МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2024 г.

Разработка веб-интерфейса сервера конфигурации и мониторинга
ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

Автор работы,
студент группы КЭ-405
_____С.А. Кирдяшкин
«___»_____2024 г.

Нормоконтролёр,
ст. преп. каф. ЭВМ
_____ С.В. Сяськов
«___»_____2024 г.

Челябинск-2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Кирдяшкину Сергею Александровичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

Тема работы: «Разработка веб-интерфейса сервера конфигурации и мониторинга» утверждена приказом по университету от 22 апреля 2024 г. № 764-13/12.

Срок сдачи студентом законченной работы: 01 июня 2024 г.

Исходные данные к работе:

Требования к функционалу разрабатываемого приложения:

— интерактивный интерфейс;

— авторизация по паролю;

— возможность выбрать различную конфигурацию оборудования;

— язык разработки: C#, платформа .NET 8, Javascript.

Перечень подлежащих разработке вопросов:

Выпускная квалификационная работа должна содержать разработку следующих вопросов:

1. Аналитический обзор аналогов и основных технологических решений для реализации проекта.
2. Определение требований.
3. Разработка веб-интерфейса.
4. Тестирование.

Дата выдачи задания: ___ декабря 2023 г.

Руководитель работы _____/ Д.В. Топольский/

Студент _____/ Кирдяшкин С.А. /

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Аналитический обзор аналогов и основных технологических решений для реализации проекта	10.03.2024	
Определение требований	21.03.2024	
Разработка веб-интерфейса	04.04.2024	
Тестирование	25.04.2024	
Компоновка текста работы и сдача на нормоконтроль	16.05.2024	
Подготовка презентации и доклада	24.05.2024	

Руководитель работы _____/ Д.В. Топольский /

Студент _____/ С.А. Кирдяшкин /

АННОТАЦИЯ

Кирдяшкин С.А Разработка веб-интерфейса сервера конфигурации и мониторинга – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 43 с., библиогр. список – 29 наим.

Тема выпускной квалификационной работы – Разработка веб-интерфейса сервера конфигурации и мониторинга. Работа является частью проекта «Разработка и создание производства следящих гидроприводов с гидростатическими направляющими (СГ с ГСН)».

В ходе работ был проведён аналитический обзор аналогов и основных технологический решений для разработки решения. Определены основные требования к проекту, как функциональные, так и нефункциональные.

Была спроектирована, разработана и протестирована на соответствие запланированным требованиям клиентская и серверная части приложения.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Актуальность разработки веб-интерфейса сервера конфигурации и мониторинга заключается в том, что в эпоху интернета веб-интерфейсы играют решающую роль в облегчении коммуникации и взаимодействия между пользователями и системами. Веб-интерфейс служит инструментом для доступа пользователей к приложениям, веб-сайтам и различным цифровым сервисам и взаимодействия с ними. Это фасад системы, обеспечивающий визуальное представление сложных функциональных возможностей в удобной для пользователя форме.

Работа является частью проекта «Разработка и создание производства следящих гидроприводов с гидростатическими направляющими (СГ с ГСН)» и создание веб-интерфейса напрямую необходимо предприятию.

За прошедшие годы веб-сайты значительно эволюционировали, перейдя от базовых статических страниц к динамическим и интерактивным платформам. С развитием таких технологий, как HTML, CSS, JavaScript и PHP они стали более отзывчивыми, визуально привлекательными и многофункциональными. Теперь они включают в себя такие элементы, как анимация, обновления в режиме реального времени, функциональность перетаскивания и мультимедийный контент для улучшения взаимодействия с пользователем.

Дизайн веб-интерфейса имеет решающее значение, поскольку он напрямую влияет на удобство использования. Хороший интерфейс должен быть интуитивно понятным, простым в использовании, иметь привлекательный внешний вид, а также быть адаптированным к различным размерам экрана и устройствам. Он также должен уделять внимание доступности, гарантируя, что пользователи смогут эффективно взаимодействовать с системой с любой платформы.

Цель работы – создание веб-интерфейса сервера конфигурации и мониторинга.

Для достижения этой цели необходимо:

1. Провести анализ предметной области.

1.1 Обозреть аналоги и проанализировать их слабые и сильные стороны.

2.1 Провести анализ популярных веб технологий и методологий разработки.

2. Определить требования к системе.

3. Создать веб-интерфейс

3.1 Спроектировать систему с учётом технического задания и требований к системе.

3.2 Реализовать клиентскую и серверную части.

4. Протестировать готовый интерфейс на соответствие требованиям предприятия и запланированным характеристикам.

1. АНАЛИТИЧЕСКИЙ ОБЗОР АНАЛОГОВ И ОСНОВНЫХ ТЕХНОЛОГИЧЕСКИХ РЕШЕНИЙ ДЛЯ РЕАЛИЗАЦИИ ПРОЕКТА

1.1. Обзор аналогов

В данный момент на предприятии используется программное обеспечение итальянского производства – Atos E-SW. На рисунке 1 показана страница выбора нужной конфигурации. Пользователь программы выбирает нужную ему конфигурацию двойным нажатием мыши, после чего происходит переход на страницу функциональной схемы выбранной конфигурации.

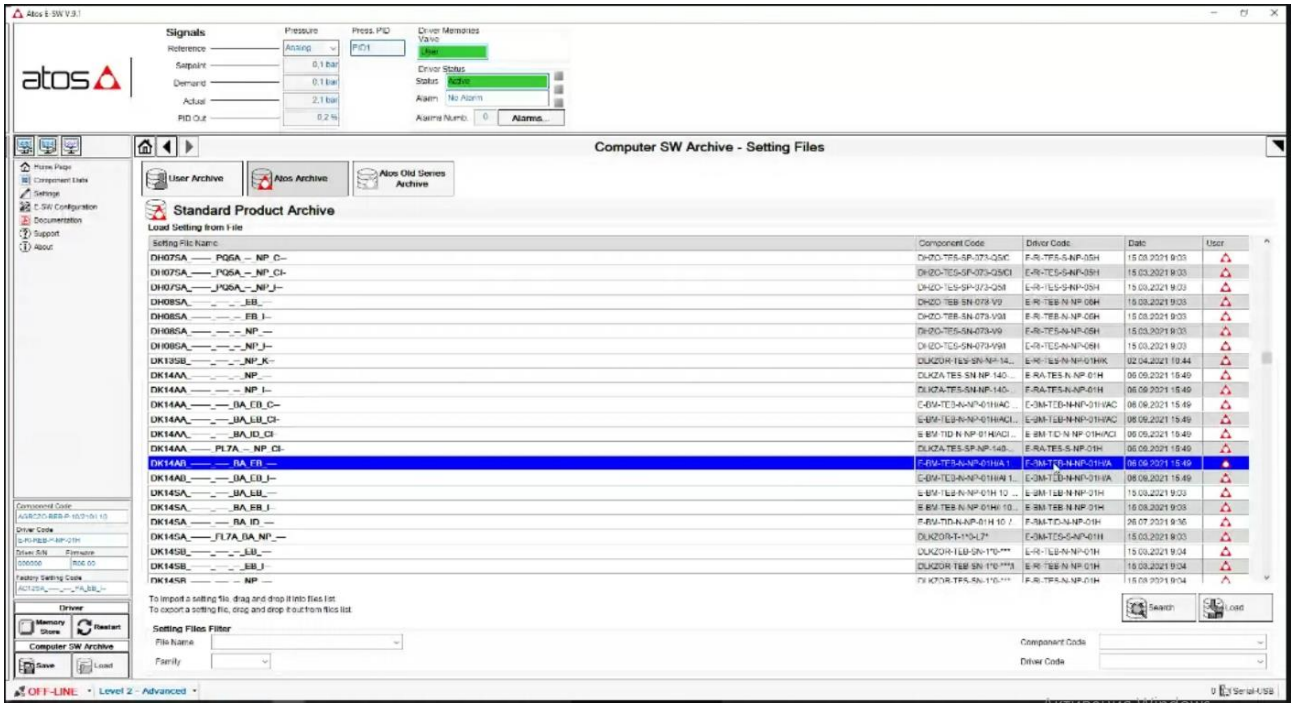

Рисунок 1 – выбор конфигурации

На рисунке 2 изображена функциональная схема выбранного прибора. Она представляет собой блок-схему элементов некоего оборудования, при нажатии на которые можно открыть ещё одну, более подробную схему.

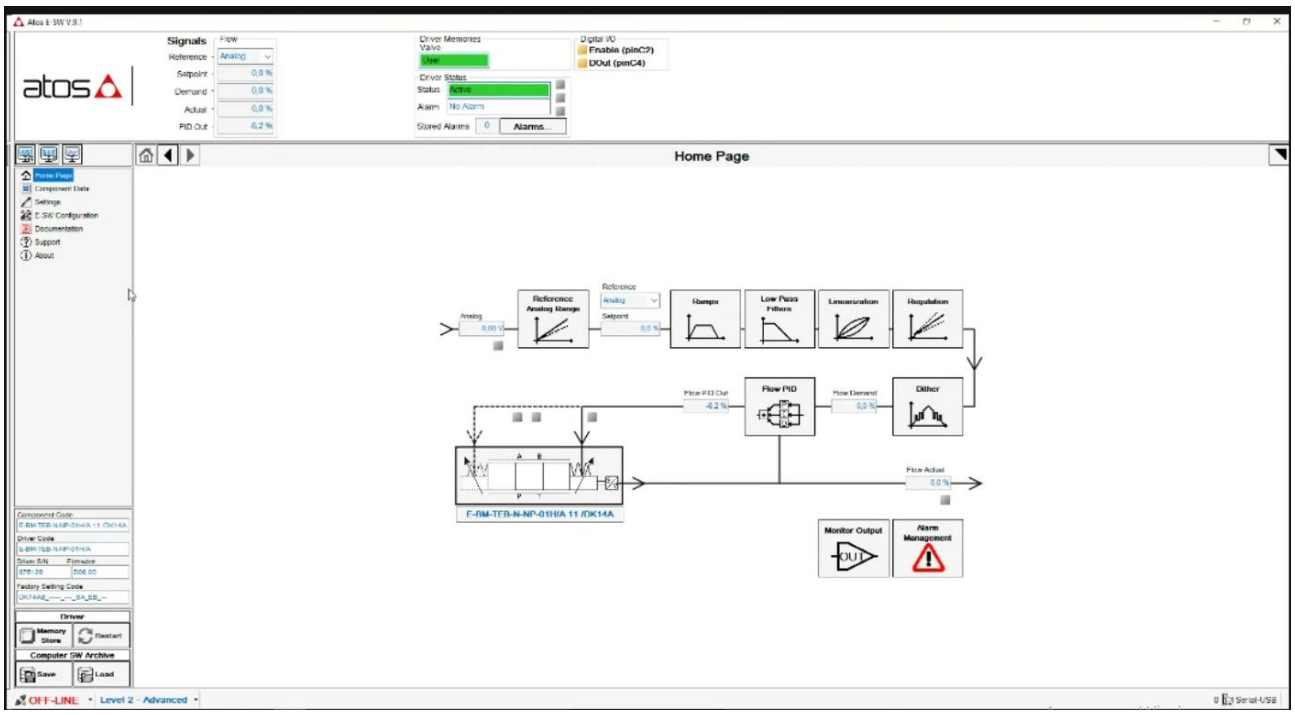

Рисунок 2 – функциональная схема

Интерфейс программы обладает следующими недостатками:

1. Интерфейс программы не русифицирован.
2. При выборе новой конфигурации, переключении вида, сохранении текущей конфигурации, интерфейс программы полностью обновляется, что занимает некоторое время.

И следующими преимуществами:

1. Интерфейс программы достаточно интуитивен.
2. Интерфейс удобный и читаемый.

Но главный недостаток Atos E-SW заключается в отсутствии веб-приложения. Мониторинг параметров производственного комплекса возможен лишь с подключенного стационарного компьютера.

1.2. Анализ основных технологических решений

Ниже представлен обзор доступных технологий и инструментов, которые могут быть задействованы для реализации проекта. Это дает возможность проанализировать плюсы и минусы каждой технологии, а также выбрать оптимальный набор программных средств для достижения поставленных задач.

Выбор программного средств играет ключевую роль. Правильный выбор упрощает и ускоряет процесс разработки, а также гарантирует

функциональность и масштабируемость созданного приложения. Поскольку разрабатывается веб-приложение, выбор языка программирования должен быть обоснован его возможностями в сфере веб-разработки.

При выборе необходимо учитывать:

— его популярность и поддержку в сообществе разработчиков. Продукт с обширным сообществом может быть более надежным и безопасным выбором;

— порог вхождения. Изучение языка программирования с высоким порогом вхождения может занять время, в то время как язык с низким порогом вхождения будет изучен и применён гораздо быстрее;

— выбранное средство должен покрывать все требования проекта.

Далее изучим основные языки, используемые для разработки фронтенда приложения.

HyperText Markup Language являет собой стандартный язык гипертекстовой разметки, применяемый для создания содержимого веб-страниц. Он не является языком программирования в классическом понимании, но необходим для любой веб разработки, он определяет структуру документа, как содержание должно быть представлено на странице. Чаще всего он используется в сочетании с CSS и Javascript.

Cascading Style Sheets – это параметры форматирования страницы, определяющий внешний вид документа, также не является полноценным языком программирования, но необходим для стилизации и оформления веб страниц. CSS описывает внешний вид элементов, их расположение, цвета, шрифты и другие аспекты дизайна.

JavaScript – это язык программирования, который дает возможность создавать динамично обновляемое содержимое страницы и управлять им, а также создавать различную графику, анимацию и многое другое [3].

Рассмотрим основные преимущества:

— Javascript является самым популярным языком программирования среди разработчиков на 2023 год [1];

— Javascript имеет множество библиотек и фреймворков, которые позволяют расширить его функционал и ускорить разработку.

Рассмотрим недостатки:

— динамическая типизация и неявное преобразование: компилятор не строго следит за типами данных и может производить нежелательные преобразования, что приводит к разночтениям в интерпретации данных, увеличению количества ошибок и замедлению процесса компиляции;

— уязвимость для злоумышленников: в JavaScript возможно внедрить вредоносный код, который может причинить ущерб пользователю.

Заместить JavaScript в браузере может Typescript. Typescript – это язык программирования со статической типизацией, язык расширяющий возможности Javascript. Помимо того, что он лишён как недостатков, так и преимуществ, которые даёт динамическая типизация, Typescript обладает расширенным функционалом в области ООП, позволяет настраивать модификаторы доступа к классам и объектам. Typescript компилируется напрямую в JavaScript код, также обладает обратной совместимостью.

Недостатками TypeScript, по сравнению с JavaScript являются:

— порог входа: помимо того, что нужно изучить Javascript на приемлемом для разработки уровне, нужно дополнительно изучать принципы работы TypeScript и различные типы данных используемые в нём;

— совместимость: в том случае если какая-либо библиотека, фреймворк или компонент не поддерживают TypeScript придётся потратить время описывая все используемые типы данных.

JavaScript фреймворк – это программная платформа определяющая структуру системы программы, она предоставляет инструменты, которые помогают структурировать код и улучшить производительность веб-приложений на JavaScript, реализуют готовые решения для создания различного функционала и облегчают работу разработчиков, особенно при создании крупных и сложных проектов.

Ключевые преимущества использования JavaScript фреймворков включают:

— ускорение разработки: Фреймворки предлагают готовые компоненты, модули и инструменты, которые позволяют ускорить процесс разработки, так как разработчику не нужно писать все с нуля;

— структурирование кода: Фреймворки часто предлагают определенную архитектуру приложения, что помогает разработчикам организовать свой код и легче его поддерживать;

— удобство использования: Фреймворки обеспечивают удобный API и инструменты для управления состоянием, маршрутизации, запросами к серверу и другими аспектами разработки.

Использование фреймворков JavaScript помогает сделать разработку веб-приложений более эффективной, структурированной и удобной, что в конечном итоге приводит к созданию качественных и производительных приложений.

Ниже приведены самые используемые фреймворки JavaScript:

React – это созданный компанией Facebook фреймворк, представляющий собой мощную инструмент для создания пользовательских интерфейсов. Основанный на компонентной структуре, он даёт возможность создавать повторно используемые компоненты пользовательского интерфейса.

Ключевые преимущества React:

— виртуальный DOM (Document Object Model): React использует виртуальное представление DOM, что позволяет эффективно обновлять только необходимые части страницы, улучшая производительность;

— однонаправленный поток данных: React передаёт данные сверху вниз, обеспечивая простоту отслеживания изменений и уменьшая возможности ошибок;

— JSX: React использует JSX - синтаксис, который позволяет объединять JavaScript и HTML, улучшая читаемость и обслуживаемость кода;

— расширяемость и удобство тестирования: React легко расширяется с помощью сторонних библиотек и инструментов, что облегчает процесс разработки и тестирования.

Angular – это фреймворк с открытым исходным кодом, разработанный компанией Google для создания клиентских приложений, с акцентом на создание одностраничных приложений (SPA). Он представляет собой эволюцию предыдущего инструмента AngularJS, полностью переписанного командой, создавшей его [4].

Вот несколько ключевых преимуществ Angular:

— двустороннее связывание данных: Angular использует двустороннее связывание данных, что позволяет автоматически отслеживать и обновлять изменения в модели и представлении без необходимости ручного вмешательства;

— компонентный подход: архитектура, основанная на компонентах, упрощает созданию модульной и переиспользуемой структуры приложения;

— мощный маршрутизатор: Angular предоставляет встроенный маршрутизатор для управления навигацией в приложении, позволяя определять маршруты и представления;

— многообразие директив: Angular обеспечивает множество встроенных и настраиваеммых директив для управления DOM, что способствует созданию интерактивных пользовательских интерфейсов;

— модульность и инкапсуляция: Angular поддерживает модульность и инкапсуляцию, позволяя разработчикам создавать структурированный и чистый код.

Vue.js — прогрессивный фреймворк JavaScript для создания интерактивных пользовательских интерфейсов. В отличие от традиционных фреймворков монолитов, таких как Laravel, Vue разработан с учетом возможности постепенного внедрения. Основная задача Vue в первую очередь решает задачи уровня представления, что упрощает интеграцию с другими библиотеками и существующими проектами. С другой стороны, Vue идеально

подходит и для разработки сложных одностраничных приложений, при использовании современных инструментов и дополнительных библиотек [5].

Преимущества:

— легкость изучения и интеграции: Vue.js легок для освоения даже новичкам и может быть постепенно интегрирован в существующие проекты без проблем;

— простота и гибкость: Vue.js предлагает простой и интуитивно понятный синтаксис, а также гибкие возможности для создания интерфейсов и компонентов;

— реактивность: Vue.js обеспечивает реактивное поведение данных, что означает автоматическое изменение страницы при модификации данных, упрощая разработку интерактивных приложений;

— малый размер и быстрая скорость: Vue.js имеет компактный размер, что способствует быстрой загрузке приложений, а также обеспечивает высокую производительность, он является одним из самых быстрых фреймворков [6];

— поддержка компонентов: Vue.js поощряет создание многократно используемых компонентов, что упрощает разделение интерфейса на мелкие части и повторное использование кода.

В результате аналитического обзора основных фреймворков было принято решение в пользу Vue.js из-за лёгкости освоения, компактный размер и высокую производительность, а также наличия большого совместимых библиотек для реализации функционала приложения.

Помимо фреймворка Vue.js в проекте задействуются следующие библиотеки:

1. Pinia – это библиотека для создания централизованного хранилища состояний для Vue, которая позволяет использовать эти состояния между компонентами и страницами [27]. Pinia была выбрана вместо Vuex [11], так как имеет более простой и надёжный API и поддерживает наличия нескольких состояний в одном хранилище.

2. Axios – это библиотека, позволяющая делать запросы по протоколу http, преобразовывать данные запросов и ответов, а также преобразовывать данные поступающие в формате JSON [12].

3. VueFlow – это библиотека позволяющая легко создавать и настраивать различные диаграммы, блок-схемы и графические представления [15]. Была выбрана среди других библиотек, таких как mxGraph [16] и GoJS [17], так как она бесплатная (имеет лицензию MIT), совместима с Vue.js, поддерживается и обновляется разработчиками.

Серверная часть приложения предназначена для того, чтобы формировать контент сайта, хранить и обрабатывать различные данные, отправлять запросы в базы данных. Рассмотрим основные языки программирования и программные средства для реализации бэкенд-части приложения. При выборе предпочтений учитываются следующие критерии: производительность, удобство разработки, наличие инструментов, активное сообщество, безопасность.

Для создания серверной части приложения чаще всего используются следующие языки программирования: PHP, Go, Python, Ruby, C#, Javascript.

PHP – это распространенный язык программирования общего назначения с открытым исходным кодом. PHP был разработан непосредственно для использования в сфере веб-разработки и его код может быть напрямую встроен в HTML. Различия PHP и Javascript заключается в том, что PHP-скрипты выполняются на сервере и генерируют HTML, который посылается клиенту. Несмотря на то, что PHP изначально был создан для использования на веб-серверах, его применение не заканчивается на этом. [7].

Для создания серверной части приложения при помощи Python, создано множество различных фреймворков и библиотек, таких как Bottle, Flask, Django, Web2Py, CherryPy, Dash, Falcon и прочие.

Go или же golang — компилируемый, многопоточный, статически-типизированный язык программирования, разработанный компанией Google и обладающий открытым исходным кодом [22,23,24].

Преимущества Go:

— простота: простой синтаксис, отсутствие объектов и классов, механизмов наследования намного упрощает программу для начинающего программиста;

— большое количество библиотек: go обладает открытым кодом и большим сообществом, почти для каждой задачи существует библиотека. Также Go совместим с библиотеками, предназначенными для C и C++;

— высокая производительность.

Недостатки Go:

— простота: простота Go является и недостатком, так как отсутствие сложных функций присутствующих в других языках, ограничивает возможности Go;

— небольшой функционал: из-за своей простоты go не подходит для выполнения некоторых задач, таких как разработка графических интерфейсов;

— низкая популярность у работодателей: намного чаще в коммерческих проектах требуются программисты со знанием таких языков программирования как Javascript, Python, Java.

Преимущества Python:

— низкий порог вхождения: Python имеет простой, запоминающийся синтаксис, что значительно ускоряет его изучение и внедрение;

— активное сообщество: Python имеет большое и активное сообщество разработчиков опыт которых поможет в решении возникнувших проблем;

— большое количество разнообразных фреймворков для серверной разработки: для создания серверной части приложения могут быть использованы такие фреймворки как Bottle, Flask, Django, Web2Py, CherryPy, Dash, Falcon и прочие;

Недостатки Python:

— производительность: Python медленнее остальных языков программирования, т.к. является интерпретируемым, а не компилируемым
— GIL (Global Interpreter Lock): GIL в Python ограничивает возможность использования нескольких ядер процессора для выполнения кода, что снижает производительность и может привести к проблемам с параллельной обработки запросов.

Серверная часть приложения может быть создана при помощи языка Ruby, Ruby – это тщательно сбалансированный язык. Его создатель Юкихиро Мацумото, объединил части его любимых языков чтобы сформировать новый язык, в котором парадигма функционального программирования сбалансирована принципами императивного программирования [20, 21].

Для создания веб-проектов используется Ruby on Rails, это объектно-ориентированный фреймворк с открытым кодом. При работе с ним используется широко распространённый паттерн, шаблон программирования MVC.

Преимущества Ruby и Ruby on Rails:

— скорость разработки: Ruby on Rails упрощает разработку проекта и предлагает готовые к использованию решения;
— активное сообщество: для Ruby создано большое количество библиотек и гемов (gems);
— заинтересованность больших компаний: на Ruby on Rails созданы такие проекты как GitHub, GitLab, AirBnB, Twitch, Shopify, Fiverr, Twitter, InSales, UCHI.ru, Aviasales.

Недостатки Ruby и Ruby on Rails:

— производительность: являясь интерпретируемым, а не компилируемым, по производительности Ruby отстаёт от других языков;
— высокий порог входа: для использования Ruby on Rails необходимо изучить язык программирования Ruby, изучить используемые в Ruby концепции и парадигмы;

— динамическая типизация: отсутствие строгой типизации приводит к усложнению тестирования, выявить ошибки становится труднее.

Javascript может быть использован для реализации и бэкенд части приложения. Для этого была создана платформа Node.js. Она представляет собой среду выполнения кода на Javascript которая построена на основе движка JavaScript Chrome V8, который позволяет транслировать вызовы на языке JavaScript в машинный код [25, 26].

Преимущества Node.js:

— масштабируемость: Node.js легко масштабируется и может быть использован для создания микросервисов в распределенной архитектуре;

— единый язык программирования: Node.js позволяет использовать Javascript как на клиентской, так и на серверной стороне, что упрощает разработку приложения и упрощает обмен данными между клиентом и сервером.

Недостатки Node.js:

— отсутствие многопоточности: Node.js работает в однопоточном режиме, что может привести к проблемам с производительностью при обработке сложных задач;

— высокий порог вхождения: разработка приложений в Node.js требует глубокого понимания концепций асинхронного программирования.

Для облегчения развёртки и управления веб-приложения было выбрано программное обеспечение Docker [28, 29]. Оно позволяет собрать приложение со всеми зависимостями и окружением в контейнер, который может быть запущен на любой системе Linux.

1.3. Итоги

Анализ используемого аналога показал, что он не удовлетворяет запрошенным требованиям, что подтвердило актуальность разработки нового приложения.

В результате анализа основных технологических решений были выбраны следующие инструменты для разработки приложения:

1. Язык разметки HTML.
2. Язык стилей CSS.
3. Javascript в качестве языка программирования клиентской части приложения.
4. Javascript фреймворк Vue.js.
5. Библиотека управления состояниями Pinia.
6. Библиотека http запросов Axios.
7. Библиотека для создания блок-схем и отрисовки графов VueFlow.
8. C# в качестве языка программирования серверной части приложения.
9. Платформа Visual Studio ASP.Net.
10. Программное обеспечение для автоматизации развёртывания веб-приложения с поддержкой контейнеризации Docker.
11. Figma в качестве графического редактора для создания макетов сайта.

2. ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ

2.1. Определение функциональных требований

Функциональные требования (functional requirements) определяют, каким должно быть поведение продукта в тех или иных условиях. Они определяют поведение системы в разных ситуациях, чтобы пользователи смогли выполнить свои задачи (пользовательские требования) в рамках бизнес-требований. Функциональные требования описываются в форме традиционных утверждений со словами «должен» или «должна» [2, с.12]. Должны обеспечиваться следующие функциональные требования:

1. Вход для пользователя с помощью пароля.
2. Возможность диагностики параметров в режиме реального времени.
3. Возможность выбора уровня доступа.
4. Доступ к функционалу приложения после ввода пароля.
5. Визуализация функциональных блок-схем оборудования.
6. Совместимость ПО с СГ с ГСН, датчиком положения, а также с электрогидравлическим усилителем мощности.
7. Веб-сервер для сбора, хранения и анализа данных.
8. Регулярное резервное копирование.
9. Ведение журнала ошибок.

2.2 Определение нефункциональных требований

Нефункциональное требование - описание свойства или особенности, которым должна обладать система, или ограничение, которое должна соблюдать система.

Должны обеспечиваться следующие нефункциональные требования:

1. Удобный читаемый интерфейс.
2. Требования к временным характеристикам программного обеспечения веб-сервера определяются скоростью сети Ethernet и общей производительностью аппаратной части сервера.

3. Поддерживаемые ОС программного модуля и серверной системы управления базами данных: Windows 10 и выше, Linux Debian 12 и выше, Astra Linux 1.7.3 и выше.

4. Язык разработки: C#, платформа .NET 8

5. Всё программное обеспечение, необходимое для функционирования программного комплекса – свободное.

2.3 Итоги

Были разработаны функциональные и нефункциональные требования к приложению с учётом технического задания заказчика и текущих возможностей.

3. РАЗРАБОТКА ВЕБ-ИНТЕРФЕЙСА

3.1. Проектирование

Формально каждое веб-приложение можно разбить на 3 взаимно независимые части [13].

1. Модуль, который исполняется браузером.
2. Модуль, исполняемый на серверной стороне под управлением сервера.
3. База данных.

UML диаграмма разбития веб-приложения на части представлена на рисунке 3.

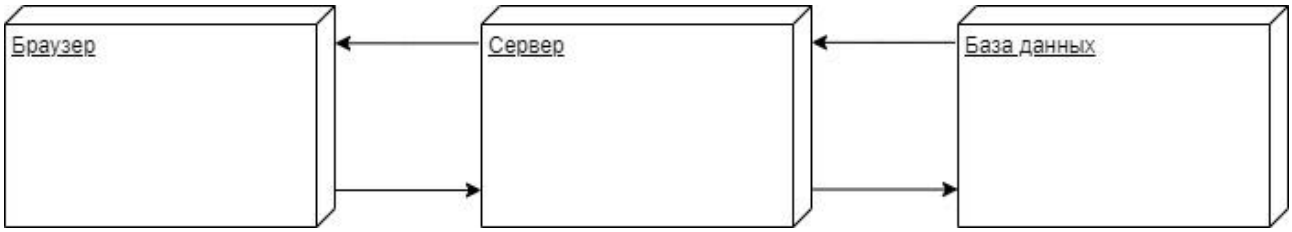

Рисунок 3 – архитектура веб-приложения

Более подробное рассмотрение архитектуры приложения представлено на рисунке 4.

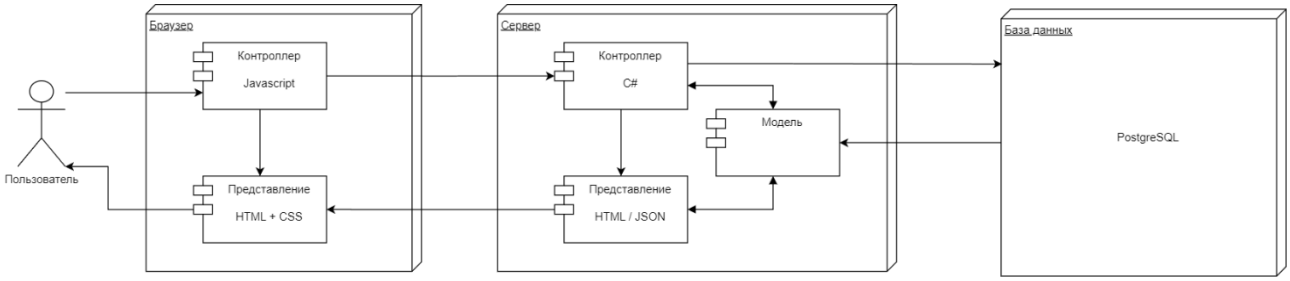

Рисунок 4 – подробная архитектура приложения

Пользователь видит перед собой страницу браузера, которая создаётся на основе представления присланного сервером, представление состоит из HTML и CSS кода. Обработкой действий пользователя в браузере занимается контроллер браузера, реализованный на языке Javascript, он обеспечивает динамичность страницы, посылает, принимает и обрабатывает запросы от сервера. Контроллер на сервере, реализованный на языке C# редактирует модель данных и

представления на основе принятых запросов от базы данных и контроллера браузера. После чего представление в вид HTML и CSS кода либо запрошенные данные в виде json файла отправляется обратно браузеру.

Приложение должно быть реализовано согласно концепции SPA, то есть одностраничного приложения, где страница обновляется автоматически без необходимости перезагружать её либо переходить на другие страницы. Технически, сайт использует единственный html документ, обновляемый сервером. Для этого должна быть тщательна продумана логика приложения, контент должен появляться и исчезать в рамках одной страницы.

Для реализации SPA, упрощения разработки, расширения и поддержки продукта в дальнейшем, приложение должно быть создано согласно паттерну разработки MVC. MVC (Model-View-Controller) – это схема разделения приложения на три отдельных компонента:

- модель, которая предоставляет данные и имеет возможность изменять своё состояние;
- представление, которое отвечает за визуализацию данных для пользователя;
- контроллер, который реагирует на действия пользователя, изменяет модель и представления.

Исходя из этого, необходимо создать представления, показываемые пользователю. В проекте представлениями являются Vue компоненты. Контроллеры браузерной и серверной стороны, которые будут изменять данные, отправлять запросы в базы данных и изменять модели. Модели, для хранения информации полученной от базы данных или пользователя.

3.2. Реализация

Начальная страница веб-приложения являет собой экранную форму выбора конфигурации и уровня доступа пользователя, она представлена на рисунке 5. При открытии этой страницы отправляется запрос на сервер, который обрабатывается контроллером ConfigsController, его исходный код представлен в листинге Б.2 приложения 2, с сервера приходят данные в виде JSon, эти данные

хранятся в централизованном хранилище pinia, в них хранятся конфигурации. За содержимое этой страницы отвечает представление LoginPage.vue, его исходный код представлен в листинге А.1 приложения А. В нём, а также в последующих листингах представлений не представлен код, отвечающий за стилизацию страницы в связи с большим объёмом и отсутствием практического смысла предоставлять его. Помимо HTML разметки и CSS стилей, в компоненте имеются два метода, OpenModal и CloseModal отвечающие за открытие и закрытие всплывающего модального окна запроса пароля.

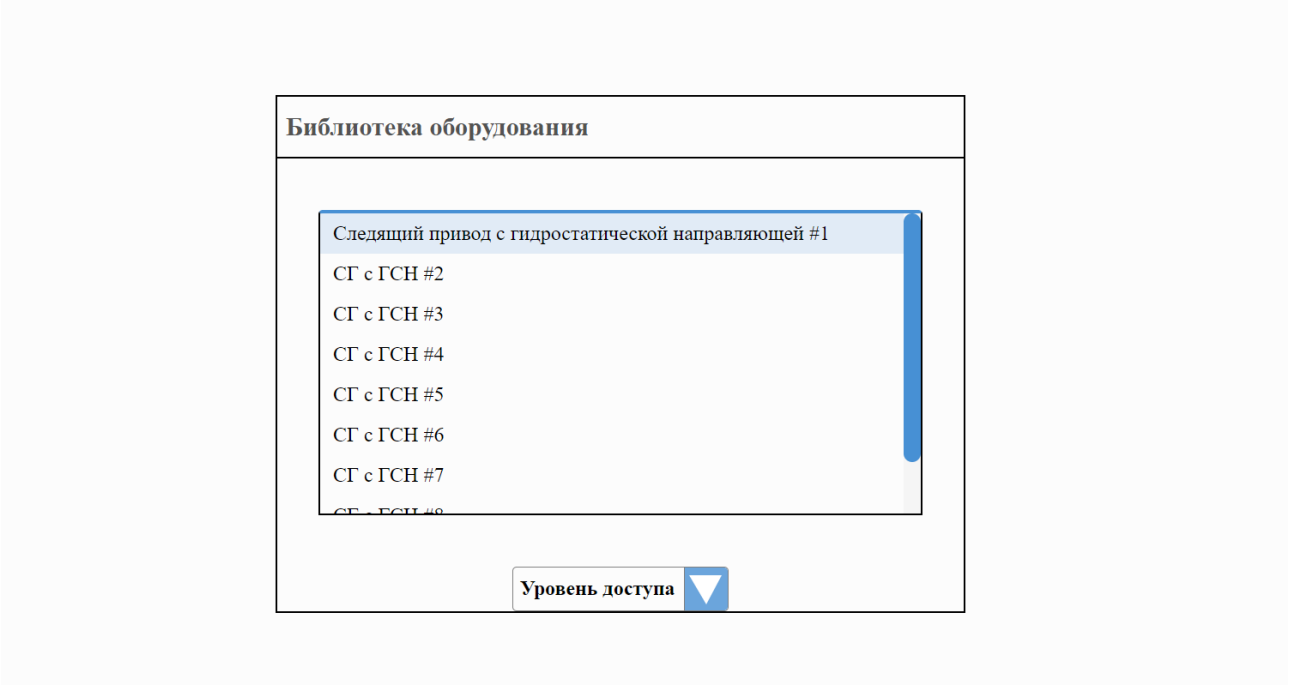

Рисунок 5 — начальная страница веб-приложения

При нажатии на кнопку «Уровень доступа», появляется выпадающее меню, на котором можно выбрать нужный уровень доступа. Результат нажатия на кнопку представлен на рисунке 6. За кнопку и выпадающее меню отвечает отдельный компонент AccessLevel.vue, его исходный код представлен в листинге А.2 приложения А. В компоненте представлен метод parentOpenModal, передающий действие нажатия на элемент списка в родительский компонент.

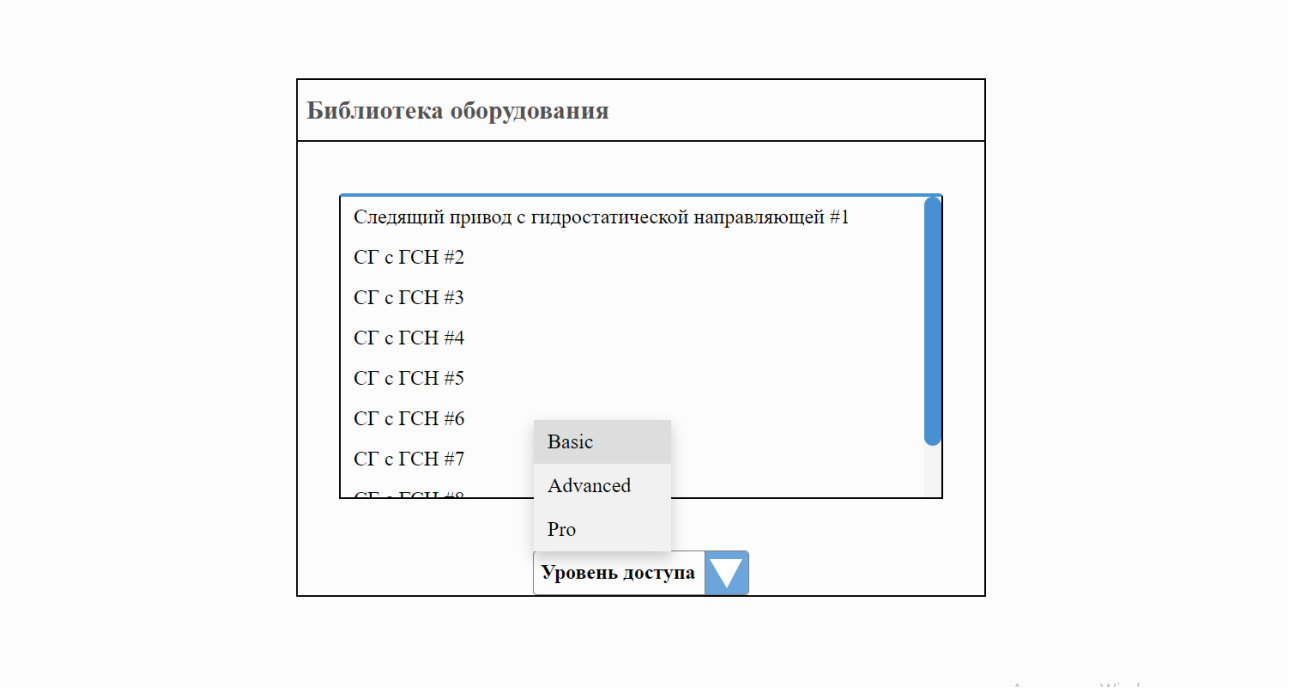

Рисунок 6 — выпадающее меню выбора уровня доступа

При выборе уровня доступа Basic, Advanced либо «Pro» появляется всплывающая экранная форма запроса пароля, она представлена на рисунке 7. За всплывающую экранную форму отвечает компонент «PasswordModal», его исходный код представлен в листинге А.3 приложения А. При нажатии на кнопку «Подтвердить» либо нажатии клавиши Enter с введённым паролем, вызывается метод updatePassword, он передаёт пароль в экземпляр централизованного хранилища Pinia, его исходный код представлен в листинге В.1 приложения В. А также вызывает метод sendPasswordToServer, который отправляет пароль на сервер и принимает от него ответ об успешности проверки пароля. В случае совпадения пароля, введённого пользователем, с паролем на сервере, значение переменной isAutenticated устанавливается true. Эта переменная является флагом для дальнейшей проверки авторизован пользователь или нет и каким уровнем доступа он обладает. В случае несовпадения пароля сервер возвращает ошибку.

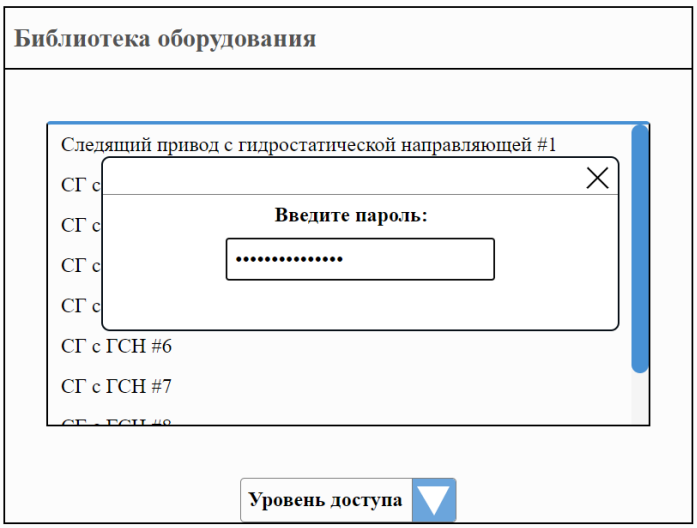

Рисунок 7 — всплывающая экранная форма запроса пароля

Обработкой пароля на серверной стороне занимается компонент HomeController.cs, его исходный код представлен в листинге Б.1 приложения Б. Там пароль сравнивается.

После установки переменной isAutenticated хранилища useUserPasswordStore, в компоненте App.vue, исходный код которого представлен в листинге А.4 приложения А, перестаёт отображаться начальная страница и начинает отображаться главная страница приложения согласно условиям v-if каждого из компонентов.

Новая страница с функциональной схемой представлена на рисунке 8. За её отображение отвечает компонент MainPage.vue, исходный код которого представлен в листинге А.4 приложения А. За реализацию блок-схемы на странице отвечает компонент VueFlow из одноимённой библиотеки. Этот компонент принимает на вход массив данных, содержащий информацию о элементах блок схем, линиях, соединяющих элементы и различных настройках стилизации элементов. Массив реактивно обновляется вместе с изменением данных в pinia. Данные хранятся в экземпляре централизованного хранилища Pinia useGraphStore исходный код которого представлен в листинге В.2 приложения В. Он содержит в себе метод запроса данных с сервера

При нажатии на какой-либо из элементов блок-схемы срабатывает событие, которое отправляет запрос на сервер, за обработку этого сервера отвечает контроллер GraphController, его исходный код представлен в листинге Б.3 приложения Б, он получает на вход id нажатого элемента и в ответ возвращает данные в формате JSon, которые будут использованы для создание следующего уровня блок-схемы.

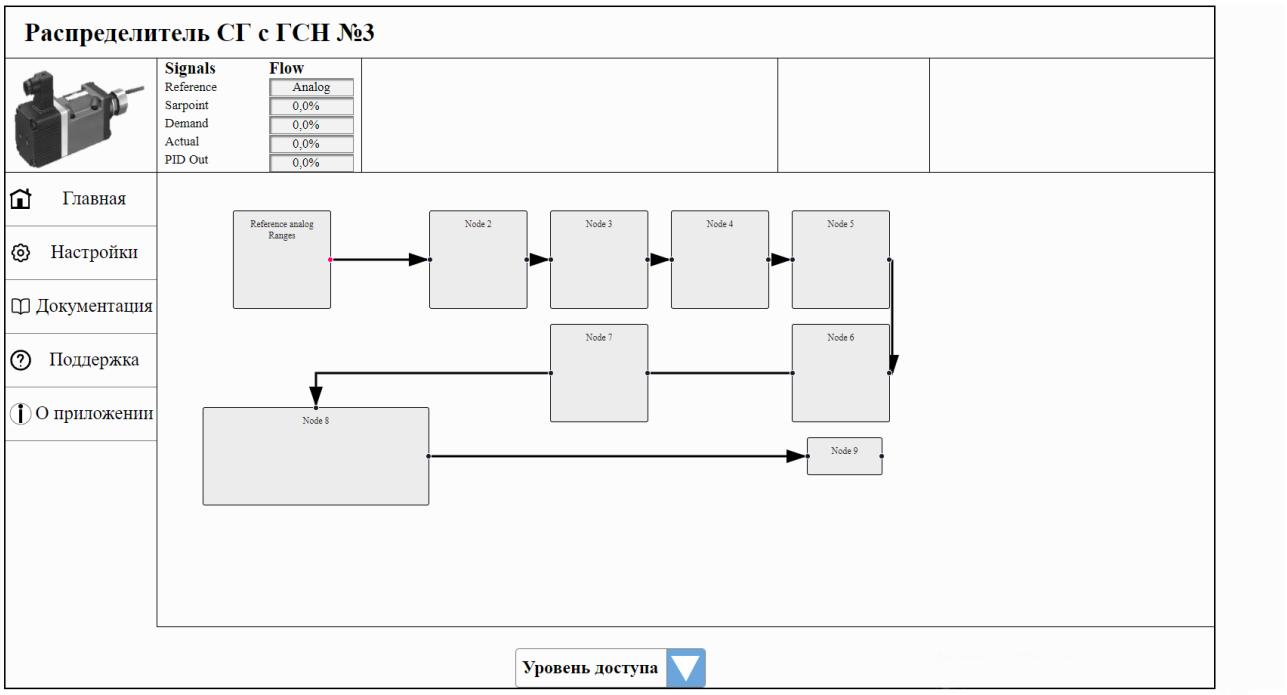

Рисунок 8 — отображение функциональной схемы

3.3. Итоги

Было разработано веб-приложение, в котором можно выбрать нужную конфигурацию и уровень доступа. Разработка приложения продолжается и в дальнейшем будут добавлены графики и компоненты меню, интерфейса по необходимости и запросу заказчика.

4. ТЕСТИРОВАНИЕ

4.1. Проведение тестирования.

Было проведены модульное и функциональное тестирования.

Модульное тестирование - это процесс проверки программного обеспечения, в ходе которого осуществляется тестирование отдельных модулей или компонентов программы. Основная цель модульного тестирования заключается в проверке корректной работы каждого отдельного участка программного кода. Модульные тесты проводятся на этапе создания приложения, чтобы изолировать части кода и убедиться в их работоспособности. В качестве объектов для тестирования могут выступать функции, методы, процедуры, модули или объекты [18].

Модульное тестирование показало, что компоненты программы работают должным образом и выполняют все возложенные на них функции.

В рамках функционального тестирования, то есть тестирования, которое проверяет конкретные функции приложения, которые заложены его логикой работы, работает ли приложение в целом [19], приложение обеспечило выполнение требуемых задач.

4.2. Итоги

Тестирование проводилось локально. Проведённое тестирование не выявило отклонений от требований, предъявляемых к приложению. Приложение выполнило все нужные задачи.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы была выполнена разработка веб-интерфейса сервера мониторинга и конфигурации параметров.

Для достижения этой цели были выполнены следующие задачи:

1. Был проведён аналитический обзор аналогов и основных технологических решений для реализации приложения в ходе которого были выбраны средства для реализации проекта.
2. Определены функциональные и нефункциональные требования к проекту.
3. Спроектирован и реализован веб-интерфейс.
4. Приложение было протестировано на соответствие заявленным требованиям.

Также был оставлен задел для модернизации системы, в дальнейшем возможна реализация следующих идей:

— добавление пользователей в систему авторизации с разным уровнем доступа;

— добавление возможности редактирования параметров в режиме реального времени либо через подтверждение оператором;

— интеграция с реальной базой данных;

— расширение функционала главной страницы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Most used programming languages among developers worldwide as of 2023. [Электронный ресурс] URL: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/ (Дата обращения: 08.05.2024).

2. Ахмедова, Х. Г. Обоснование и разработка требований к программным системам : учебное пособие / Х. Г. Ахмедова. — Москва : РТУ МИРЭА, 2023. — 104 с. — ISBN 978-5-7339-1934-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/382694 (дата обращения: 03.03.2024). — Режим доступа: для авториз. пользователей.

3. Что такое JavaScript? [Электронный ресурс]. URL:https://developer.mozilla.org/ru/docs/Learn/JavaScript/First_steps/What_is_JavaScript (Дата обращения: 30.04.2024).

4. Введение в Angular. [Электронный ресурс]. URL: https://metanit.com/web/angular2/1.1.php (Дата обращения: 07.03.2024).

5. Русскоязычная документация Vue.js [Электронный ресурс]. URL: https://v3.ru.vuejs.org/ru/guide/introduction.html (Дата обращения: 07.03.2024).

6. A RealWorld Comparison of Front-End Frameworks with Benchmarks. [Электронный ресурс]. URL: https://www.freecodecamp.org/news/a-realworld-comparison-of-front-end-frameworks-with-benchmarks-2019-update-4be0d3c78075/ (Дата обращения: 07.03.2024).

7. Что такое PHP? [Электронный ресурс] URL: https://php.ru/manual/intro-whatis.html (Дата обращения: 06.05.2024).

8. Официальный сайт Typescript. [Электронный ресурс] URL: https://www.typescriptlang.org/ (Дата обращения 23.05.2024).

9. Теория тестирования ПО просто и понятно. [Электронный ресурс]. URL: https://habr.com/ru/articles/587620/ (Дата обращения: 08.05.2024).

10. Проектно-производственное предприятие Гидростанок. [Электронный ресурс]. URL: http://www.atos-rf.ru/ (Дата обращения 07.05.2024).

11. What is Vuex? URL: https://vuex.vuejs.org/ [Электронный ресурс]. (Дата обращения: 13.05.2024).

12. Что такое Axios? URL: https://axios-http.com/ru/docs/intro [Электронный ресурс]. (Дата обращения: 13.05.2024).

13. Проектирование и разработка корпоративных web приложений. URL: https://habr.com/ru/articles/249863/ [Электронный ресурс] (Дата обращения 13.05.2024).

14. Справочник Typescript. URL: https://scriptdev.ru/guide/ [Электронный ресурс] (Дата обращения 23.05.2024).

15. Vue Flow Guide. URL: https://vueflow.dev/guide/ [Электронный ресурс]. (Дата обращения 13.05.2024).

16. mxGraph 4.2.2. URL: https://jgraph.github.io/mxgraph/ [Электронный ресурс]. (Дата обращения 13.05.2024).

17. GoJs URL: https://gojs.net/latest/index.html [Электронный ресурс]. (Дата обращения 13.05.2024).

18. Модульное тестирование: что это? Типы, инструменты. URL: https://logrocon.ru/news/unit_testing [Электронный ресурс] (Дата обращения 13.05.2024).

19. Функциональное тестирование ПО: задачи, виды, методы проведения. URL: https://practicum.yandex.ru/blog/funkcionalnoe-testirovanie-po/ [Электронный ресурс] (Дата обращения 13.05.2024).

20. О Ruby. URL: https://www.ruby-lang.org/ru/about/ [Электронный ресурс] (Дата обращения 19.05.2024).

21. Что такое Ruby? URL: https://elbrusboot.camp/blog/chto-takoie-ruby/ [Электронный ресурс] (Дата обращения 19.05.2024).

22. О плюсах и минусах GO. URL: https://habr.com/ru/articles/229169/ [Электронный ресурс] (Дата обращения 19.05.2024).

23. Официальный сайт Go. URL: https://go.dev/ [Электронный ресурс] (Дата обращения 19.05.2024).

24. Go at Google: Language Design in the Service of Software Engineering. URL: https://go.dev/talks/2012/splash.article [Электронный ресурс] (Дата обращения 19.05.2024).

25. Официальный сайт Node.js. URL: https://nodejs.org/en [Электронный ресурс] (Дата обращения 19.05.2024).

26. Введение в Node.js. URL: https://metanit.com/web/nodejs/1.1.php [Электронный ресурс] (Дата обращения 19.05.2024).

27. Pinia Интуитивное хранилище для Vue.js. URL: https://pinia-ru.netlify.app/ [Электронный ресурс] (Дата обращения 20.05.2024).

28. Docker. URL: https://www.docker.com/ [Электронный ресурс] (Дата обращения 20.05.2024).

29. Понимая Docker. URL: https://habr.com/ru/articles/253877/ [Электронный ресурс] (Дата обращения 20.05.2024).

ПРИЛОЖЕНИЕ А

Листинг 1 – фрагмент исходного кода Vue компонента LoginPage.

```
<template>
    <div class="main">
        <div class="container">
            <div class="header">
                <h1 class="header__title">Библиотека оборудования</h1>
            </div>
            <div class="settings">
            <li v-for="item in items" :key="item.id" class="settings__element">
                {{ item.name }}
            </li>
    </div>
    <AccessLevel @open-modal='openModal'></AccessLevel>
    <ModalWindow v-if="modalIsOpen" @close-modal='closeModal'></ModalWindow>
    </div>
    </div>
</template>

<script setup>
  import { ref,onMounted,watch } from 'vue'
  import AccessLevel from '@/components/AccessLevelButton.vue'
  import ModalWindow from '@/components/ModalWindow.vue'
  import { useConfigStore } from '@/stores/useConfigStore.js'

  const store = useConfigStore();
  watch(() => store.getItems, (newValue, oldValue) => {
    items.value = newValue;
  });
  onMounted(() => {
    store.getConfigsFromServer()
  })
  const items = ref([]);;
  items.value = store.getItems;
  const modalIsOpen = ref(false);
  const openModal = () => {
    modalIsOpen.value = true;
  };
  const closeModal = () => {
    modalIsOpen.value = false;
  };
</script>
```

Листинг 2 – фрагмент исходого кода компонента AccessLevel.

```
<template>
    <div class="wrapper" @click="isOpen = !isOpen">
        <h5 class="wrapper__title">Уровень доступа</h5>
             <svg  width="50"  height="50"  viewBox="0  0  50  50"  fill="none"
xmlns="http://www.w3.org/2000/svg"> <path d="M0 0H46C48.2091 0 50 1.79086 50
4V46C50 48.2091 48.2091 50 46 50H0V0Z" fill="#4790D4" fill-opacity="0.8"/> <path
d="M25 43L5 9L43 9L25 43Z" fill="white"/></svg>
        <div class="wrapper__dropdown" v-if="isOpen">
            <a @click="parentOpenModal">Basic</a>
            <a @click="parentOpenModal">Advanced</a>
            <a @click="parentOpenModal">Pro</a>
        </div>
    </div>
</template>

<script>
```

```
    import ModalWindow from '@/components/ModalWindow.vue';
    export default {
        name: 'AccessLevel',
        components:{
            ModalWindow
        },
        data () {
            return {
                isOpen: false,
            }
        },
        methods:{
            parentOpenModal(){
                this.$emit('open-modal');
            }
        }
    }
</script>
```

Листинг 3 – фрагмент исходного кода компонента ModalWindow.vue.

```
<template>
    <div class="wrapper">
        <div class="header">
            <svg @click="$emit('close-modal')" width="27" height="28" viewBox="0
0   27   28"   fill="none"   xmlns="http://www.w3.org/2000/svg"><rect   width="27"
height="28"    fill="white"/><line    x1="1.70711"    y1="1.29289"    x2="25.7071"
y2="25.2929"  stroke="black"  stroke-width="2"/><line  x1="1.29289"  y1="25.2929"
x2="25.2929" y2="1.29289" stroke="black" stroke-width="2"/></svg>
        </div>
        <h1 class="wrapper__title">Введите пароль:</h1>
        <div class="wrapper__form" @keyup.enter="updatePassword">
                            <input    type="password"    v-model="enteredPassword"
@keyup.enter="updatePassword" class="wrapper__input">
            <button v-on:click="updatePassword">Подтвердить</button>
        </div>
    </div>
</template>

<script setup>
    import { ref } from "vue";
    import { useUserPasswordStore } from '@/stores/useUserPasswordStore.js';

    const store = useUserPasswordStore();

    const enteredPassword = ref('');

    const updatePassword = () =>     {
          console.log('Entered Password in modal:', enteredPassword.value); //
Добавлено для отладки
         store.setEnteredPassword(enteredPassword.value); // Принимает пароль из
input и передаёт его в центральное хранилище
        store.sendPasswordToServer(); // вызывает метод отправки данных на сервер
    };
</script>
```

Листинг 4 – исходный код компонента App.vue

```
<template>
    <MainPage  v-if="isAuthenticated"></MainPage>   <!-- Главная страница -->
    <LoginPage v-else></LoginPage> <!-- Начальная страница логина -->
</template>

<script>
  import MainPage from "@/components/MainPage.vue"
  import LoginPage from "@/components/LoginPage.vue"
```

```
  import AccessLevel from "@/components/AccessLevelButton.vue"
  import ModalWindow from '@/components/ModalWindow.vue'
  import { useUserPasswordStore } from '@/stores/useUserPasswordStore.js'
  export default{
    name: 'app',
    components:{
      LoginPage,
      AccessLevel,
      ModalWindow,
      MainPage
    },
    data(){
      return{
        currentPage: 'MainPage'
      };
    },
    computed:{
      isAuthenticated() {
        const store = useUserPasswordStore();
        return store.isAuthenticated;
      },
    },
  }
</script>
```

Листинг 5 – исходный код компонента MainPage.vue.

```
<template>
    <div class="main">
        <div class="container">
            <div class="header">
              <h1 class="header__title">Распределитель СГ с ГСН №3</h1>
            </div>
            <div class="settings">
              <img class="settings__image" src="@/images/image_01.png" alt="">
              <div class="mainsettings">
                <div class="mainsettings__signals">
                  <div class="mainsettings__header">Signals</div>
                  <div class="signals__element">Reference</div>
                  <div class="signals__element">Sarpoint</div>
                  <div class="signals__element">Demand</div>
                  <div class="signals__element">Actual</div>
                  <div class="signals__element">PID Out</div>
                </div>
                <div class="mainsettings__flow">
                  <div class="mainsettings__header">Flow</div>
                  <div class="flow__element">Analog</div>
                  <div class="flow__element">0,0%</div>
                  <div class="flow__element">0,0%</div>
                  <div class="flow__element">0,0%</div>
                  <div class="flow__element">0,0%</div>
                </div>
              </div>
              <div class="settings__status">
                <h2></h2>
              </div>
            </div>
            <div class="leftpanel">
              <li class = 'leftpanel__element'>
                  <svg  class="leftpanel__svg" "><!-- svg вырезан т.к. слишком
большой --!>svg>
                <div class="leftpanel__text">Главная</div>
              </li>
              <li class = 'leftpanel__element'>
```

```
                <svg class="leftpanel__svg"><!-- svg вырезан т.к. слишком большой
--!></svg>
                <div class="leftpanel__text">Настройки</div>
              </li>
              <li class = 'leftpanel__element'>
                <svg class="leftpanel__svg"><!-- svg вырезан т.к. слишком большой
--!><svg>
                <div class="leftpanel__text">Документация</div>
              </li>
              <li class = 'leftpanel__element'>
                  <svg class="leftpanel__svg"> "><!-- svg вырезан т.к. слишком
большой --!></svg>
                <div class="leftpanel__text">Поддержка</div>
              </li>
              <li class = 'leftpanel__element'>
               <svg class="leftpanel__svg" "><!-- svg вырезан т.к. слишком большой
--!>svg>
                <div class="leftpanel__text">О приложении</div>
              </li>
            </div>
            <div class="container__VueFlow">
              <div class="container__buttons">

              </div>
              <VueFlow v-model="elements" v-bind="flowSettings">
                <template #edge-custom="props">
                  <VueFlow__arrowhead v-bind="props" />

                </template>
              </VueFlow>
            </div>
                          <AccessLevel    class="AccessLevelButton"    @open-modal
='openModal'></AccessLevel>
                           <ModalWindow    v-if="modalIsOpen"    @close-modal    =
'closeModal'></ModalWindow>
        </div>
    </div>
</template>

<script setup>
  import { ref,watch } from "vue";
  import {useGraphStore} from '@/stores/useGraphStore.js';
  import AccessLevel from "@/components/AccessLevelButton.vue"
  import ModalWindow from '@/components/ModalWindow.vue';
  import { VueFlow, useVueFlow, Position, MarkerType } from '@vue-flow/core'
                  import                  VueFlow__arrowhead                  from
'@/components/VueFlowComponents/CustomConnectionLine.vue'
  import '@vue-flow/core/dist/style.css';
   import '@vue-flow/core/dist/theme-default.css'; /* import the default theme,
this is optional but generally recommended */

    const store = useGraphStore();
    const modalIsOpen = ref(false);
    const openModal = () => {
          modalIsOpen.value = true;
    };
    const closeModal = () =>{
          modalIsOpen.value = false;
    };
    const flowSettings = {
        nodesDraggable: false, /* Отвечает за возможность перетаскивать ноды,
принимает boolean значение true/false */
         nodesConnectable: false, /* Отвечает за возможность интерактивного
соединения нодов, принимает boolean значение true/false */
```

```
      zoomOnScroll: false, /* Отвечает за возможность приближать поле на колёсико мыши, принимает boolean значение true/false */
       zoomOnDoubleClick: false,/* Отвечает за возможность приближать поле на двойное нажатие ЛКМ, принимает boolean значение true/false */
      zoomOnPinch: false,
      panOnScroll: false,
      panOnScrollMode: false,
      panOnDrag: false
    };
    const {
      onNodeDragStart,
      onNodeDrag,
      onNodeDragStop,
      onNodeClick,
      onNodeDoubleClick,
      onNodeContextMenu,
      onNodeMouseEnter,
      onNodeMouseLeave,
      onNodeMouseMove
    } = useVueFlow()
    onNodeDoubleClick((event) => {
      console.log('Node clicked', event)
      const elementId = event.node.id;
      store.getArrayFromServer(elementId);
    });
    const elements = ref([]);
    elements.value=store.getArray;
    watch(() => store.getArray, (newArray) => {
      elements.value = newArray;
    });
</script>
```

Приложение Б

Листинг 1 – исходный код контроллера HomeController.css

```
using Microsoft.AspNetCore.Mvc;

namespace VueApp1.Server.Controllers
{
    [Route("api/password")]
    public class HomeController : Controller
    {
        [HttpPost]
        [Route("check")]
        public IActionResult UpdateContent(string password, string selectedField)
        {
            if (password == "admin")
            {
                return Ok(new {sucess = true}); // Возвращаем успех

            }
            else
            {
                return BadRequest(new { error = "Неверный пароль" }); // Возвращаем ошибку
            }
        }
    }
}
```

Листинг 2 – исходный код контроллера ConfigsController.cs

```
using Microsoft.AspNetCore.Mvc;
using static VueApp1.Server.Controllers.ConfigsController;

namespace VueApp1.Server.Controllers
{
    [Route("api/configs")]
    public class ConfigsController : Controller
    {
        public class ConfigItem
        {
            public int id { get; set; }
            public string name { get; set; }
        }
        [HttpGet]
        public IActionResult GetConfigs()
        {
            ConfigItem[] configItems = new ConfigItem[]
            {
                new ConfigItem { id = 1, name = "Следящий привод с гидростатической
направляющей #1" },
                new ConfigItem { id = 2, name = "СГ с ГСН #2" },
                new ConfigItem { id = 3, name = "СГ с ГСН #3" },
                new ConfigItem { id = 4, name = "СГ с ГСН #4" },
                new ConfigItem { id = 5, name = "СГ с ГСН #5" },
                new ConfigItem { id = 6, name = "СГ с ГСН #6" },
                new ConfigItem { id = 7, name = "СГ с ГСН #7" },
                new ConfigItem { id = 8, name = "СГ с ГСН #8" },
                new ConfigItem { id = 9, name = "СГ с ГСН #9" }
            };
            return Json(configItems);
        }
    }
}
```

Листинг 3 – исходный код контроллера GraphController.cs.

```
using Microsoft.AspNetCore.Mvc;
using System.Reflection.Emit;


namespace VueApp1.Server.Controllers
{
    [Route("api/graph")]
    public class GraphController : Controller
    {
        public class Position
        {
            public int x { get; set; }
            public int y { get; set; }
        }
        public class Style
        {
            public string backgroundColor { get; set; }
            public string border {  get; set; }
            public string width { get; set; }
            public string height { get; set; }
        }
        public class NodeClass
        {
            public string id { get; set; }
            public string type { get; set; }
            public string targetPosition { get; set; }
            public string sourcePosition { get; set; }
            public string label { get; set; }
            public Position position { get; set; }
```

```
            public Style style { get; set; }
            public NodeClass()
            {
                position = new Position(); // Инициализация объекта position
                style = new Style(); // Инициализация объекта style
            }
        }

        [HttpGet]
        [Route("graphdata")]
        public IActionResult GetGraphData(int nodeId)
        {
            if (nodeId == 1)
            {
                NodeClass[] nodeArray = new NodeClass[]
                {
                    new NodeClass { id = "1", type = "output", targetPosition =
"Right", label = "Reference analog Ranges", position = { x = 100, y = 50 }, style
= { backgroundColor = "#ECECEC", border = "1px solid black", width = "130px",
height = "130px" }},
                    new NodeClass { id = "2", type = "default", targetPosition =
"Left", sourcePosition = "Right", label = "Node 2", position = { x = 360, y = 50
}, style = { backgroundColor = "#ECECEC", border = "1px solid black", width =
"130px", height = "130px" }},

                 };
                return Json(nodeArray);
            }
            if (nodeId == 2)
            {
                NodeClass[] nodeArray = new NodeClass[]
                {
                    new NodeClass { id = "1", type = "output", targetPosition =
"Right", label = "Reference analog Ranges", position = { x = 100, y = 50 }, style
= { backgroundColor = "#ECECEC", border = "1px solid black", width = "130px",
height = "130px" }},
                    new NodeClass { id = "2", type = "default", targetPosition =
"Left", sourcePosition = "Right", label = "Node 2", position = { x = 360, y = 50
}, style = { backgroundColor = "#ECECEC", border = "1px solid black", width =
"130px", height = "130px" }}
                 };
                return Json(nodeArray);
            }
            else
            {
                return NotFound();
            }
        }
    }
}
```

Приложение В

Листинг 1 – исходный код экземпляра централизованного хранилища Pinia useUserPasswordStore.js.

```
import { defineStore } from 'pinia';
import axios from "axios";
export const useUserPasswordStore = defineStore({
  id: 'password',
  state: () => ({
    enteredPassword: '', // Введённый  пароль для отправки на сервер
    isAuthenticated: true
  }),
  actions: {
     setEnteredPassword(password) { //Метод принимающий пароль из поля ввода и
устанавливающий его хранение в компоненте
        this.enteredPassword = password;
        console.log('Entered Password:', this.enteredPassword); // Добавлено для
отладки
    },
    async sendPasswordToServer() {
        console.log('Recieving data from server:', this.enteredPassword); //
Добавлено для отладки
           axios.post('http://localhost:3000/api/password/check', { password:
this.enteredPassword } ) // Посылает пароль на сервер
          .then(response => {
             if (response.data.success) {
                  this.isAuthenticated = true; // Теперь доступен контент для
авторизованных пользователей
             }
          })
          .catch(error => {
            console.error(error); // Возврат ошибки в случае неправильного пароля
          });
    },
  },
});
```

Листинг 2 – исходный код экземпляра централизованного хранилища Pinia useGraphStore.js

```
import { defineStore } from 'pinia';
import axios from "axios";
import { VueFlow, Position, MarkerType } from '@vue-flow/core';
export const useGraphStore = defineStore({
  id: 'graphStore',
  state: () => ({
    graphArray: [
        // nodes
    { id: '1', type: 'output', targetPosition: Position.Right, label: 'Reference
analog Ranges', position: { x: 100, y: 50 }, style: { backgroundColor:
'#ECECEC', border: '1px solid black', width: '130px', height: '130px' }, },
    { id: '2', type: 'default', targetPosition: Position.Left, sourcePosition:
Position.Right, label: 'Node 2', position: { x: 360, y: 50 }, style: {
backgroundColor: '#ECECEC', border: '1px solid black', width: '130px', height:
'130px' }, },
    { id: '3', label: 'Node 3', type: 'default', targetPosition: Position.Left,
sourcePosition: Position.Right,  position: { x: 520, y: 50 }, style: {
```

```
backgroundColor: '#ECECEC',  border: '1px solid black', width: '130px', height:
'130px' }, },
    { id: '4', label: 'Node 4', type: 'default',  targetPosition: Position.Left,
sourcePosition: Position.Right, position: { x: 680, y: 50 }, style: {
backgroundColor: '#ECECEC', border: '1px solid black', width: '130px', height:
'130px' }, },
    { id: '5', label: 'Node 5', type: 'default', targetPosition: Position.Left,
sourcePosition: Position.Right,  position: { x: 840, y: 50 }, style: {
backgroundColor: '#ECECEC',  border: '1px solid black', width: '130px', height:
'130px' }, },
    { id: '6', label: 'Node 6', type: 'default', targetPosition: Position.Right,
sourcePosition: Position.Left,  position: { x: 840, y: 200 }, style: {
backgroundColor: '#ECECEC',  border: '1px solid black', width: '130px', height:
'130px' }, },
    { id: '7', label: 'Node 7', type: 'default', targetPosition: Position.Right,
sourcePosition: Position.Left,  position: { x: 520, y: 200 }, style: {
backgroundColor: '#ECECEC',  border: '1px solid black', width: '130px', height:
'130px' }, },
    { id: '8', label: 'Node 8', type: 'default', targetPosition: Position.Top,
targetPosition: Position.Top, sourcePosition: Position.Right,  position: { x:
60, y: 310 }, style: { backgroundColor: '#ECECEC',  border: '1px solid black',
width: '300px', height: '130px' }, },
    { id: '9', label: 'Node 9', type: 'default', targetPosition: Position.Left,
sourcePosition: Position.Right,  position: { x: 860, y: 350 }, style: {
backgroundColor: '#ECECEC',  border: '1px solid black', width: '100px', height:
'50px' }, },

      // edges
      { id: 'e1-2', type: 'custom',  source: '1', target: '2' },
      { id: 'e2-3', type: 'custom',  source: '2', target: '3'},
      { id: 'e3-4', type: 'custom',  source: '3', target: '4'},
      { id: 'e4-5', type: 'custom',  source: '4', target: '5'},
      { id: 'e5-6', type: 'custom',  source: '5', target: '6'},
      { id: 'e6-7', type: 'custom',  source: '6', target: '7'},
      { id: 'e7-8', type: 'custom',  source: '7', target: '8'},
      { id: 'e8-9', type: 'custom',  source: '8', target: '9'},],
  }),
  getters: {
    getArray(state){
        return state.graphArray;
      },
  },
  actions:{
    async getArrayFromServer(id) {
      console.log('Rescieving data from server:', this.id); // Добавлено для
отладки
      axios.get(`http://localhost:3000/api/graph/graphdata?nodeId=${id}`)
        .then(response => {
          this.graphArray = response.data;          // Обработка полученного
массива данных
```

```
          console.log(this.graphArray);
        })
        .catch(error => {
          console.error('Ошибка при получении данных:', error);        // Обработка
ошибок
        });
    },
  }
});
```

Листинг 3 – исходный код экземпляра централизованного хранилища Pinia useConfig.store.js

```
import { defineStore } from 'pinia';
import axios from 'axios';
export const useConfigStore = defineStore({
  id: 'ConfigStore',
  state: () => ({
    items:[]
  }),
  getters: {
    getItems(state){
        return state.items;
      },
  },
  actions: {
    async getConfigsFromServer(){
        console.log('Recieving configs from server');
        axios.get('http://localhost:3000/api/configs')
          .then(response => {
            this.items = response.data;          // Обработка полученного
массива данных
            console.log(this.items);
          })
          .catch(error => {
            console.error('Ошибка при получении данных:', error);        //
Обработка ошибок
          });
    }
  },
});
```