExecutor getExecutor(Class<?> clazz) {
        return registry.get(clazz);
    }
}
```

```
package org.example.rule.executor;

import org.example.rule.entity.Action;

public class CustomExecutorForSpecificObject implements ActionExecutor {
    @Override
    public void execute(Action action){
        System.out.println("Executing custom action for specific object: " +
action.getTargetObject());
    }
}
```

```
package org.example.rule.executor;

import org.example.entity.Device;
import org.example.factory.DeviceFactory;
import org.example.rule.entity.Action;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisher;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisherFactory;

import java.lang.reflect.InvocationTargetException;
import java.util.NoSuchElementException;
import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.util.Random;


public class DefaultActionExecutor implements ActionExecutor {
    @Override
    public void execute(Action action)  {
//        System.out.println("ActionExecute");
        Object targetObject = action.getTargetObject();
        String fieldName = action.getFieldName();
        Object newValue = action.getExpression().evaluate();
        if (newValue instanceof Double number && number > 140){
            System.out.println(action);
            System.out.println("newValue " + number);
        }

        Class<?> clazz = targetObject.getClass();
        try {
            Field field = clazz.getDeclaredField(fieldName);
            field.setAccessible(true);
            field.set(targetObject, newValue);
        } catch (NoSuchFieldException | IllegalAccessException e) {
            Method method = getDeclaredMethod(clazz, fieldName);
            method.setAccessible(true);
            invoke(method, targetObject, newValue);
        }
        if (targetObject instanceof Device device){
            Random random = new Random();
            MQTTPublisher mqttPublisher =
MQTTPublisherFactory.getPublisher(String.valueOf(random.nextInt()));
            mqttPublisher.writeData(device);
```

```
        }
    }

    private void invoke(Method method, Object targetObject, Object newValue) {
        try {
            System.out.println(method);
            System.out.println(targetObject);
            System.out.println(newValue);
            Class<?> parameterType = method.getParameterTypes()[0];
            Object convertedValue =
convertValueToMethodParameterType(parameterType, newValue);
            System.out.println(newValue);
            method.invoke(targetObject, convertedValue);
        } catch (IllegalAccessException | InvocationTargetException ex) {
            throw new RuntimeException(ex);
        }
    }

    private Object convertValueToMethodParameterType(Class<?> parameterType,
Object newValue) {
        if (newValue == null) {
            return null;
        }

        if (parameterType.equals(String.class)) {
            return newValue.toString();
        } else if (parameterType.equals(Double.class)) {
            return Double.valueOf(newValue.toString());
        } else if (parameterType.equals(Integer.class)) {
            return Integer.valueOf(newValue.toString());
        } else if (parameterType.equals(Boolean.class)) {
            return Boolean.valueOf(newValue.toString());
        } else {
            return newValue;
        }
    }


    private Method getDeclaredMethod(Class<?> clazz, String fieldName) {
        for (Method method : clazz.getDeclaredMethods()) {
            if (method.getName().equals(fieldName)) {
                return method;
            }
        }
        throw new NoSuchElementException("Метод з іменем '" + fieldName + "' не
знайдено у класі " + clazz.getName());
    }
}
```

```
package org.example.rule.executor;

import org.example.rule.entity.*;

import java.util.List;
import java.util.NoSuchElementException;

public class RuleExecutor {
    private final Rule rule;

    public RuleExecutor(Rule rule) {
        this.rule = rule;
```

```
    }

    public void execute() {
        List<Action> actions = rule.getActions();
        if (conditionsMet(0, true)) {
            System.out.println(rule.getDescription());
            for (Action action : actions) {
                performAction(action);
            }
        } else {
        }
    }

    private boolean conditionsMet(int index, boolean accumulatedResult) {
        List<ConditionWithOperator> conditionsWithOperators =
rule.getConditionsWithOperators();
        if (index >= conditionsWithOperators.size()) {
            return accumulatedResult;
        }

        ConditionWithOperator current = conditionsWithOperators.get(index);
        boolean conditionResult = evaluateCondition(current.getCondition());

        String logicalOperator = current.getLogicalOperator();
        boolean newResult;


        if ("AND".equalsIgnoreCase(logicalOperator)) {
            newResult = accumulatedResult && conditionResult;
        } else if ("OR".equalsIgnoreCase(logicalOperator)) {
            newResult = accumulatedResult || conditionResult;
        } else {
            throw new IllegalStateException("Невідомий логічний оператор: " +
logicalOperator);
        }

        return conditionsMet(index + 1, newResult);
    }

    private boolean evaluateCondition(Expression condition) {
        try {
            Object object = condition.evaluate();
            return (Boolean) object ;
        }catch (Exception e){
            throw new NoSuchElementException(e);
        }

    }

    private void performAction(Action action) {
        ActionExecutor actionExecutor = new DefaultActionExecutor();
        actionExecutor.execute(action);
    }
}
```

```
package org.example.rule.executor;

import org.example.rule.entity.Rule;
import org.example.service.RuleService;

public class RuleExecutorThread extends Thread{

    private Rule rule;
```

```
    public RuleExecutorThread(Rule rule) {
        this.rule = rule;
    }


    @Override
    public void run() {
        while (true){
            RuleExecutor ruleExecutor = new RuleExecutor(rule);
            ruleExecutor.execute();
            try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
            RuleService ruleService =new RuleService();
//            System.out.println(rule);
            rule = ruleService.getRuleDataById(rule.getId());
        }

    }
}
```

```
package org.example.rule.serializer;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.example.rule.entity.Rule;

public class RuleDeserializer {

    public Rule deserialize(String ruleSerialized) {
        ObjectMapper objectMapper = new ObjectMapper();
        try {
            return objectMapper.readValue(ruleSerialized, Rule.class);
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }
    }

}
```

```
package org.example.rule.serializer;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.example.rule.entity.Rule;

public class RuleSerializer {
    public String serializeRule(Rule rule){
        ObjectMapper objectMapper = new ObjectMapper();
        try {
            return objectMapper.writeValueAsString(rule);
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }
    }
}
```

```
package org.example.rule.serializer;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;
import com.fasterxml.jackson.databind.JsonNode;
import org.example.entity.*;
import org.example.factory.DeviceFactory;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisher;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisherFactory;
import org.jetbrains.annotations.NotNull;

import java.io.IOException;
import java.util.LinkedHashMap;

public class TargetObjectDeserializer extends JsonDeserializer<Object> {
    @Override
    public Object deserialize(JsonParser parser, DeserializationContext context)
throws IOException {
        JsonNode node = parser.getCodec().readTree(parser);


        if (node.isNumber()) {
            return node.numberValue();
        } else if (node.isTextual()) {
            return node.textValue();
        } else if (node.isBoolean()) {
            return node.booleanValue();
        }

        if (isGenerator(node)) {
            return getGenerator(node);
        } else if (isCurrentSensor(node)) {
            return getCurrentLineSensor(node);
        } else if (isTransformer(node)) {
            return getTransformer(node);
        } else if (isSwitchBoard(node)) {
            return getSwitchBoard(node);
        } else if (isMQTTPublisher(node)) {
            return getMqttPublisher(node);
        } else if (isWindSpeed(node)) {
            return getWindSensor(node);
        } else {
            return parser.getCodec().treeToValue(node, LinkedHashMap.class);
        }
    }

    private boolean isSwitchBoard(JsonNode node) {
        return node.has("sensorId") &&
                node.has("current");
    }

    private Object getSwitchBoard(JsonNode node) {
        String sensorId = getStringField(node, "sensorId");
        Double current = getDoubleField(node, "current");
        return new SwitchBoard(sensorId, current);
    }

    private boolean isGenerator(JsonNode node) {
        return node.has("sensorId") &&
                node.has("voltage") &&
                node.has("current") &&
                node.has("isWorking");
    }
```

```
    private Object getGenerator(JsonNode node) {
        String sensorId = getStringField(node, "sensorId");
        Double voltage = getDoubleField(node, "voltage");
        Double current = getDoubleField(node, "current");
        boolean isWorking = getBooleanField(node, "isWorking");
        return new Generator(sensorId, voltage, current, isWorking);
    }

    private boolean isCurrentSensor(JsonNode node) {
        return node.has("sensorId") &&
                node.has("voltage") &&
                node.has("current");
    }

    private @NotNull CurrentLineSensor getCurrentLineSensor(JsonNode node) {
        String sensorId = getStringField(node, "sensorId");
        Double voltage = getDoubleField(node, "voltage");
        Double current = getDoubleField(node, "current");
        return new CurrentLineSensor(sensorId, voltage, current);
    }

    private boolean isTransformer(JsonNode node) {
        return node.has("sensorId") &&
                node.has("turnsRation");
    }

    private @NotNull Transformer getTransformer(JsonNode node) {
        String sensorId = getStringField(node, "sensorId");
        Double turnsRatio = getDoubleField(node, "turnsRation");
        Transformer transformer = (Transformer)
DeviceFactory.getDevice(DeviceType.TRANSPORTER).get();
        transformer.setSensorId(sensorId);
        transformer.setTurnsRation(turnsRatio);
        return transformer;
    }

    private boolean isMQTTPublisher(JsonNode node) {
        return node.has("clientId");
    }

    private @NotNull MQTTPublisher getMqttPublisher(JsonNode node) {
        return MQTTPublisherFactory.getPublisher(getStringField(node,
"clientId"));
    }

    private boolean isWindSpeed(JsonNode node) {
        return node.has("windSpeed");
    }

    private Object getWindSensor(JsonNode node) {
        String sensorId = getStringField(node, "sensorId");
        Double windSpeed = getDoubleField(node, "windSpeed");
        WindSensor windSensor = new WindSensor(windSpeed);
        windSensor.setSensorId(sensorId);
        return windSensor;
    }

    private @NotNull Double getDoubleField(JsonNode node, String doubleFieldName)
{
        return Double.valueOf(getStringField(node, doubleFieldName));
    }

    private boolean getBooleanField(JsonNode node, String booleanFieldName) {
```

```
        return Boolean.parseBoolean(getStringField(node, booleanFieldName));
    }

    private String getStringField(JsonNode node, String sensorId) {
        return String.valueOf(node.get(sensorId)).replace("\"", "");
    }

}
```

```
package org.example.rule;

import org.example.client.controllers.ActionController;
import org.example.client.controllers.ConditionalController;
import org.example.client.ruleparser.UiActionsExtractor;
import org.example.client.ruleparser.UiConditionsExtractor;
import org.example.rule.entity.Action;
import org.example.rule.entity.ConditionWithOperator;
import org.example.rule.entity.Rule;

import java.util.ArrayList;
import java.util.List;

public class RuleBuilder {
    private List<ConditionWithOperator> conditionsWithOperators = new
ArrayList<>();
    private List<Action> actions = new ArrayList<>();

    public RuleBuilder addCondition(ConditionWithOperator conditionWithOperator) {
        if (conditionsWithOperators.isEmpty() &&
!"AND".equalsIgnoreCase(conditionWithOperator.getLogicalOperator())) {
            throw new IllegalStateException("Перша умова має починатися з
оператора 'AND'.");
        }
        conditionsWithOperators.add(conditionWithOperator);
        return this;
    }

    public RuleBuilder addAction(Action action) {
        System.out.println("Додавання дії: об'єкт =" + action.getTargetObject() +
", поле=" + action.getFieldName() + ", значення=" +
action.getExpression().evaluate());
        actions.add(action);
        return this;
    }

    public Rule build() {
        System.out.println("Побудова правила з" + conditionsWithOperators.size() +
" умови та " + actions.size() + " діями.");
        return new Rule(conditionsWithOperators, actions);
    }

    public RuleBuilder fromUi(ConditionalController conditionalControllers,
List<ActionController> actionControllers) {
        List<ConditionWithOperator> conditionsFromUi =
getConditionsFromUi(conditionalControllers);
        System.out.println("conditionsFromUi " +conditionsFromUi);
        List<Action> actionsFromUi = getActionsFromUi(actionControllers);
        conditionsFromUi.forEach(this::addCondition);
        actionsFromUi.forEach(this::addAction);
        return this;
    }

    private List<Action> getActionsFromUi(List<ActionController> actionsContainer)
```

```
{
        UiActionsExtractor uiActionsExtractor = new UiActionsExtractor();
        return uiActionsExtractor.extractActions(actionsContainer);
    }

    private List<ConditionWithOperator> getConditionsFromUi(ConditionalController
conditionsContainer) {
        UiConditionsExtractor uiConditionsExtractor = new UiConditionsExtractor();
        return new
ArrayList<>(uiConditionsExtractor.extractConditions(conditionsContainer));
    }



}
```

```
package org.example.service.hibernate;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class EntityManagerUtil {
    private static final EntityManagerFactory emf =
Persistence.createEntityManagerFactory("my-persistence-unit");
    private static final ThreadLocal<EntityManager> threadLocal = new
ThreadLocal<>();

    public static EntityManager getEntityManager() {
        EntityManager em = threadLocal.get();
        if (em == null || !em.isOpen()) {
            em = emf.createEntityManager();
            threadLocal.set(em);
        }
        em.clear();
        return em;
    }

    public static void closeEntityManager() {
        EntityManager em = threadLocal.get();
        if (em != null) {
            em.close();
            threadLocal.remove();
        }
    }

    public static void closeEntityManagerFactory() {
        if (emf.isOpen()) {
            emf.close();
        }
    }
}
```

```
package org.example.service.hibernate;

import lombok.Getter;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.cfg.Configuration;
```

```
public class HibernateUtil {
    @Getter
    private static final SessionFactory sessionFactory = buildSessionFactory();
    private static final ThreadLocal<Session> threadLocal = new ThreadLocal<>();

    private static SessionFactory buildSessionFactory() {
        try {
            return new Configuration().configure("в.xml").buildSessionFactory();
        } catch (Throwable ex) {
            throw new ExceptionInInitializerError(ex);
        }
    }

    public static Session getSession() {
        Session session = threadLocal.get();
        if (session == null || !session.isOpen()) {
            session = sessionFactory.openSession();
            threadLocal.set(session);

        }
        return session;
    }

    public static void closeSession() {
        Session session = threadLocal.get();
        if (session != null) {
            session.close();
            threadLocal.remove();
        }
    }

}
```

```
package org.example.service.inflexdb;

import org.example.entity.Device;
import org.example.utlity.ClassMethodService;

import java.lang.reflect.Method;

public class DeviceHandler {

    public static void handleField(Device device, String field, Object value) {
        try {
            ClassMethodService classMethodService = new ClassMethodService();
            String titleSetterMethod =
classMethodService.getSetterTitleByField(field);
            Class<? extends Device> deviceClass = device.getClass();
            Method setterMethod = classMethodService.getMethod(deviceClass,
titleSetterMethod);
            setterMethod.invoke(device, value);
        } catch (Exception e) {
            System.err.println("Помилка при обробці поля " + field + ": " +
e.getMessage());
        }
    }


}
```

```java
package org.example.service.inflexdb;

import com.influxdb.client.InfluxDBClient;

import com.influxdb.client.InfluxDBClientFactory;
import com.influxdb.client.QueryApi;
import com.influxdb.client.WriteApiBlocking;
import com.influxdb.client.domain.WritePrecision;
import com.influxdb.query.FluxRecord;
import com.influxdb.query.FluxTable;
import org.example.entity.Device;
import org.example.entity.DeviceType;
import org.jetbrains.annotations.Nullable;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Optional;


public class InflexDBRepository {
    private final String URL = "http://localhost:8086";
    private final String TOKEN = "UzGyKeMA-1Vy1mMNGWjdQSCbIi4MUzfdSlO0J2iVBCqXH-
LJOI6pn5MYN9Nh1hAzKTaArKIAz3B3A_9Ce9iePQ==";
    private final String ORG = "nure";
    private final String BUCKET = "eecsystem";
    private final InfluxDBClient CLIENT = InfluxDBClientFactory.create(URL,
TOKEN.toCharArray(), ORG, BUCKET);
    private final WriteApiBlocking WRITE_API = CLIENT.getWriteApiBlocking();

    public void writeData(Device device) {
        String lineProtocol = device.toInfluxDBLineProtocol();
        WRITE_API.writeRecord(WritePrecision.S, lineProtocol);
    }

    public List<Device> getAllDevices() {
        QueryApi queryApi = CLIENT.getQueryApi();

        String fluxQuery = getAllDeviceFluxQuery();
        List<FluxTable> tables = queryApi.query(fluxQuery);

        Map<String, Device> deviceMap = new HashMap<>();
        for (FluxTable table : tables) {
            for (FluxRecord record : table.getRecords()) {
                String measurement = record.getMeasurement();
                String deviceId = (String) record.getValueByKey("device");
                String field = (String) record.getValueByKey("_field");
                Object value = record.getValueByKey("_value");
                String keyMap = String.format("%s/%s", measurement, deviceId);
                Device device = deviceMap.get(keyMap);
                device = getOrCreateDevice(device, measurement);
                if (device != null) {
                    device.setSensorId(deviceId);
                    deviceMap.put(keyMap, device);
                }
                handleDevice(device, field, value, deviceId);
            }
        }
        return deviceMap.values().stream().toList();
    }

    private String getAllDeviceFluxQuery() {
        QueryCreator queryCreator = new QueryCreator();
        return  queryCreator.getAllDeviceFluxQuery(BUCKET);
```

```
    }
private String getDeviceByIdFluxQuery(String measurement, String deviceId) {
        QueryCreator queryCreator = new QueryCreator();
        return  queryCreator.getDeviceByIdFluxQuery(BUCKET, measurement,
deviceId);
    }


    private void handleDevice(Device device, String field, Object value, String
deviceId) {
        if (device != null) {
            DeviceHandler.handleField(device, field, value);
        } else {
            System.err.println("Пристрій з deviceId " + deviceId + " не знайдено
або не створено.");
        }
    }

    private @Nullable Device getOrCreateDevice(Device device, String measurement)
{
        if (device == null) {
            DeviceType deviceType = DeviceType.fromType(measurement);
            if (deviceType != null) {
                device = deviceType.createDevice();
            }
        }
        return device;
    }

    public Optional<Device> getDeviceById(DeviceType deviceType, String sensorId)
{
        QueryApi queryApi = CLIENT.getQueryApi();
        String fluxQuery = getDeviceByIdFluxQuery(deviceType.getMeasurement(),
sensorId);
        List<FluxTable> tables = queryApi.query(fluxQuery);
        Device device = null;

        for (FluxTable table : tables) {
            for (FluxRecord record : table.getRecords()) {
                String recordMeasurement = record.getMeasurement();
                String recordDeviceId = (String) record.getValueByKey("device");
                String field = (String) record.getValueByKey("_field");
                Object value = record.getValueByKey("_value");


                if (recordDeviceId.equals(sensorId)) {
                    device = getOrCreateDevice(device, recordMeasurement);
                    if (device != null) {
                        device.setSensorId(recordDeviceId);
                    }
                    handleDevice(device, field, value, recordDeviceId);
                }
            }
        }

        return Optional.ofNullable(device);
    }
}
```

```
package org.example.service.inflexdb;

import org.example.entity.DeviceType;
import org.jetbrains.annotations.NotNull;
```

```
import java.util.stream.Collectors;
import java.util.stream.Stream;

public class QueryCreator {
    public @NotNull String getAllDeviceFluxQuery(String bucket) {
        return String.format(
                "from(bucket: \"%s\") " +
                        "|> range(start: -30d) " +
                        "|> filter(fn: (r) => %s) " +
                        "|> last()",
                bucket,
                Stream.of(DeviceType.values())
                        .map(this::getMeasurement)
                        .collect(Collectors.joining(" or "))
        );
    }


    public @NotNull String getDeviceByIdFluxQuery(String bucket, String
measurement, String deviceId ){
        return String.format("from(bucket: \"%s\") " +
                "  |> range(start: -30d)" +
                "  |> filter(fn: (r) => r[\"_measurement\"] == \"%s\")" +
                "  |> filter(fn: (r) => r[\"device\"] == \"%s\")" +
                "  |> last()", bucket, measurement, deviceId);
    }

    private @NotNull String getMeasurement(DeviceType deviceType) {
        return String.format("r[\"_measurement\"] == \"%s\"",
deviceType.getMeasurement());
    }
}
```

```
package org.example.service.inflexdb;

public class ValueConverter {

    public static Object convertValue(Object value, Class<?> targetType) {
        if (value == null) {
            return null;
        }
        if (targetType == Double.class || targetType == double.class) {
            return ((Number) value).doubleValue();
        } else if (targetType == Integer.class || targetType == int.class) {
            return ((Number) value).intValue();
        } else if (targetType == String.class) {
            return value.toString();
        }
        return value;
    }
}
```

```
package org.example.service.postgres.entity;

import lombok.Data;
import lombok.Getter;
import lombok.Setter;

import javax.persistence.*;
import java.time.LocalDateTime;
```

```
@Data
@Entity
public class RuleData {


    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "description", columnDefinition = "text")
    private String description;
    @Column(name = "data", columnDefinition = "text", nullable = false)
    private String data;

    @Column(name = "created_at", nullable = false, updatable = false)
    private LocalDateTime createdAt = LocalDateTime.now();

}
```

```
package org.example.service.postgres;


import org.example.service.hibernate.EntityManagerUtil;
import org.example.service.postgres.entity.RuleData;

import javax.persistence.*;
import java.util.List;
import java.util.function.Consumer;

public class RuleDataRepository {

    public void save(RuleData ruleData) {
        executeInTransaction(em -> em.persist(ruleData));
    }

    public RuleData getById(Long id) {
        EntityManager em = EntityManagerUtil.getEntityManager();
        return em.find(RuleData.class, id);
    }

    public List<RuleData> getAll() {
        EntityManager em = EntityManagerUtil.getEntityManager();
        return em.createQuery("SELECT r FROM RuleData r",
RuleData.class).getResultList();
    }

    public void update(RuleData ruleData) {
        executeInTransaction(em -> em.merge(ruleData));
    }

    public void delete(Long id) {
        executeInTransaction(em -> {
            RuleData ruleData = em.find(RuleData.class, id);
            if (ruleData != null) {
                em.remove(ruleData);
            }
        });
    }

    private void executeInTransaction(Consumer<EntityManager> action) {
        EntityManager em = EntityManagerUtil.getEntityManager();
        EntityTransaction transaction = em.getTransaction();
        try {
            transaction.begin();
```

```
            action.accept(em);
            transaction.commit();
        } catch (Exception e) {
            if (transaction.isActive()) {
                transaction.rollback();
            }
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
            throw new RuntimeException("Transaction failed, rolled back", e);
        } finally {
            EntityManagerUtil.closeEntityManager();
        }
    }
}
```

```
package org.example.service.postgres;

import org.example.factory.RuleDataRepositoryFactory;
import org.example.service.postgres.entity.RuleData;

import java.util.List;

public class RuleDataService {
    RuleDataRepository ruleDataRepository;

    public RuleDataService() {
        this.ruleDataRepository = RuleDataRepositoryFactory.getInstance();
    }

    public void saveRuleData(String jsonData) {
        RuleData ruleData = new RuleData();
        ruleData.setData(jsonData);
        ruleDataRepository.save(ruleData);
    }

    public RuleData getRuleDataById(Long id) {
        return ruleDataRepository.getById(id);
    }

    public List<RuleData> getAllRuleData() {
        return ruleDataRepository.getAll();
    }
}
```

```
package org.example.service.postgres;

import org.example.entity.Topic;
import org.example.service.hibernate.EntityManagerUtil;

import javax.persistence.*;
import java.time.Duration;
import java.time.ZonedDateTime;
import java.util.List;
import java.util.function.Consumer;

import javax.persistence.EntityManager;
import javax.persistence.EntityTransaction;


public class TopicRepository {

    public void save(Topic topic) {
```

```
        executeInTransaction(em -> em.persist(topic));
    }

    public Topic getById(Integer id) {
        EntityManager em = EntityManagerUtil.getEntityManager();
        return em.find(Topic.class, id);
    }

    public List<Topic> getAll() {
        EntityManager em = EntityManagerUtil.getEntityManager();
        return getAllTopicQuery(em)
                .getResultList();
    }

    private TypedQuery<Topic> getAllTopicQuery(EntityManager em) {
        return em.createQuery("SELECT t FROM Topic t", Topic.class);
    }

    public void update(Topic topic) {
        executeInTransaction(em -> em.merge(topic));
    }

    public void delete(Integer id) {
        executeInTransaction(em -> {
            Topic topic = em.find(Topic.class, id);
            if (topic != null) {
                em.remove(topic);
            }
        });
    }

    public void deleteByTitle(String title) {
        executeInTransaction(em -> {
            TypedQuery<Topic> query = getTopicByTitleQuery(em);
            query.setParameter("title", title);
            List<Topic> topics = query.getResultList();

            for (Topic topic : topics) {
                em.remove(topic);
            }
        });
    }

    private TypedQuery<Topic> getTopicByTitleQuery(EntityManager em) {
        return em.createQuery(
                "SELECT t FROM Topic t WHERE t.title = :title", Topic.class
        );
    }

    public void processTopic(Topic incomingTopic) {
        executeInTransaction(em -> {
            TypedQuery<Topic> query = getAllTopicQuery(em);
            List<Topic> topics = query.getResultList();

            ZonedDateTime now = ZonedDateTime.now();
            proceedUpdateTopicList(incomingTopic, em, topics, now);

            if (isTopicNotExist(incomingTopic, topics)) {
                incomingTopic.setData(now.toInstant());
                em.persist(incomingTopic);
            }
        });
    }
```

```
    private void proceedUpdateTopicList(Topic incomingTopic, EntityManager em,
List<Topic> topics, ZonedDateTime now) {
        for (Topic topic : topics) {
            if (isTooOld(topic, now)) {
                em.remove(topic);
            } else if (isTopicsEquels(incomingTopic, topic)) {
                topic.setData(now.toInstant());
                em.merge(topic);
            }
        }
    }

    private boolean isTopicNotExist(Topic incomingTopic, List<Topic> topics) {
        return topics.stream().noneMatch(t -> isTopicsEquels(incomingTopic, t));
    }

    private boolean isTopicsEquels(Topic incomingTopic, Topic topic) {
        return topic.getTitle().equals(incomingTopic.getTitle());
    }

    private boolean isTooOld(Topic topic, ZonedDateTime now) {
        return Duration.between(topic.getData(), now).toSeconds() > 30;
    }

    private void executeInTransaction(Consumer<EntityManager> action) {
        EntityManager em = EntityManagerUtil.getEntityManager();
        EntityTransaction transaction = em.getTransaction();
        try {
            transaction.begin();
            action.accept(em);
            transaction.commit();
        } catch (Exception e) {
            if (transaction.isActive()) {
                transaction.rollback();
            }
            throw new RuntimeException("Transaction failed, rolled back", e);
        } finally {
            EntityManagerUtil.closeEntityManager();
        }
    }
}
```

```
package org.example.service;

import org.example.entity.Device;
import org.example.entity.DeviceType;
import org.example.utlity.ClassMethodService;
import org.jetbrains.annotations.NotNull;

import java.lang.reflect.Field;
import java.lang.reflect.Method;
import java.lang.reflect.InvocationTargetException;
import java.util.Optional;

public class DeviceUpdater {

    private final InflexDBService inflexDBService= new InflexDBService();
    private final ClassMethodService classMethodService = new
ClassMethodService();;


    public void updateDevice(Device targetDevice, DeviceType deviceType, String
```

```
sensorId) {
        System.out.printf("%s %s\n",deviceType, sensorId);
        Optional<Device> newDeviceOptional =
inflexDBService.getDeviceById(deviceType, sensorId);

        if (newDeviceOptional.isPresent()) {
            Device newDevice = newDeviceOptional.get();
            Field[] fields = getFields(newDevice);

            for (Field field : fields) {
                Method getter = getGetterNewDevice(field, newDevice);
                Method setter = getSetterOldDevice(targetDevice, field);

                if (isMethodsNotNull(getter, setter)) {
                    System.out.println("Getter or setter not found for field: " +
field.getName());
                    continue;
                }

                Object value = getNewValue(getter, newDevice);
                updateField(targetDevice, field, setter, value);
            }
        } else {
            System.out.println("Can't update device - new device is null");
        }
    }

    private Field [] getFields(Device newDevice) {
        Class<?> clazz = newDevice.getClass();
        return clazz.getDeclaredFields();
    }

    private Method getGetterNewDevice(Field field, Device newDevice) {
        return classMethodService.getGetterMethodByFieldName(field.getName(),
newDevice.getClass());
    }

    private Method getSetterOldDevice(Device targetDevice, Field field) {
        return classMethodService.getSetterMethodByField(field.getName(),
targetDevice.getClass());
    }

    private Object getNewValue(Method getter, Device newDevice) {
        try {
            return getter.invoke(newDevice);
        } catch (IllegalAccessException | InvocationTargetException e) {
            throw new RuntimeException(e);
        }
    }

    private boolean isMethodsNotNull(Method getter, Method setter) {
        return getter == null || setter == null;
    }

    private void updateField(Device targetDevice, Field field, Method setter,
Object value) {
        try {
            setter.invoke(targetDevice, value);
        } catch (IllegalAccessException | InvocationTargetException e) {
            throw new RuntimeException("Error updating field: " + field.getName(),
e);
        }
    }
}
```

```java
package org.example.service;

import org.example.entity.Device;
import org.example.entity.DeviceType;
import org.example.service.inflexdb.InflexDBRepository;

import java.util.List;
import java.util.Optional;

public class InflexDBService{

    private final InflexDBRepository inflexDBRepository;

    public InflexDBService() {
        this.inflexDBRepository = new InflexDBRepository();
    }

    public void writeData(Device device) {
       inflexDBRepository.writeData(device);
    }

    public Optional<Device> getDeviceById(DeviceType deviceType, String deviceId){
        return inflexDBRepository.getDeviceById(deviceType, deviceId);
    }

    public List<Device> getAllDevices() {
       return inflexDBRepository.getAllDevices();
    }

}
```

```java
package org.example.service;

import org.example.rule.entity.Rule;
import org.example.rule.serializer.RuleDeserializer;
import org.example.rule.serializer.RuleSerializer;
import org.example.service.postgres.entity.RuleData;
import org.example.service.postgres.RuleDataService;

import java.util.List;

public class RuleService {
    RuleDataService ruleDataService;

    public RuleService() {
        this.ruleDataService = new RuleDataService();
    }

    public void saveRule(Rule rule) {
        RuleSerializer ruleSerializer = new RuleSerializer();
        String jsonData = ruleSerializer.serializeRule(rule);
        ruleDataService.saveRuleData(jsonData);
    }

    public Rule getRuleDataById(Long id) {
        RuleData ruleData = ruleDataService.getRuleDataById(id);
        return getRuleByData(ruleData);
    }

    public Rule getRuleByData(RuleData ruleData) {
        RuleDeserializer deserializer = new RuleDeserializer();
```

```
        Rule rule = deserializer.deserialize(ruleData.getData());
        rule.setId(ruleData.getId());
        rule.setDescription(ruleData.getDescription());
        rule.setTimeStamp(ruleData.getCreatedAt());
        return rule;
    }

    public List<Rule> getAllRule() {
        List<RuleData> ruleData = ruleDataService.getAllRuleData();
        return ruleData.stream()
                .map(this::getRuleByData)
                .toList();
    }
}
```

```
package org.example.service;

import org.example.entity.Topic;
import org.example.factory.TopicRepositoryFactory;
import org.example.service.postgres.TopicRepository;

import java.util.List;

public class TopicService {

    private final TopicRepository topicRepository =
TopicRepositoryFactory.getInstance();

    public void addTopic(Topic topic) {
        topicRepository.save(topic);
    }

    public Topic getTopic(Integer id) {
        return topicRepository.getById(id);
    }

    public List<Topic> getAllTopics() {
        return topicRepository.getAll();
    }

    public void updateTopic(Topic topic) {
        topicRepository.update(topic);
    }

    public void deleteTopic(Integer id) {
        topicRepository.delete(id);
    }

    public void deleteByTitle(String title){
        topicRepository.deleteByTitle(title);
    }

    public void subscribeTopic(String topic) {
        Topic topicEntity = new Topic();
        topicEntity.setTitle(topic);
        topicRepository.processTopic(topicEntity);
    }
}
```

```
        package org.example.subscriber.parser;

import org.example.entity.CurrentLineSensor;
import org.example.entity.Device;
```

```
import org.example.subscriber.DeviceDataParser;

import java.util.Map;

public class CurrentLineDataParser extends DeviceDataParser {
    @Override
    protected void setDeviceSpecificData(Device device, Map<String, String>
messageData) {
        double current = Double.parseDouble(messageData.get("current"));
        double voltage = Double.parseDouble(messageData.get("voltage"));
        CurrentLineSensor currentLineSensor = (CurrentLineSensor) device;
        currentLineSensor.setCurrent(current);
        currentLineSensor.setVoltage(voltage);
    }


}
```

```
package org.example.subscriber.parser;

import org.example.entity.Device;
import org.example.entity.Generator;
import org.example.subscriber.DeviceDataParser;

import java.util.Map;

public class GeneratorDataParser extends DeviceDataParser {

    @Override
    protected void setDeviceSpecificData(Device device, Map<String, String>
messageData) {
        double voltage = Double.parseDouble(messageData.get("voltage"));
        double current = Double.parseDouble(messageData.get("current"));
        boolean isWorking = Boolean.parseBoolean(messageData.get("isWorking"));
        Generator generator = (Generator) device;
        generator.setVoltage(voltage);
        generator.setCurrent(current);
        generator.setIsWorking(isWorking);
    }
}
```

```
package org.example.subscriber.parser;


import org.example.entity.Device;
import org.example.entity.SwitchBoard;
import org.example.subscriber.DeviceDataParser;

import java.util.Map;

public class SwitchBoardParser extends DeviceDataParser {


    @Override
    protected void setDeviceSpecificData(Device device, Map<String, String>
messageData) {
        double current = Double.parseDouble(messageData.get("current"));
        SwitchBoard switchBoard = (SwitchBoard) device;
        switchBoard.setCurrent(current);
    }
}
```

```
package org.example.subscriber.parser;

import org.example.entity.Device;
import org.example.entity.Transformer;
import org.example.subscriber.DeviceDataParser;

import java.util.Map;

public class TransformerDataParser extends DeviceDataParser {

    @Override
    protected void setDeviceSpecificData(Device device, Map<String, String>
messageData) {
        Transformer transporter = (Transformer) device;

transporter.setTurnsRation(Double.valueOf(messageData.get("turnsRation")));
    }
}
```

```
package org.example.subscriber.service.mqtt.publisher;

import lombok.ToString;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttConnectOptions;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.example.entity.Device;

import java.util.Objects;


public class MQTTPublisher {
    @Override
    public String toString() {
        return String.format("MQTTPublisher{clientId='%s'}",getClientId());
    }

    MqttClient client;

    public MQTTPublisher(MqttClient client) {
        this.client = client;
    }

    public String getClientId() {
        return client.getClientId();
    }

    public <T extends Device> void writeData(T device){
        try {
            MqttConnectOptions options = new MqttConnectOptions();
            options.setCleanSession(true);
            client.connect(options);
            String payload = device.getMqttPayload();
            MqttMessage message = new MqttMessage(payload.getBytes());
            message.setQos(1);
            message.setRetained(true);
            System.out.println(message.isRetained());
            String topic = device.getMqttTopic();
            client.publish(topic, message);
//            System.out.println("Message published to topic " + topic);
            client.disconnect();
//            System.out.println("Disconnected from broker");
```

```
        } catch (MqttException e) {
            System.err.println("Error: " + e.getMessage());
            e.printStackTrace();
        }
    }


    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        MQTTPublisher that = (MQTTPublisher) o;
        return Objects.equals(client.getClientId(), that.client.getClientId());
    }

    @Override
    public int hashCode() {
        return Objects.hashCode(client);
    }
}
```

```
package org.example.subscriber.service.mqtt.publisher;

import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.example.subscriber.service.mqtt.MQTTClientFactory;

public class MQTTPublisherFactory {
    public static MQTTPublisher getPublisher(String clientId){
        MqttClient mqttClient = getMqttClient(clientId);
        return new MQTTPublisher(mqttClient);
    }

    private static MqttClient getMqttClient(String clientId){
        MqttClient mqttClient = null;
        try {
            mqttClient= MQTTClientFactory.getMqttClient(clientId);
        } catch (MqttException e) {
            throw new RuntimeException(e);
        }
        return mqttClient;
    }
}
```

```
package org.example.subscriber.service.mqtt;

import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.persist.MemoryPersistence;

public class MQTTClientFactory {
    private static final String BROKER = "tcp://localhost:1883";


    public static MqttClient getMqttClient(String clientId) throws MqttException {
        return new MqttClient(BROKER, clientId, new MemoryPersistence());
    }
}
```

```
package org.example.subscriber.service.mqtt;


import java.io.IOException;
import java.net.URI;

import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

public class MQTTRepository {
    private final String BASE_URL = "http://localhost:18083/api/v5/";
    private final String USERNAME = "687d602a30c44db7";
    private final String PASSWORD =
"9AUnIZ9AFn7wd9C9AxSb6rtFfYJ089BemSIF9CtFhBnm1RXDA";

    public HttpRequest getAuthorization(String endpoint) {
        return HttpRequest.newBuilder()
                .uri(URI.create(BASE_URL + endpoint))
                .header("Authorization", "Basic " +

java.util.Base64.getEncoder().encodeToString(String.format("%s:%s", USERNAME,
PASSWORD).getBytes()))
                .header("Content-Type", "application/json")
                .GET()
                .build();
    }

    public String getResponseBodyByEndPoint(String endPoint) throws IOException,
InterruptedException {
        HttpClient client = HttpClient.newHttpClient();
        HttpRequest request = getAuthorization(endPoint);
        HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());
        return response.body();
    }


}
```

```
package org.example.subscriber.service.mqtt;

import java.util.Set;

public class MQTTService {
    private MQTTRepository mqttRepository;

    public MQTTService() {
        this.mqttRepository = new MQTTRepository();
    }

    public Set<String> getTopics() {
        try {
            String endpoint = "topics";
            String responseBody =
mqttRepository.getResponseBodyByEndPoint(endpoint);
            ResponseTopicUnwraper responseTopicUnwraper = new
ResponseTopicUnwraper();
            return responseTopicUnwraper.unwrapResponse(responseBody);
        } catch (Exception e) {
            throw new IllegalArgumentException(e);
        }
```

```
    }
}
```

```
package org.example.subscriber.service.mqtt;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;

import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;

public class ResponseTopicUnwraper {
    public Set<String> unwrapResponse (String responseBody){
        Map<String, Object> responseBodyUnwrapped = getMessageData(responseBody);
        List<Object> nodes = (List<Object>) responseBodyUnwrapped.get("data");
        return nodes.stream().map(x -> ((Map<String, String>)
x).get("topic")).filter(x -> !x.equals("client/#")).collect(Collectors.toSet());
    }

    private Map<String, Object> getMessageData(String message) {
        ObjectMapper objectMapper = new ObjectMapper();
        Map<String, Object> messageData;
        try {
            messageData = objectMapper.readValue(message, new
TypeReference<Map<String, Object>>() {
            });
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }
        return messageData;
    }
}
```

```
package org.example.subscriber.strategy;

import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.eclipse.paho.client.mqttv3.MqttMessage;
import org.example.entity.Device;
import org.example.entity.DeviceType;
import org.example.entity.Topic;
import org.example.service.InflexDBService;
import org.example.subscriber.DeviceDataParser;
import org.example.subscriber.DeviceDataParserFactory;
import org.example.subscriber.SubscriberBehaviour;

import java.util.Optional;
import java.util.Set;

public class DeviceDataListener extends SubscriberBehaviour {

    private final DeviceDataParserFactory parserFactory;
    private final Set<Topic> initTopics;


    public DeviceDataListener(DeviceDataParserFactory parserFactory, Set<Topic>
initTopics) {
        this.parserFactory = parserFactory;
        this.initTopics = initTopics;
```

```
    }

    @Override
    public void execute(MqttClient client, String topic, MqttMessage message) {
        String[] topicParts = topic.split("/");

        if (isTopicNotEmpty(topicParts)) {
            proceedData(topic, message, topicParts);
//            updateSubscribedTopic(client);
        }
    }

    private void proceedData(String topic, MqttMessage message, String[]
topicParts) {
        DeviceType deviceType = DeviceType.fromType(topicParts[1]);
        DeviceDataParser parser = parserFactory.getParser(deviceType);
        Optional<Device> device = handleData(topic, message, parser, deviceType);
        if (device.isPresent()) {
            writeToDataBase(device.get());
        }
    }

    private void updateSubscribedTopic(MqttClient client) {
        try {
            UpdatedSubscribedTopic updatedSubscriberTopic = new
UpdatedSubscribedTopic();
            updatedSubscriberTopic.update(client, initTopics);
        } catch (MqttException e) {
            throw new RuntimeException(e);
        }
    }

    private void writeToDataBase(Device device) {
        InflexDBService inflexDBService = new InflexDBService();
        inflexDBService.writeData(device);
    }

    private Optional<Device> handleData(String topic, MqttMessage message,
DeviceDataParser parser, DeviceType deviceType) {
        Optional<Device> device = Optional.empty();
        if (isParserNotNull(parser)) {
//            System.out.println(message);
            String[] topicParts = topic.split("/");
            device = parser.parserData(topicParts, new
String(message.getPayload()));
        } else {
            System.err.println("DeviceDataParser '" + deviceType + "' not
found.");
        }
        return device;
    }

    private boolean isTopicNotEmpty(String[] topicParts) {
        return topicParts.length > 1;
    }

    private boolean isParserNotNull(DeviceDataParser parser) {
        return parser != null;
    }
}
```

```
package org.example.subscriber.strategy;
```

```
import lombok.Getter;
import org.eclipse.paho.client.mqttv3.*;
import org.example.service.TopicService;
import org.example.subscriber.SubscriberBehaviour;

@Getter
public class TopicSubscriber extends SubscriberBehaviour {

    @Override
    public void execute(MqttClient client, String topic, MqttMessage message) {
        TopicService topicService = new TopicService();
        topicService.subscribeTopic(topic);
    }
}
```

```
package org.example.subscriber.strategy;

import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.example.entity.Topic;
import org.example.service.TopicService;
import org.jetbrains.annotations.NotNull;

import java.time.Duration;
import java.time.ZonedDateTime;
import java.util.*;
import java.util.stream.Collectors;

public class UpdatedSubscribedTopic {
    TopicService topicService ;
    public UpdatedSubscribedTopic() {
        this.topicService = new TopicService();
    }

    public void update(MqttClient client, Set<Topic> initTopics) throws
MqttException {
        Set<Topic> newTopics = new HashSet<>(topicService.getAllTopics());
        Set<String> oldTopicsString = getTopicsString(initTopics);
        proceedTopics(client, initTopics, newTopics, oldTopicsString);
    }

    private @NotNull Set<String> getTopicsString(Set<Topic> initTopics) {
        return initTopics.stream()
                .map(Topic::getTitle)
                .collect(Collectors.toSet());
    }

    private void proceedTopics(MqttClient client, Set<Topic> initTopics,
Set<Topic> newTopics, Set<String> oldTopicsString) throws MqttException {
        for (Topic topic : newTopics) {
            String topicTitle = topic.getTitle();
            if (isTopicTooOld(topic)) {
                client.unsubscribe(topicTitle);
//                removeInitTopicItem(initTopics, topic, topicTitle);
            }

            if (!oldTopicsString.contains(topicTitle)) {
                client.subscribe(topicTitle);
//                System.out.println("Subscribed new topic: " + topicTitle);
                addInitTopicItem(initTopics, topic);
            }
        }
```

```
    }
    private void removeInitTopicItem(Set<Topic> initTopics, Topic topic, String
topicTitle) {
        if (!initTopics.isEmpty()){
            Optional<Topic> topicFromInit = initTopics.stream()
                    .filter(x -> x.getTitle().equals(topic.getTitle()))
                    .findFirst();

            topicFromInit.ifPresent(initTopics::remove);
            System.out.println("Unsubscribed topic: " + topicTitle);
            waitNewTopics(initTopics);
        }
    }

    private void waitNewTopics(Set<Topic> initTopics) {
        while (initTopics.isEmpty()){
            Set<Topic> newTopics = new HashSet<>(getAllNewTopic());
            if (!newTopics.isEmpty()){
                initTopics = newTopics;
            }
        }
    }

    private List<Topic> getAllNewTopic() {
        return topicService.getAllTopics().stream()
                .filter(this::isNewTopic).toList();
    }

    private boolean isTopicTooOld(Topic topic) {
        ZonedDateTime now = ZonedDateTime.now();
        return Duration.between(topic.getData(), now).toSeconds() > 30;
    }

    private boolean isNewTopic(Topic topic){
        return !isTopicTooOld(topic);
    }

    private void addInitTopicItem(Set<Topic> initTopics, Topic topic) {
        initTopics.add(topic);
    }

}
```

```
package org.example.subscriber;

import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.example.entity.Device;
import org.example.entity.DeviceType;
import org.example.factory.DeviceFactory;

import java.util.Arrays;
import java.util.Map;
import java.util.NoSuchElementException;
import java.util.Optional;

public abstract class DeviceDataParser {
    public Optional<Device>  parserData(String[] topicParts, String message){
        DeviceType measurement = DeviceType.fromType(topicParts[1]);
        System.out.println(Arrays.toString(topicParts));
        if (measurement != null){
            Optional<Device> device = DeviceFactory.getDevice(measurement);
```

```
            device.ifPresent(value -> setDeviceData(value, topicParts, message));
            return device;
        }else {
            throw new NoSuchElementException();
        }
    }

    protected void setDeviceData(Device device, String[] topicParts, String
message) {
        device.setSensorId(topicParts[2]);
        Map<String, String> messageData = getMessageData(message);
        setDeviceSpecificData(device, messageData);
    }

    protected abstract void setDeviceSpecificData(Device device, Map<String,
String> messageData);

    protected Map<String, String> getMessageData(String message) {
        ObjectMapper objectMapper = new ObjectMapper();
        Map<String, String> messageData;
        try {
            messageData =  objectMapper.readValue(message, new TypeReference<>() {
            });
        } catch (JsonProcessingException e) {
            throw new RuntimeException(e);
        }
        return messageData;
    }
}
```

```
package org.example.subscriber;

import org.example.entity.DeviceType;

import java.util.HashMap;
import java.util.Map;

public class DeviceDataParserFactory {
    private final Map<DeviceType, DeviceDataParser> parsers = new HashMap<>();

    public void registerParser(DeviceType deviceType, DeviceDataParser parser) {
        parsers.put(deviceType, parser);
    }

    public DeviceDataParser getParser(DeviceType deviceType) {
        return parsers.get(deviceType);
    }
}
```

```
package org.example.subscriber;

import lombok.Getter;
import org.eclipse.paho.client.mqttv3.*;
import org.example.entity.Topic;

import java.util.*;


@Getter
public class MQTTClientSubscriber {
    private final MqttClient client;
    private final SubscriberBehaviour subscriberBehaviour;
```

```
    public MQTTClientSubscriber(MqttClient client, SubscriberBehaviour
subscriberBehaviour) throws MqttException {
        this.client = client;
        this.subscriberBehaviour = subscriberBehaviour;
        setClientCallback(client);
    }

    private void setClientCallback(MqttClient client) {
        client.setCallback(new MqttCallback() {
            @Override
            public void connectionLost(Throwable cause) {
                cause.printStackTrace();
                System.err.println("Connection error: " + cause.getCause());
            }

            @Override
            public void messageArrived(String topic, MqttMessage message){
                System.out.println(message);
                subscriberBehaviour
                        .execute(client, topic, message);
            }

            @Override
            public void deliveryComplete(IMqttDeliveryToken token) {
            }
        });
    }

    public void connectAndSubscribe(Set<Topic> topics) throws MqttException {
        MqttConnectOptions options = new MqttConnectOptions();
        options.setKeepAliveInterval(60);
        options.setAutomaticReconnect(true);
        client.connect(options);
        for (Topic topic: topics){
            client.subscribe(topic.getTitle());
//            System.out.println("Subscribe topic: " + topic);
        }
    }


}
```

```
package org.example.subscriber;

import org.example.entity.Device;

import java.lang.reflect.Field;
import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;

public class MQTTPayloadBuilder {
    public String getPayloadBuilder(Device device){
        StringBuilder jsonBuilder = new StringBuilder();
        jsonBuilder.append("{");

        Field[] fields = device.getClass().getDeclaredFields();

        for (int i = 0; i < fields.length; i++) {
            Field field = fields[i];
            field.setAccessible(true);

            String fieldName = field.getName();
```

```
            String methodName = getMethodName(fieldName);
            Method method;
            Object value;
            try {
                method = device.getClass().getMethod(methodName);
                value = method.invoke(device);
            } catch (NoSuchMethodException | InvocationTargetException |
IllegalAccessException e) {
                throw new RuntimeException(e);
            }

            jsonBuilder.append(String.format("\"%s\":\"%s\"", field.getName(),
value));

            if (i < fields.length - 1) {
                jsonBuilder.append(",");
            }
        }
        jsonBuilder.append("}");
        return jsonBuilder.toString();
    }

    private String getMethodName(String fieldName) {
        String s1 = fieldName.substring(0, 1).toUpperCase();
        String nameCapitalized = s1 + fieldName.substring(1);
        return "get" + nameCapitalized;
    }
}
```

```
package org.example.subscriber;

import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttMessage;

public abstract class SubscriberBehaviour {
    public abstract void execute(MqttClient client, String topic, MqttMessage
message);

}
```

```
package org.example.subscriber;

import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.example.entity.Topic;
import org.example.subscriber.service.mqtt.MQTTClientFactory;

import java.util.Set;

public class SubscriberThread implements Runnable {

    private String clientId;
    private SubscriberBehaviour subscriberBehaviour;
    private Set<Topic> topics;

    public SubscriberThread(String clientId, SubscriberBehaviour
subscriberBehaviour, Set<Topic> topics) {
        this.clientId = clientId;
        this.subscriberBehaviour = subscriberBehaviour;
        this.topics = topics;
    }
```

```
    @Override
    public void run() {
        try {
            MqttClient client = MQTTClientFactory.getMqttClient(clientId);
            MQTTClientSubscriber clientSubscriber = new
MQTTClientSubscriber(client, subscriberBehaviour);
            clientSubscriber.connectAndSubscribe(topics);
        } catch (MqttException e) {
            e.printStackTrace();
        }
    }
}
```

```
package org.example.utlity;

import org.example.entity.Device;
import org.jetbrains.annotations.NotNull;

import java.lang.reflect.Method;

public class ClassMethodService {
    public @NotNull String getSetterTitleByField(String field) {
        return "set" + capitalize(field);
    }

    public String capitalize(String fieldName) {
        return fieldName.substring(0, 1).toUpperCase() + fieldName.substring(1);
    }

    public Method getMethod(Class<? extends Device> deviceClass, String
titleMethod) {
        Method method = null;
        Method[] methods = deviceClass.getDeclaredMethods();
        for (Method methodItem : methods) {
            if (methodItem.getName().equals(titleMethod)) {
                method = methodItem;
            }
        }
        return method;
    }

    public String getGetterTitleByField(String fieldName) {
        String s1 = fieldName.substring(0, 1).toUpperCase();
        String nameCapitalized = s1 + fieldName.substring(1);
        return "get" + nameCapitalized;
    }

    public Method getGetterMethodByFieldName(String fieldName, Class<? extends
Device> deviceClass){
        String getterTitle = getGetterTitleByField(fieldName);
        return getMethod(deviceClass, getterTitle);
    }

    public Method getSetterMethodByField(String fieldName, Class<? extends Device>
deviceClass){
        String setterTitle = getSetterTitleByField(fieldName);
        return getMethod(deviceClass, setterTitle);
    }
}
```

```java
package org.example;

import javafx.application.Application;
import javafx.application.Platform;
import javafx.stage.Stage;
import lombok.Getter;
import org.example.client.service.ObjectService;
import org.example.client.UIManager;

import java.util.List;

public class App extends Application {
    @Getter
    private static List<Object> objects;

    @Override
    public void start(Stage primaryStage) {
        ObjectService objectService = ObjectService.getInstance();
        objects = objectService.getObjects();
        UIManager uiManager = new UIManager(primaryStage);
        uiManager.init();
    }

    @Override
    public void stop() throws Exception {
        super.stop();
        System.out.println("Closing all windows");
        Platform.exit();
        System.exit(0);
    }

    public static void main(String[] args) {
        launch(args);
    }

}
```

```java
package org.example;

import org.example.entity.DeviceType;
import org.example.entity.Topic;
import org.example.rule.entity.Rule;
import org.example.rule.executor.RuleExecutorThread;
import org.example.service.RuleService;
import org.example.service.TopicService;
import org.example.subscriber.DeviceDataParserFactory;
import org.example.subscriber.SubscriberBehaviour;
import org.example.subscriber.SubscriberThread;
import org.example.subscriber.parser.CurrentLineDataParser;
import org.example.subscriber.parser.SwitchBoardParser;
import org.example.subscriber.parser.TransformerDataParser;
import org.example.subscriber.parser.GeneratorDataParser;
import org.example.subscriber.strategy.DeviceDataListener;
import org.example.subscriber.strategy.TopicSubscriber;

import java.time.ZonedDateTime;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

public class SubscribersThread
{
    public static void main(String[] args) {
```

```
        SubscribersThread subscribersThread = new SubscribersThread();
        subscribersThread.topicListener();
        subscribersThread.deviceDataListener();

        try {
            Thread.sleep(15000);
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }

//        subscribersThread.ruleExecutor();
        try {
            Thread.currentThread().join();
        } catch (InterruptedException e) {
            throw new RuntimeException(e);
        }
    }

    private void deviceDataListener() {
        DeviceDataParserFactory parserFactory = new DeviceDataParserFactory();
        parserFactory.registerParser(DeviceType.CURRENT_LINE_SENSOR, new
CurrentLineDataParser());
        parserFactory.registerParser(DeviceType.GENERATOR, new
GeneratorDataParser());
        TopicService topicService = new TopicService();
        Set<Topic> topics = new HashSet<>(topicService.getAllTopics());
        while (topics.isEmpty()){
            try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
            topics = new HashSet<>(topicService.getAllTopics());
        }
        DeviceDataListener subscriberBehaviour = new
DeviceDataListener(parserFactory, topics);
        String idClient = "DeviceSubscriberClient";

        Thread thread = new Thread(new SubscriberThread(idClient,
subscriberBehaviour, topics));
        thread.start();
    }

    private void topicListener() {
        Set<Topic> topics = new HashSet<>(Set.of(new Topic("client/#",
ZonedDateTime.now().toInstant())));
        SubscriberBehaviour subscriberBehaviour = new TopicSubscriber();
        String idClient = "DeviceEnvironmentListener";
        Thread thread = new Thread(new SubscriberThread(idClient,
subscriberBehaviour, topics));
        thread.start();
    }

    private  void ruleExecutor(){
        RuleService ruleService = new RuleService();
        List<Rule> ruleList= ruleService.getAllRule();
//        System.out.println(ruleList);
        for (Rule rule : ruleList){
            Thread thread = new RuleExecutorThread(rule);
            thread.start();
        }
    }
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" version="2.1">
    <persistence-unit name="my-persistence-unit" transaction-
type="RESOURCE_LOCAL">
        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

        <!-- JDBC settings -->
        <properties>
            <property name="hibernate.dialect"
value="org.hibernate.dialect.PostgreSQLDialect"/>
            <property name="hibernate.connection.driver_class"
value="org.postgresql.Driver"/>
            <property name="hibernate.connection.url"
value="jdbc:postgresql://localhost:5432/eecsystem"/>
            <property name="hibernate.connection.username" value="admin"/>
            <property name="hibernate.connection.password" value="admin123"/>

            <!-- Hibernate behavior -->
            <property name="hibernate.hbm2ddl.auto" value="update"/>
            <property name="hibernate.show_sql" value="false"/>
            <property name="hibernate.format_sql" value="false"/>
            <property name="hibernate.use_sql_comments" value="false"/>
            <property name="hibernate.generate_statistics" value="false"/>
            <property name="eclipselink.logging.level" value="OFF" />
            <!-- Connection pool (optional) -->
            <property name="hibernate.hikari.minimumIdle" value="5"/>
            <property name="hibernate.hikari.maximumPoolSize" value="20"/>
            <property name="hibernate.hikari.connectionTimeout" value="30000"/>
            <property name="hibernate.hikari.idleTimeout" value="600000"/>
            <property name="hibernate.hikari.maxLifetime" value="1800000"/>
            <property name="hibernate.hikari.poolName" value="HikariCP"/>
            <property name="hibernate.cache.use_second_level_cache" value="false"
/>
            <property name="hibernate.cache.use_query_cache" value="false" />

        </properties>
    </persistence-unit>
</persistence>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ComboBox?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>

<HBox spacing="5" xmlns="http://javafx.com/javafx/23.0.1"
xmlns:fx="http://javafx.com/fxml/1">
    <padding>
        <Insets bottom="5" left="5" right="5" top="5" />
    </padding>
    <children>
        <Button onAction="#handleDeleteButtonAction" text="Delete" />
        <ComboBox fx:id="actionObjectSelector" promptText="Choice object">
            <!-- Items will be populated from the controller -->
        </ComboBox>
        <ComboBox fx:id="actionFieldSelector" promptText="Choice method">
            <!-- Items will be populated dynamically -->
        </ComboBox>
        <VBox fx:id="expressionsContainer" HBox.hgrow="ALWAYS">
            <!-- Additional children can be added here -->
        </VBox>
```

```
    </children>
</HBox>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.layout.VBox?>

<VBox fx:id="vBox" VBox.vgrow="ALWAYS" xmlns="http://javafx.com/javafx/23.0.1"
xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Button mnemonicParsing="false" onAction="#onAddConditional" text="Add
condition" />
        <Separator prefWidth="200.0" VBox.vgrow="ALWAYS" />
        <ScrollPane fitToWidth="true" prefHeight="200" VBox.vgrow="ALWAYS">
            <VBox fx:id="conditionalBox" VBox.vgrow="ALWAYS" />
        </ScrollPane>
    </children>
</VBox>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ComboBox?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.text.Text?>

<?import javafx.scene.control.Label?>
<Pane focusTraversable="true" prefHeight="150.0" prefWidth="150.0" style="-fx-
background-color: #f4f4f4; -fx-border-radius: 10; -fx-border-color: #ccc; -fx-
border-width: 2;" xmlns="http://javafx.com/javafx/23.0.1"
xmlns:fx="http://javafx.com/fxml/1">
   <children>
      <Label fx:id="labelClass"></Label>
      <Label fx:id="labelTitle"></Label>
      <Button layoutX="115.0" layoutY="5.0" maxHeight="-Infinity" maxWidth="-
Infinity" minHeight="30.0" minWidth="30.0" mnemonicParsing="false"
onAction="#OnDelete" prefHeight="30.0" prefWidth="30.0" text="x" style="-fx-
background-color: #e74c3c; -fx-text-fill: white; -fx-font-size: 14px; -fx-border-
radius: 15px; -fx-background-radius: 15px;" />
      <Text fx:id="deviceId" layoutX="10.0" layoutY="50.0" text="Device" style="-
fx-font-size: 14px; -fx-font-weight: bold; -fx-fill: #2c3e50;" />
      <ComboBox fx:id="methodBox" layoutX="10.0" layoutY="70.0" prefHeight="30.0"
prefWidth="130.0" visibleRowCount="5" style="-fx-background-color: #ffffff; -fx-
border-color: #ccc; -fx-border-radius: 5px; -fx-font-size: 12px;" />
   </children>
</Pane>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.HBox?>

<HBox fx:id="expressionsBox" alignment="CENTER_LEFT"
maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308" minHeight="-
Infinity" minWidth="-Infinity" prefHeight="90.0"
xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1">
    <children>
```

```
    </children>
</HBox>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.ComboBox?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.VBox?>

<VBox fx:id="conditionWithOperator" prefHeight="141.0" prefWidth="378.0"
xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <HBox fx:id="controlElement" prefHeight="35.0" prefWidth="378.0"
spacing="10">
            <ComboBox fx:id="deviceChoiceBox" prefHeight="29.0" prefWidth="120.0"
style="-fx-font-size: 14px;" />
            <Button mnemonicParsing="false" onAction="#onAddObject"
text="AddObject"
                    style="-fx-background-color: #4CAF50; -fx-text-fill: white; -
fx-font-size: 14px;" />
            <Button mnemonicParsing="false" onAction="#onAddValue" text="AddValue"
                    style="-fx-background-color: #4CAF50; -fx-text-fill: white; -
fx-font-size: 14px;" />
        </HBox>
        <ScrollPane fitToWidth="true" fitToHeight="true" prefWidth="200.0">
            <HBox fx:id="deviceContainer" alignment="TOP_CENTER"
prefHeight="100.0" prefWidth="378.0"/>
        </ScrollPane>
    </children>
</VBox>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.control.Label?>
<VBox xmlns:fx="http://javafx.com/fxml/1"
fx:controller="org.example.client.controllers.FullDeviceController"
      spacing="5" alignment="TOP_LEFT" style="-fx-padding: 10; -fx-background-
insets: 5; -fx-background-radius: 5; -fx-border-radius: 5; -fx-border-width: 1; -
fx-border-color: #cccccc;">
    <children>
        <Label fx:id="deviceId" text="Device Details" style="-fx-font-size: 14px;
-fx-font-weight: bold; -fx-padding: 0 0 5 0;"/>
        <VBox fx:id="deviceDetailsBox" spacing="3" alignment="TOP_LEFT" style="-
fx-padding: 5;">
        </VBox>
    </children>
</VBox>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.text.Font?>
<?import javafx.scene.text.Text?>

<VBox xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1"
```

```
fx:controller="org.example.client.controllers.MainSceneController">
   <Pane minHeight="201.0" minWidth="798.0" prefHeight="201.0" prefWidth="798.0">
      <children>
         <Button fx:id="mainButton" layoutX="510.0" layoutY="102.0"
onAction="#buttonClicked" text="Apply" />
         <TextField fx:id="turnsRation" layoutX="333.0" layoutY="102.0" />
         <Text layoutX="205.0" layoutY="61.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="Transporter Data" wrappingWidth="225.13671875">
            <font>
               <Font size="21.0" />
            </font>
         </Text>
         <Text layoutX="205.0" layoutY="120.0" strokeType="OUTSIDE"
strokeWidth="0.0" text="turns ratio">
            <font>
               <Font size="15.0" />
            </font>
         </Text>
      </children>
   </Pane>
</VBox>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>
<?import javafx.scene.control.ScrollPane?>
<AnchorPane xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/23.0.1"
fx:controller="org.example.client.controllers.MapController">
    <children>
        <VBox fx:id="listBox" layoutX="20" layoutY="20" spacing="10">
        </VBox>
        <ScrollPane fx:id="scrollPane"  layoutX="300" layoutY="20"
fitToWidth="true" fitToHeight="true" hbarPolicy="ALWAYS" vbarPolicy="ALWAYS"
prefWidth="1600" prefHeight="800">
        <content>
            <AnchorPane fx:id="dropPane" prefWidth="2000" prefHeight="2000">
            </AnchorPane>
        </content>
    </ScrollPane>
    </children>
</AnchorPane>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Accordion?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TitledPane?>
<?import javafx.scene.layout.HBox?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.layout.VBox?>

<HBox maxHeight="1.7976931348623157E308" maxWidth="1.7976931348623157E308"
xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1">
   <children>
      <Pane fx:id="toolBarPane" HBox.hgrow="NEVER" />
```

```
      <VBox alignment="TOP_CENTER" maxHeight="1.7976931348623157E308"
maxWidth="1.7976931348623157E308" spacing="10" HBox.hgrow="ALWAYS">
         <children>
            <Pane VBox.vgrow="NEVER">
               <children>
                  <Label layoutX="353.0" style="-fx-font-size: 18px; -fx-font-
weight: bold; -fx-text-fill: #333333;" text="Rule builder" />
                  <Button layoutX="10.0" layoutY="13.0" onAction="#onSaveRule"
style="-fx-background-color: #4CAF50; -fx-text-fill: white; -fx-font-size: 14px;"
text="Save" />
               </children>
            </Pane>
            <Separator prefWidth="865.0" />
            <VBox fx:id="ruleContainer" VBox.vgrow="ALWAYS"/>
         </children>
         <padding>
            <Insets bottom="10" left="10" right="10" top="10" />
         </padding>
      </VBox>
   </children>
</HBox>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Accordion?>
<?import javafx.scene.control.Button?>

<?import javafx.scene.control.ScrollPane?>
<?import javafx.scene.control.Separator?>
<?import javafx.scene.control.TitledPane?>
<?import javafx.scene.layout.VBox?>

<?import javafx.scene.layout.BorderPane?>
<BorderPane xmlns="http://javafx.com/javafx/23.0.1"
xmlns:fx="http://javafx.com/fxml/1">
    <center>
        <Accordion maxHeight="Infinity" maxWidth="Infinity">
            <panes>
                <TitledPane animated="false" text="Conditionals">
                    <content>
                        <ScrollPane fitToHeight="true" fitToWidth="true">
                            <content>
                                <VBox fx:id="conditionsContainer" spacing="5" />
                            </content>
                        </ScrollPane>
                    </content>
                </TitledPane>
                <TitledPane animated="false" text="Actions">
                    <content>
                        <VBox spacing="5">
                            <children>
                                <Button onAction="#onAddAction"
                                        style="-fx-background-color: #2196F3; -fx-
text-fill: white; -fx-font-size: 14px;"
                                        text="Add action" />
                                <Separator />
                                <ScrollPane fitToWidth="true">
                                    <content>
                                        <VBox fx:id="actionsContainer" spacing="5"
/>
                                    </content>
                                </ScrollPane>
                            </children>
```

```
                        </VBox>
                    </content>
                </TitledPane>
            </panes>
        </Accordion>
    </center>
</BorderPane>
```

```
<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.layout.GridPane?>


<GridPane fx:id="ruleGrid" maxHeight="1.7976931348623157E308"
maxWidth="1.7976931348623157E308" minHeight="-Infinity"  minWidth="-Infinity"
xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1">
   <children>
   </children>
</GridPane>
```

```
<?xml version="1.0" encoding="UTF-8"?>


<?import javafx.scene.layout.Pane?>


<?import javafx.scene.control.TextField?>
<?import javafx.scene.control.Button?>
<Pane focusTraversable="true" prefHeight="100" prefWidth="120"
      style="-fx-background-color: #f4f4f4; -fx-border-radius: 8; -fx-border-
color: #ccc; -fx-border-width: 1;"
      xmlns="http://javafx.com/javafx/23.0.1" xmlns:fx="http://javafx.com/fxml/1">
    <children>
        <Button layoutX="85" layoutY="5" prefWidth="20" prefHeight="20" text="x"
                style="-fx-background-color: #e74c3c; -fx-text-fill: white;
                       -fx-font-size: 10px; -fx-border-radius: 10px; -fx-
background-radius: 10px;"
                onAction="#OnDelete" />
        <TextField fx:id="textField" layoutX="10" layoutY="35" prefWidth="100"
prefHeight="25" />
    </children>
</Pane>
```

```
package org.example.ClientUI;


import junit.framework.TestCase;
import org.example.entity.Device;
import org.example.entity.DeviceType;
import org.example.service.InflexDBService;
import org.example.service.inflexdb.InflexDBRepository;

import java.util.Optional;

public class ClientUITest extends TestCase {
    public void testName() {
        InflexDBService inflexDBService = new InflexDBService();
        InflexDBRepository inflexDBRepository = new InflexDBRepository();
```

```
        Optional<Device> device =
inflexDBRepository.getDeviceById(DeviceType.CURRENT_LINE_SENSOR, "consumer1");
        device.ifPresent(Device::updateDevice);
    }
}
```

```
package org.example.ClientUI;

import org.example.client.controllers.parser.ExpressionParser;
import org.example.entity.DeviceType;
import org.example.rule.entity.Action;
import org.example.rule.entity.ConditionWithOperator;
import org.example.rule.entity.Expression;
import org.example.rule.entity.Rule;
import org.example.service.RuleService;
import org.example.service.inflexdb.InflexDBRepository;
import org.junit.jupiter.api.Test;


import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class ExpressionParserTest {
    public class Person {
        private int age;

        public Person(int age) {
            this.age = age;
        }

        public int getAge() {
            return age;
        }

        public void setAge(int age) {
            this.age = age;
        }
    }
    @Test
    public void testComplexExpressionWithObjects() {
        Person person1 = new Person(10);
        Person person2 = new Person(5);
        Person person3 = new Person(3);
        Person person4 = new Person(2);


        List<Object[]> expressionList = Arrays.asList(
                new Object[]{person1, "getAge", "+"}, // person1.getAge() +
                new Object[]{person2, "getAge", "-"}, // person2.getAge() -
                new Object[]{person3, "getAge", "/"}, // person3.getAge() /
                new Object[]{person4, "getAge", "=="}, // person4.getAge()
                new Object[]{6.0, null, null} // person4.getAge()
        );

        ExpressionParser parser = new ExpressionParser();
        Expression parsedExpression = parser.parseExpressionList(expressionList);
        Object result = parsedExpression.evaluate();
        System.out.println("Result: " + result); // ((10 + 5) - 3) / 2 = 6.0
        assertEquals(true, result);
    }
    @Test
```

```
    public void testComplex(){
        RuleService ruleService = new RuleService();
        List<Rule> ruleList= ruleService.getAllRule();

        Rule rule = ruleList.get(0);

        Expression conditionExpression =
rule.getActions().getLast().getExpression();
        Expression one = conditionExpression.getLeftOperand();
        Expression two = one.getLeftOperand();



    }
}
```

```
package org.example.inflexdb;

import junit.framework.TestCase;
import org.example.entity.Device;
import org.example.entity.CurrentLineSensor;
import org.example.service.InflexDBService;

import java.util.List;

public class InflexDBServiceTest extends TestCase {
    public void testWriteData() {
        CurrentLineSensor device = new CurrentLineSensor();
        device.setVoltage(201);
        device.setCurrent(3);
        device.setSensorId("testDevice");
        InflexDBService inflexDBService = new InflexDBService();
        for (int i =0 ; i<10; i++){
            inflexDBService.writeData(device);

        }
    }

    public void testGetDevices(){
        InflexDBService inflexDBService = new InflexDBService();
        List<Device> deviceList = inflexDBService.getAllDevices();
    }
}
```

```
package org.example.mqtt;

import junit.framework.TestCase;
import org.example.entity.Device;
import org.example.service.inflexdb.InflexDBRepository;

import java.util.ArrayList;
import java.util.List;

public class DeviceGetMethodTest extends TestCase {
    public void testGetMQTTTopic() {
        InflexDBRepository inflexDBRepository = new InflexDBRepository();
        List<Device> devices = inflexDBRepository.getAllDevices();
        for (Device device : devices){
            System.out.println(device.getMqttTopic());
        }
    }
```

```
    public void testGetMQTTPayload() {
        InflexDBRepository inflexDBRepository = new InflexDBRepository();
        List<Device> devices = inflexDBRepository.getAllDevices();
        for (Device device : devices){
            System.out.println(device.getMqttPayload());
        }
    }

    public void testGetInflex() {
        InflexDBRepository inflexDBRepository = new InflexDBRepository();
        List<Device> devices = inflexDBRepository.getAllDevices();
        for (Device device : devices){
            System.out.println(device.toInfluxDBLineProtocol());
        }
    }
}
```

```
package org.example.mqtt;

import junit.framework.TestCase;
import org.example.entity.Device;
import org.example.entity.Generator;
import org.example.entity.SwitchBoard;
import org.example.entity.Transformer;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisher;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisherFactory;
import org.jetbrains.annotations.NotNull;

public class MQTTPublisherTest extends TestCase {
    public void testMQTTWrite() {
        MQTTPublisher mqttPublisher =
MQTTPublisherFactory.getPublisher("testClient");
        Device device = getGenerator();
        mqttPublisher.writeData(device);
    }

    private Device getTransformer() {
        Transformer transformer = new Transformer();
        transformer.setSensorId("device1");
        transformer.setTurnsRation(1.0);
        return transformer;
    }

    private @NotNull Generator getGenerator() {
        Generator generator = new Generator();
        generator.setSensorId("device2");
        generator.setVoltage(230);
        generator.setCurrent(7.0);
        generator.setIsWorking(false);
        return generator;
    }

    private Device getSwitchBoard(){
        SwitchBoard switchBoard = new SwitchBoard();
        switchBoard.setSensorId("device2");
        switchBoard.setCurrent(120);
        return switchBoard;
    }
}
```

```
package org.example.mqtt;
```

```
import junit.framework.TestCase;
import org.example.service.TopicService;

import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Scanner;

public class MqttTopicsExample  extends TestCase {
    public void testName() {
        try {
            URL url = new URL("http://localhost:1883/api/v4/topics");
            HttpURLConnection conn = (HttpURLConnection) url.openConnection();
            conn.setRequestMethod("GET");
//            conn.setRequestProperty("Authorization", "Bearer <YOUR_API_TOKEN>");

            int responseCode = conn.getResponseCode();
            if (responseCode == 200) {
                Scanner scanner = new Scanner(conn.getInputStream());
                while (scanner.hasNextLine()) {
                    System.out.println(scanner.nextLine());
                }
                scanner.close();
            } else {
                System.out.println("Error: " + responseCode);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void test() {
        TopicService topicService = new TopicService();
        System.out.println(topicService.getAllTopics());
    }
}
```

```
package org.example.rule;

import org.example.client.entity.CurrentSensorUI;
import org.example.client.ruleparser.ExpressionReverseParser;
import org.example.entity.Device;
import org.example.rule.entity.Expression;
import org.junit.jupiter.api.Test;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import static org.junit.jupiter.api.Assertions.assertArrayEquals;
import static org.junit.jupiter.api.Assertions.assertEquals;

class ExpressionReverseParserTest {

    @Test
    void testSingleNodeExpression() {
        Expression expression = new Expression("Device1", "method1");

        ExpressionReverseParser parser = new ExpressionReverseParser();
        List<Object[]> result = parser.reverseParseExpression(expression);
        List<Object[]> expected = new ArrayList<>();
        expected.add(new Object[]{"Device1", "method1", null});
        assertEquals(expected.size(), result.size());
        for (int i = 0; i < expected.size(); i++) {
```

```
            assertArrayEquals(expected.get(i), result.get(i));
        }
    }

    @Test
    void testSimpleBinaryExpression() {
        Expression expression = new Expression(
            new Expression("Device1", "method1"),
            "+",
            new Expression(42.0, null)
        );

        ExpressionReverseParser parser = new ExpressionReverseParser();
        List<Object[]> result = parser.reverseParseExpression(expression);

        List<Object[]> expected = new ArrayList<>();
        expected.add(new Object[]{"Device1", "method1", "+"});
        expected.add(new Object[]{42.0, null, null});

        assertEquals(expected.size(), result.size());
        for (int i = 0; i < expected.size(); i++) {

            assertArrayEquals(expected.get(i), result.get(i));
        }
    }

    @Test
    void testNestedExpression() {
        // Условие: Вложенное выражение
        Expression expression = new Expression(
            new Expression(
                new Expression("Device1", "method1"),
                "+",
                new Expression(42.0, null)
            ),
            "*",
            new Expression("Device2", "method2")
        );

        ExpressionReverseParser parser = new ExpressionReverseParser();
        List<Object[]> result = parser.reverseParseExpression(expression);

        List<Object[]> expected = new ArrayList<>();
        expected.add(new Object[]{"Device1", "method1", "+"});
        expected.add(new Object[]{42.0, null, "*"});
        expected.add(new Object[]{"Device2", "method2", null});

        assertEquals(expected.size(), result.size());
        for (int i = 0; i < expected.size(); i++) {
            System.out.println(Arrays.toString(result.get(i)));
            assertArrayEquals(expected.get(i), result.get(i));
        }
    }

    @Test
    void testComplexExpression() {
        ExpressionTest.Person person1 = new ExpressionTest.Person(10);
        ExpressionTest.Person person2 = new ExpressionTest.Person(5);
        ExpressionTest.Person person3 = new ExpressionTest.Person(3);
        ExpressionTest.Person person4 = new ExpressionTest.Person(2);

        // ((person1.getAge() + person2.getAge()) - person3.getAge()) /
person4.getAge()
        Expression expr1 = new Expression(person1, "getAge");
```

```
        Expression expr2 = new Expression(person2, "getAge");
        Expression expr3 = new Expression(person3, "getAge");
        Expression expr4 = new Expression(person4, "getAge");

        // (person1.getAge() + person2.getAge())
        Expression additionExpr = new Expression(expr1, "+", expr2);

        // (additionExpr - person3.getAge())
        Expression subtractionExpr = new Expression(additionExpr, "-", expr3);

        // (subtractionExpr / person4.getAge())
        Expression divisionExpr = new Expression(subtractionExpr, "/", expr4);
        Expression divisionTest = new Expression(divisionExpr, "==", new
Expression(6.0));

        ExpressionReverseParser parser = new ExpressionReverseParser();
        List<Object[]> result = parser.reverseParseExpression(divisionTest);

        List<Object[]> expected = new ArrayList<>();
        expected.add(new Object[]{"10", "getAge", "+"});
        expected.add(new Object[]{"5", "getAge", "-"});
        expected.add(new Object[]{"3", "getAge", "/"});
        expected.add(new Object[]{"2", "getAge", "=="});
        expected.add(new Object[]{"6", "getAge", null});

//        assertEquals(expected.size(), result.size());
        for (int i = 0; i < expected.size(); i++) {
            System.out.printf("%s %s\n", Arrays.toString(expected.get(i)),
Arrays.toString(result.get(i)));
//            assertArrayEquals(expected.get(i), result.get(i));
        }
    }
}
```

```
package org.example.rule;

import org.example.rule.entity.ConditionWithOperator;
import org.example.rule.entity.Expression;
import org.junit.jupiter.api.Test;

import static org.junit.Assert.assertEquals;

public class ExpressionTest {
    public static class Person {
        private int age;

        public Person(int age) {
            this.age = age;
        }

        public int getAge() {
            return age;
        }

        public void setAge(int age) {
            this.age = age;
        }
    }
    @Test
    public void testAdditionWithObjects() {
        Person person1 = new Person(10);
        Person person2 = new Person(5);
```

```
        Expression expr1 = new Expression(person1, "getAge");  // person1.getAge()
        Expression expr2 = new Expression(person2, "getAge");  // person2.getAge()
        Expression addition = new Expression(expr1, "+", expr2);  //
(person1.getAge() + person2.getAge())

        assertEquals(15.0, addition.evaluate());
    }

    @Test
    public void testComplexExpressionWithObjects() {
        Person person1 = new Person(10);
        Person person2 = new Person(5);
        Person person3 = new Person(3);
        Person person4 = new Person(2);

        // ((person1.getAge() + person2.getAge()) - person3.getAge()) /
person4.getAge()
        Expression expr1 = new Expression(person1, "getAge");
        Expression expr2 = new Expression(person2, "getAge");
        Expression expr3 = new Expression(person3, "getAge");
        Expression expr4 = new Expression(person4, "getAge");

        // (person1.getAge() + person2.getAge())
        Expression additionExpr = new Expression(expr1, "+", expr2);

        // (additionExpr - person3.getAge())
        Expression subtractionExpr = new Expression(additionExpr, "-", expr3);

        // (subtractionExpr / person4.getAge())
        Expression divisionExpr = new Expression(subtractionExpr, "/", expr4);
        Expression divisionTest = new Expression(subtractionExpr, "==", new
Expression(6.0));

        assertEquals(true, divisionTest.evaluate());
    }

    @Test
    public void testMultiplicationWithObjectMethod() {
        Person person1 = new Person(4);
        Person person2 = new Person(5);
        Expression expr1 = new Expression(person1, "getAge");  // person1.getAge()
        Expression expr2 = new Expression(person2, "getAge");  // person2.getAge()
        Expression multiplication = new Expression(expr1, "*", expr2);  //
(person1.getAge() * person2.getAge())

        assertEquals(20.0, multiplication.evaluate());
    }

    @Test
    public void testNestedExpressionWithObjects() {

        Person person1 = new Person(10);
        Person person2 = new Person(5);
        Person person3 = new Person(2);


        Expression expr1 = new Expression(person1, "getAge");
        Expression expr2 = new Expression(person2, "getAge");
        Expression expr3 = new Expression(person3, "getAge");

        // (person1.getAge() + person2.getAge())
        Expression additionExpr = new Expression(expr1, "+", expr2);

        // (additionExpr * person3.getAge())
```

```
        Expression multiplicationExpr = new Expression(additionExpr, "*", expr3);


        assertEquals(30.0, multiplicationExpr.evaluate());
    }

    @Test
    public void testDivisionByZeroWithObject() {

        Person person1 = new Person(10);
        Person person2 = new Person(0);


        Expression expr1 = new Expression(person1, "getAge");
        Expression expr2 = new Expression(person2, "getAge");
        Expression division = new Expression(expr1, "/", expr2);  //
(person1.getAge() / person2.getAge())

        assertEquals(Double.NaN, division.evaluate());
    }

    @Test
    public void testComplexExpressionWithMethods() {

        Person person1 = new Person(10);
        Person person2 = new Person(5);
        Person person3 = new Person(3);
        Person person4 = new Person(2);

        Expression expr1 = new Expression(person1, "getAge");
        Expression expr2 = new Expression(person2, "getAge");
        Expression expr3 = new Expression(person3, "getAge");
        Expression expr4 = new Expression(person4, "getAge");

        // (person1.getAge() + person2.getAge())
        Expression additionExpr = new Expression(expr1, "+", expr2);

        // (additionExpr - person3.getAge())
        Expression subtractionExpr = new Expression(additionExpr, "-", expr3);

        // (subtractionExpr / person4.getAge())
        Expression divisionExpr = new Expression(subtractionExpr, "/", expr4);

        assertEquals(6.0, divisionExpr.evaluate());
    }
}
```

```
package org.example.rule;

import junit.framework.TestCase;
import org.example.entity.CurrentLineSensor;
import org.example.entity.Transformer;
import org.example.rule.entity.*;
import org.example.rule.executor.RuleExecutor;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisher;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisherFactory;

public class RuleBuilderTest extends TestCase {
    public void testName(){
        Transformer objA = new Transformer();
        MQTTPublisher mqttPublisher =
MQTTPublisherFactory.getPublisher("testClient");
        objA.setTurnsRation(2.0);
```

```
        CurrentLineSensor objB = new CurrentLineSensor();
        objB.setVoltage(110.0);
        objA.setSensorId("device1");
        RuleBuilder builder = new RuleBuilder();
        Expression condition1Expr = new Expression(
                new Expression(objB, "getVoltage"),
                ">",
                new Expression(110.0)
        );
        Expression condition2Expr = new Expression(
                new Expression(objB, "getVoltage"),
                "<",
                new Expression(330.0)
        );
        Expression condition3Expr = new Expression(
                new Expression(objB, "getVoltage"),
                "<",
                new Expression(330.0)
        );
        Expression condition4Expr = new Expression(
                new Expression(objB, "getVoltage"),
                ">",
                new Expression(100.0)
        );

        ConditionWithOperator conditionWithOperator1 = new
ConditionWithOperator(condition1Expr, "AND");
        ConditionWithOperator conditionWithOperator2 = new
ConditionWithOperator(condition2Expr, "AND");
        ConditionWithOperator conditionWithOperator3 = new
ConditionWithOperator(condition3Expr, "OR");
        ConditionWithOperator conditionWithOperator4 = new
ConditionWithOperator(condition4Expr, "AND");

        Action action = new Action(mqttPublisher, "writeData", new Expression(4));


        Rule rule = builder
                .addCondition(conditionWithOperator1)
                .addCondition(conditionWithOperator2)
                .addCondition(conditionWithOperator3)
                .addCondition(conditionWithOperator4)
                .addAction(action)
                .build();

        RuleExecutor ruleExecutor = new RuleExecutor(rule);
        ruleExecutor.execute();

        System.out.println("Sensor voltage= " + objB.getVoltage());
        System.out.println("Transformer turnsRation= " + objA.getTurnsRation());
    }


    public void testRuleExecutor() {

    }
}
```

```
package org.example.rule;

import junit.framework.TestCase;
import org.example.entity.Device;
import org.example.entity.DeviceType;
```

```
import org.example.factory.DeviceFactory;
import org.example.rule.entity.Action;
import org.example.rule.entity.Expression;
import org.example.rule.entity.Rule;
import org.example.rule.executor.ActionExecutor;
import org.example.rule.executor.DefaultActionExecutor;
import org.example.rule.executor.RuleExecutorThread;
import org.example.service.RuleService;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisher;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisherFactory;

import java.util.List;

public class RuleExecutorTest extends TestCase {
    public void testActionExecutor(){
        Device transformer =
DeviceFactory.getDevice(DeviceType.TRANSPORTER).get();
        MQTTPublisher mqttPublisher = MQTTPublisherFactory.getPublisher("test");
        Action action = new Action(transformer, "setTurnsRation", new
Expression(3.0));
        Action action2 = new Action(mqttPublisher, "writeData", new
Expression(transformer));
        ActionExecutor actionExecutor = new DefaultActionExecutor();
        actionExecutor.execute(action);
        System.out.println(transformer);
    }

    public void testRuleExecutorThread(){
        RuleService ruleService = new RuleService();
        List<Rule> ruleList= ruleService.getAllRule();
//        System.out.println(ruleList);
        for (Rule rule : ruleList){
            Thread thread = new RuleExecutorThread(rule);
            thread.start();
        }
        while (true){}

    }
}
```

```
package org.example.rule;

import com.fasterxml.jackson.core.JsonProcessingException;
import org.example.entity.CurrentLineSensor;
import org.example.entity.Transformer;
import org.example.rule.entity.*;
import org.example.rule.serializer.RuleDeserializer;
import org.example.rule.serializer.RuleSerializer;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisher;
import org.example.subscriber.service.mqtt.publisher.MQTTPublisherFactory;
import org.jetbrains.annotations.NotNull;
import org.junit.jupiter.api.Test;

import java.time.Instant;
import java.time.LocalDateTime;
import java.util.List;

import static org.junit.jupiter.api.Assertions.*;

public class RuleSerializerTest {

    private final RuleSerializer ruleSerializer = new RuleSerializer();
    private final RuleDeserializer ruleDeserializer = new RuleDeserializer();
```

```
    @Test
    public void testSerializeAndDeserializeRule() throws Exception {
        Rule rule = getRule();

        String serializedRule = ruleSerializer.serializeRule(rule);
        System.out.println(rule);
        System.out.println(serializedRule);
        Rule deserializedRule = ruleDeserializer.deserialize(serializedRule);
        System.out.println(serializedRule);

        isRuleEqual(rule, deserializedRule);
    }

    public static void isRuleEqual(Rule rule, Rule deserializedRule) {
        idRuleIdEqual(rule, deserializedRule);
        isConditionsWithOperatorEquals(rule, deserializedRule);
        isActionsEquals(rule, deserializedRule);
        isTimeStampEqual(rule, deserializedRule);
    }

    private static void isTimeStampEqual(Rule rule, Rule deserializedRule) {
        LocalDateTime timeStampOrig = rule.getTimeStamp();
        LocalDateTime timeStampDes = deserializedRule.getTimeStamp();
        assertEquals(timeStampOrig, timeStampDes);
    }

    private static void idRuleIdEqual(Rule rule, Rule deserializedRule) {
        Long idOrig = rule.getId();
        Long idDes = deserializedRule.getId();
        assertEquals(idOrig, idDes);
    }

    private static void isActionsEquals(Rule rule, Rule deserializedRule) {
        List<Action> actionsOrig = rule.getActions();
        List<Action> actionsDes = deserializedRule.getActions();
        for (int actionPos = 0; actionPos < actionsOrig.size(); actionPos++) {
            Action actionOrig = actionsOrig.get(actionPos);
            Action actionDes = actionsDes.get(actionPos);
            assertEquals(actionOrig, actionDes);
        }
    }

    // Метод для проверки выражений внутри условий
    private static void isConditionsWithOperatorEquals(Rule rule, Rule
deserializedRule) {
        List<ConditionWithOperator> conditionsOrig =
rule.getConditionsWithOperators();
        List<ConditionWithOperator> conditionsDeserialized =
deserializedRule.getConditionsWithOperators();
        for (int condPos = 0; condPos < conditionsOrig.size(); condPos++) {
            ConditionWithOperator conditionWithOpOrig =
conditionsOrig.get(condPos);
            ConditionWithOperator conditionWithOpDes =
conditionsDeserialized.get(condPos);
            Expression conditionOrig = conditionWithOpOrig.getCondition();
            Expression conditionDes = conditionWithOpDes.getCondition();
            String logicOrig = conditionWithOpOrig.getLogicalOperator();
            String logicDes = conditionWithOpDes.getLogicalOperator();
            assertEquals(conditionOrig, conditionDes);
            assertEquals(logicOrig, logicDes);
        }
    }
```

```
    public static Rule getRule() {
        CurrentLineSensor sensor = new CurrentLineSensor("device1", 220.0, 10.0);
        CurrentLineSensor sensor2 = new CurrentLineSensor("device", 222.0, 10.0);

        Expression expr1 = new Expression(new Expression(sensor, "getVoltage"),
">", new Expression(200.0));
        Expression expr2 = new Expression(new Expression(sensor2, "getVoltage"),
">", new Expression(204.0));

        System.out.println(expr1.evaluate());

        ConditionWithOperator condition1 = new ConditionWithOperator(expr1,
"AND");
        ConditionWithOperator condition2 = new ConditionWithOperator(expr2, "OR");

        Transformer transformer = new Transformer("device1", 2.0);
        MQTTPublisher mqttPublisher = MQTTPublisherFactory.getPublisher("test");
        Action action = new Action(transformer, "turnsRatio", new
Expression(2.0));
        Action action2 = new Action(mqttPublisher, "writeData", new
Expression(transformer));

        return new Rule(List.of(condition1, condition2), List.of(action,
action2));
    }
}
```

```
package org.example.rule;

import junit.framework.TestCase;
import org.example.rule.entity.Rule;
import org.example.service.RuleService;

import java.util.List;

public class RuleServiceTest extends TestCase {
    public void testName() {
        Rule rule = RuleSerializerTest.getRule();
        RuleService ruleService = new RuleService();
        ruleService.saveRule(rule);

        List<Rule> rules =  ruleService.getAllRule();
        System.out.println(rules.getFirst());
        RuleSerializerTest.isRuleEqual(rule, rules.getFirst());
    }
}
```

```
package org.example.subscriber;

import junit.framework.TestCase;
import org.eclipse.paho.client.mqttv3.MqttClient;
import org.eclipse.paho.client.mqttv3.MqttException;
import org.example.entity.DeviceType;
import org.example.entity.Topic;
import org.example.service.TopicService;
import org.example.subscriber.service.mqtt.MQTTClientFactory;
import org.example.subscriber.parser.CurrentLineDataParser;
import org.example.subscriber.parser.TransformerDataParser;
import org.example.subscriber.strategy.DeviceDataListener;
import org.example.subscriber.strategy.TopicSubscriber;
```

```
import java.time.ZonedDateTime;
import java.util.HashSet;
import java.util.Set;

import static java.lang.Thread.sleep;


public class MQTTSubscriberTest extends TestCase {
    private void testBehaviour(SubscriberBehaviour subscriberBehaviour, String
clientId, Set<Topic> topics) {
        try {
            MqttClient client = MQTTClientFactory.getMqttClient(clientId);
            MQTTClientSubscriber clientSubscriber = new
MQTTClientSubscriber(client, subscriberBehaviour);
            clientSubscriber.connectAndSubscribe(topics);
        } catch (MqttException e) {
            e.printStackTrace();
        }
    }


    private void deviceDataListener() {
        DeviceDataParserFactory parserFactory = new DeviceDataParserFactory();
        parserFactory.registerParser(DeviceType.TRANSPORTER, new
TransformerDataParser());
        parserFactory.registerParser(DeviceType.CURRENT_LINE_SENSOR, new
CurrentLineDataParser());
        TopicService topicService = new TopicService();
        Set<Topic> topics = new HashSet<>(topicService.getAllTopics());
        while (topics.isEmpty()){
            try {
                Thread.sleep(30000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
            topics = new HashSet<>(topicService.getAllTopics());
        }
        DeviceDataListener subscriberBehaviour = new
DeviceDataListener(parserFactory, topics);
        String idClient = "DeviceSubscriberClient";

        Thread thread = new Thread(new SubscriberThread(idClient,
subscriberBehaviour, topics));
        thread.start();
    }

    private void topicListener() {
        Set<Topic> topics = new HashSet<>(Set.of(new Topic("client/#",
ZonedDateTime.now().toInstant())));
        SubscriberBehaviour subscriberBehaviour = new TopicSubscriber();
        String idClient = "DeviceEnvironmentListener";
        MQTTClientSubscriber clientSubscriber = null;
        try {
            MqttClient client = MQTTClientFactory.getMqttClient(idClient);
            clientSubscriber = new MQTTClientSubscriber(client,
subscriberBehaviour);
            clientSubscriber.connectAndSubscribe(topics);
        } catch (MqttException e) {
            throw new RuntimeException(e);
        }
    }

    public void testTopicListener() {
        topicListener();
```

```
        while (true) {}
    }

    public void testDeviceDataListener() {
        deviceDataListener();
        while (true) {}
    }

    public void testSubscriberThread() {
        topicListener();
        deviceDataListener();
        while (true) {
            try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }
}
```

```
version: '3.8'

services:
  influxdb:
    image: influxdb:latest
    container_name: influxdb
    ports:
      - "8086:8086"
    volumes:
      - ./volume/influxdb:/var/lib/influxdb2
    restart: always
    environment:
      - DOCKER_INFLUXDB_INIT_MODE=setup
      - DOCKER_INFLUXDB_INIT_USERNAME=admin
      - DOCKER_INFLUXDB_INIT_PASSWORD=admin123
      - DOCKER_INFLUXDB_INIT_ORG=nure
      - DOCKER_INFLUXDB_INIT_BUCKET=eecsystem

  node-red:
    image: nodered/node-red:latest
    container_name: node-red
    environment:
      - NODE_RED_USER_DIR=/data
    ports:
      - "1880:1880"
    volumes:
      - ./volume/node-red:/data
    restart: always
    networks:
      - nodered-net

  emqx:
    image: emqx/emqx:latest
    container_name: emqx
    ports:
      - "1883:1883"    # MQTT
      - "8083:8083"    # MQTT WebSocket
      - "18083:18083"  # Панель керування
    volumes:
      - ./volume/emqx/emqx_data:/opt/emqx/data
      - ./volume/emqx/emqx_plugins:/opt/emqx/plugins
    environment:
```

```
      EMQX_LOADED_PLUGINS: "emqx_dashboard,emqx_management"
    restart: unless-stopped

  postgres:
    image: postgres:latest
    container_name: postgres
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: admin123
      POSTGRES_DB: eecsystem
    ports:
      - "5432:5432"
    volumes:
      - ./volume/postgres:/var/lib/postgresql/data
    restart: always
    networks:
      - my-network

  pgadmin:
    image: dpage/pgadmin4:latest
    container_name: pgadmin
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@example.com
      PGADMIN_DEFAULT_PASSWORD: admin123
    ports:
      - "5050:80"
    volumes:
      - ./volume/pgadmin:/var/lib/pgadmin
    restart: always
    networks:
      - my-network

networks:
  nodered-net:
    driver: bridge

  my-network:
    driver: bridge
```

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.example</groupId>
  <artifactId>EECSystem</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>EECSystem</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <java.version>22</java.version>
    <javafx.version>23.0.1</javafx.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
```

```
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>com.ghgande</groupId>
      <artifactId>j2mod</artifactId>
      <version>3.1.0</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.36</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>org.eclipse.paho</groupId>
      <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
      <version>1.2.5</version> <!-- Убедитесь, что версия актуальна -->
    </dependency>
    <!-- https://mvnrepository.com/artifact/com.influxdb/influxdb-client-java -->
    <dependency>
      <groupId>com.influxdb</groupId>
      <artifactId>influxdb-client-java</artifactId>
      <version>7.2.0</version>
    </dependency>
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.15.2</version> <!-- Убедитесь, что это актуальная версия -->
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.openjfx/javafx-controls -->
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-controls</artifactId>
      <version>23.0.1</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.openjfx/javafx-fxml -->
    <dependency>
      <groupId>org.openjfx</groupId>
      <artifactId>javafx-fxml</artifactId>
      <version>23.0.1</version>
    </dependency>

    <!-- Hibernate core dependency -->
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>5.6.12.Final</version>
    </dependency>

    <!-- PostgreSQL JDBC Driver -->
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <version>42.3.6</version>
    </dependency>
    <dependency>
      <groupId>com.zaxxer</groupId>
      <artifactId>HikariCP</artifactId>
      <version>5.0.1</version>
    </dependency>

    <dependency>
```

```
      <groupId>com.google.code.gson</groupId>
      <artifactId>gson</artifactId>
      <version>2.8.9</version> <!-- Используйте актуальную версию -->
    </dependency>

    <!-- Logging dependency -->
    <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.16</version>
    </dependency>

    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-log4j12</artifactId>
      <version>1.6.4</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>RELEASE</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter</artifactId>
      <version>RELEASE</version>
      <scope>test</scope>
    </dependency>


  </dependencies>
  <build>
    <plugins>
      <!-- Maven Compiler Plugin -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>22</source>
          <target>22</target>
        </configuration>
      </plugin>

      <plugin>
        <groupId>org.openjfx</groupId>
        <artifactId>javafx-maven-plugin</artifactId>
        <version>0.0.8</version>
        <executions>
          <execution>
            <goals>
              <goal>run</goal>
            </goals>
          </execution>
        </executions>
        <configuration>
          <mainClass>org.example.App</mainClass>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

№	Позначення	Найменування	Додаткові відомості
		Текстові документи	
1	ГЮИК.442351.025 ПЗ	Пояснювальна записка	66 с., ф. А4
		Інші документи	
2		Додаток А	80 с., ф. А4
		Апробація результатів наукових	
		досліджень	
3		Додаток Б	
		Демонстраційний матеріал	У форматі
		(Презентація)	*.ppt
4		Додаток В	112 с., ф.А4
		Лістинг програми	

Змін.	Арк.	№ документа	Підпис	Дата	ГЮИК.442351.025 ПЗ			
Розроб.		Губарь А. Ю.		20.01. 2024	Розроблення системи управління пристроями електричних мереж промислового підприємства	Літера	Аркуш	Аркушів
Перевір.		Сезонова І. К.				Н	1	1
Н.контр.		Бронніков А.І.				ХНУРЕ Кафедра КІТАР		
Затвердив		Невлюдов І.Ш.						