МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий
Кафедра Систем информатики

Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Компьютерные науки и системотехника

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА

Герасимовой Александры Павловны

Тема работы:

РАЗРАБОТКА АВТОМАТИЗИРОВАННОГО ИНТЕРФЕЙСА УПРАВЛЕНИЯ ОБОРУДОВАНИЯ СТАНЦИИ 1-3 «БЫСТРОПРОТЕКАЮЩИЕ ПРОЦЕССЫ» ЦКП «СКИФ»

«К защите допущена»
Заведующий кафедрой,
д.ф.-м.н., профессор
Лаврентьев М.М./……
(ФИО) / (подпись)
«……»……20…г.

Руководитель ВКР
д. ф.-м. н.,
доцент КафММГФ ММФ НГУ
Платов Г.А./……...
(ФИО) / (подпись)
«……»……20…г.

Соруководитель ВКР
б/с,
старший преподаватель КафОФ НГУ
Максимов Д.А./……..
(ФИО) / (подпись)
«…»……20…г.

Новосибирск, 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
(НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)
Факультет информационных технологий

Кафедра систем информатики
(название кафедры)

Направление подготовки 09.03.01 Информатика и вычислительная техника
Направленность (профиль): Компьютерные науки и системотехника

УТВЕРЖДАЮ

Зав. кафедрой……
(фамилия, И., О.)

…….
(подпись)
«……»……20…г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

Студентке Герасимовой Александре Павловне, группы 20214
(фамилия, имя, отчество, номер группы)

Тема Разработка автоматизированного интерфейса управления оборудованием станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ»
(полное название темы выпускной квалификационной работы)

утверждена распоряжением проректора по учебной работе от «23» октября 2023 г. №0377

Срок сдачи студентом готовой работы «31» мая 2024 г.

Исходные данные (или цель работы) Разработать автоматизированный интерфейс управления оборудованием станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ»

Структурные части работы: 1. Введение; 2. Обзор литературы; 3. Методология; 4. Разработка автоматизированного интерфейса; 5. Тестирование и оценка; 6. Заключение; 7. Список литературы; 8. Приложения.

Руководитель ВКР: Платов Геннадий Алексеевич
Доцент кафедры математических методов геофизики ММФ НГУ, д. ф.-м. н.

……/……..
(ФИО) / (подпись)

«…»……20…г.

Задание принял к исполнению

……..……/……..
(ФИО студента) / (подпись)

«…»……20…г.

Соруководитель ВКР: Максимов Дмитрий Александрович, старший преподаватель кафедры общей физики ФФ НГУ

……/……..
(ФИО) / (подпись)

«…»……20…г.

СОДЕРЖАНИЕ

Введение....4
1 Требования к автоматизированному интерфейсу.... 7
2 Архитектура автоматизированной системы управления....9
3 Обзор платформ создания систем управления.... 12
4 Инструментарий для разработки и архитектура приложения....15
5 Разработка пользовательского интерфейса....18
5.1 Элементы пользовательского интерфейса....18
5.2 Структура проекта....18
5.3 Компоненты ui....19
5.3.1. Компонент Table....19
5.3.2 Компонент Button....20
5.3.3 Компонент Input....20
5.3.4 Компонент Modal....21
5.4 Взаимодействие пользовательского интерфейса с сервером....22
5.5 Конфигурационные файлы....22
5.6 Создание интерфейса для конкретной системы оборудования....23
6 Разработка сервера....26
6.1 Основной файл....27
6.2 Конфигурационные файлы....27
6.3 Модуль контроллеров....27
6.4 Модуль сервисов....28
6.5 Модуль клиентов....29
6.6 Модуль маршрутизаторов....29
6.7 Модуль обработки ошибок....29
6.8 Настройка сервера для конкретного блока оборудования станции....30
7 Тестирование разработанного автоматизированного интерфейса.... 31
Заключение.... 32
Список использованных источников и литературы....34
Приложение А....36
Приложение Б....46

ВВЕДЕНИЕ

Центр коллективного пользования "Сибирский кольцевой источник фотонов" (ЦКП «СКИФ») является комплексом зданий, сооружений, инженерного и технологического оборудования, обеспечивающим выполнение научных исследований на пучках синхротронного излучения. СКИФ — это источник синхротронного излучения поколения 4+, который создаётся в Наукограде Кольцово под Новосибирском. Синхротронное излучение (СИ) является эффективным и универсальным инструментом для широкого спектра исследований и для технологических применений во множестве областей науки и техники [1]. Источники синхротронного излучения предоставляют возможности характеризовать структурные особенности новых функциональных материалов [2], проводить вирусологические исследования [3] и решать множество других актуальных в настоящее время научных задач, для которых требуются исследования с использованием пучков СИ.

Станция 1-3 «Быстропротекающие процессы» — одна из экспериментальных станций первой очереди ЦКП “СКИФ”. Станция состоит из двух секций: «Динамические процессы» и «Плазма». Секция «Динамические процессы» позволит проводить исследования процессов, происходящих в условиях взрыва и импульсных ударных нагрузок, ее особенностью является изучение быстрых процессов при скоростях до 20 км/с и временем экспозиции порядка 100 пс и менее. Работа секции «Плазма» направлена на исследование воздействия на материалы мощного импульсного нагрева, приводящего к деформации, возникновению механических напряжений и механическому разрушению материалов. Научные задачи этих секций соответствуют Стратегии научно-технологического развития Российской Федерации. Разработка станции 1-3 «Быстропротекающие процессы» позволит укрепить лидирующие позиции РФ в области исследования быстропротекающих процессов с использованием синхротронного излучения [4].

Помимо разработки оборудования станции, требуется создание

автоматизированной системы управления оборудованием. Автоматизированная система управления технологическими процессами (АСУ ТП) [5] — это человеко-машинная система управления, представляющая собой совокупность аппаратно-программных средств, осуществляющее управление и контроль технологическими процессами. В функции АСУ ТП станции входит автоматизация управления и настройки оборудования. Разработка системы управления сейчас ведётся в Институте автоматики и электрометрии СО РАН.

Для обеспечения эффективного взаимодействия оператора станции с оборудованием в состав АСУ ТП должны входить пользовательские интерфейсы, обеспечивающие управление элементами оборудования и предоставление информации о них.

Автоматизированный интерфейс — это современный человеко-машинный интерфейс, который позволяет операторам дистанционно управлять оборудованием и системами автоматизации. Он использует качественные графические интерфейсы для легкой интерпретации данных об оборудовании, поддерживает обработку данных в реальном времени и гибко адаптируется к изменяющимся требованиям и интеграции нового оборудования [6]. Интеграция автоматизированного интерфейса в АСУ ТП предоставляет операторам возможность эффективной и удобной работы с оборудованием.

Целью данной работы является предоставление работникам станции 1-3 «Быстропротекающие процессы» удобного и эффективного способа взаимодействия с оборудованием.

Основной задачей работы является разработка автоматизированного интерфейса управления оборудованием станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ».

В соответствие с поставленным целью и задачей были определены основные этапы работы:

1. определить требования к разрабатываемому автоматизированному интерфейсу;

2. определить архитектуру автоматизированной системы управления;
3. провести обзор существующих решений;
4. выбрать инструментарий для разработки;
5. разработать автоматизированный интерфейс;
6. провести тестирование разработанного автоматизированного интерфейса.

В результате работы реализован автоматизированный интерфейс управления несколькими системами оборудования станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ». Архитектура кода автоматизированного интерфейса предоставляет возможность настройки и изменения его под требования к другим системам оборудования станции.

1 Требования к автоматизированному интерфейсу

Одним из требований к автоматизированному интерфейсу управления станции является возможность удалённого взаимодействия с оборудованием, а также минимизация подготовки рабочего места для запуска управляющего интерфейса. Исходя из данных требований наиболее подходящим вариантом реализации автоматизированного интерфейса является веб-приложение.

Веб-приложение [7] — клиент-серверное приложение, клиентская часть которого выполняется в веб-браузере пользователя.

Применение веб-приложения для управления оборудованием имеет ряд преимуществ перед использованием десктопных приложений:

- пользовательский интерфейс может быть запущен на устройстве с любой операционной системой;
- на устройстве оператора требуется только наличие веб-браузера, не требуется установка отдельного приложения;
- обновления и изменения в пользовательском интерфейсе могут быть внесены централизованно на сервере, что устраняет необходимость обновления приложения на каждом устройстве оператора.

Экспериментальное оборудование на станции состоит из различных систем, ответственных за выполнение определенных функции. Каждый блок управляется своим программируемым логическим контроллером. Программируемый логический контроллер [8] — это относительно небольшой компьютер, управляемый микропроцессором с собственной операционной системой, выполняющий функции управления физическими процессами в соответствии с заложенным алгоритмом, с использованием информации, получаемой от датчиков и выводимой на окончательные устройства. Для каждой системы оборудования станции или группы систем требуется свой пользовательский интерфейс, так как управление системами происходит независимо друг от друга. Параметры оборудования, с которыми оператору требуется работать, у различных систем могут различаться. Основываясь на

этом, требованием к разрабатываемому автоматизированному интерфейсу является возможность быстрой и простой реализации его под управление определенной системы или группы систем оборудования.

Система оборудования станции включает элементы физического управления системой и (или) элементы для измерения. Элементом управления может быть механическая подвижка, которая является устройством, осуществляющим физическое перемещение в определенном направлении плоскости. Элементами для измерения являются датчики. Датчики — это устройства, измеряющие физические величины: положение, температуру, давление и т.д.

Функциональными требованиями к интерфейсу являются:

- Отображение параметров элементов системы экспериментального оборудования. Примером параметра механической подвижки является значение координаты положения подвижки, а для датчика — измеренные им значения температуры.
- Отображение значений, которые находятся в регистрах памяти микроконтроллера элемента системы.
- Отправка команд для изменения параметров элементов управления.
- Отправка команд для изменения значений, находящихся в регистрах памяти микроконтроллеров элементов системы.
- Отображения статуса каждого элемента системы.
- Отображение статуса работы системы управления.
- Отображение допустимости отправляемых пользователем команд.

2 Архитектура автоматизированной системы управления

Компоненты системы управления по своим функциональным признакам можно разделить на три категории:

1. Компоненты, обеспечивающие взаимодействие с конечным пользователем.
2. Компоненты, работающие с оборудованием. Компоненты из этой категории обеспечивают непосредственное исполнение команд, полученных от пользователя.
3. Служебные компоненты. Компоненты из этой категории обеспечивают связь между компонентами, обеспечивающих взаимодействие с конечным пользователем и компонентами, работающими с оборудованием.

В автоматизированной системе управления оборудованием станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ» в состав служебных компонентов входят сервер веб-приложения и брокер MQTT.

MQTT (Message Queuing Telemetry Transport) [9] — это протокол, работающий поверх протокола TCP/IP, обмена сообщениями. Протокол построен по принципу «издатель-подписчик».

Схема обмена сообщениями состоит из трёх элементов:

1. издатель — устройство, которое отправляет сообщения;
2. подписчик — устройство, которое получает и обрабатывает сообщения;
3. шлюз или брокер — устройство, обеспечивающее логистику сообщений от издателей к подписчикам, хранение сообщений и ответственное за аутентификацию и управление доступом.

Сообщения имеют тему, на которую устройство может подписываться для получения определенных сообщений. Публикация сообщений происходит с указанием темы, что позволяет направлять их к соответствующим подписчикам.

С помощью протокола MQTT реализуется взаимодействие контроллеров с сервером веб-приложения через брокер MQTT. Контроллеры и сервер являются одновременно и издателями, и подписчиками. Сервер же взаимодействует с

клиентской части. Это означает, что брокер MQTT и сервер, обеспечивают передачу данных между контроллерами систем и конечным пользователем.

Система управления оборудованием состоит из следующих компонентов:

1. Клиентская часть — пользовательский интерфейс.
2. Сервер. Выполняет функции посредника между клиентской частью и брокером.
3. Брокер — центральный компонент для обмена сообщениями между контроллерами и сервером.
4. Контроллеры — устройства, взаимодействующие с драйверами для управления физическими устройствами.
5. Драйверы — аппаратные устройства, принимающие команды от контроллеров и непосредственно взаимодействующие с исполнительными механизмами и датчиками. Исполнительные механизмы — это двигатели механических подвижек, насосы, клапана и другое оборудование.
6. Redis (remote dictionary server) [10] — база данных класса NoSQL, использующаяся как в качестве баз данных, так и для реализации кэшей. Она ориентирована на достижение максимальной производительности при выполнении атомарных операций: set и get. В системе управления является хранилищем данных, используемым для обмена информацией между программами внутри контроллера.

Общая схема архитектуры автоматизированной системы управления оборудованием станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ» представлена на рисунке 1.

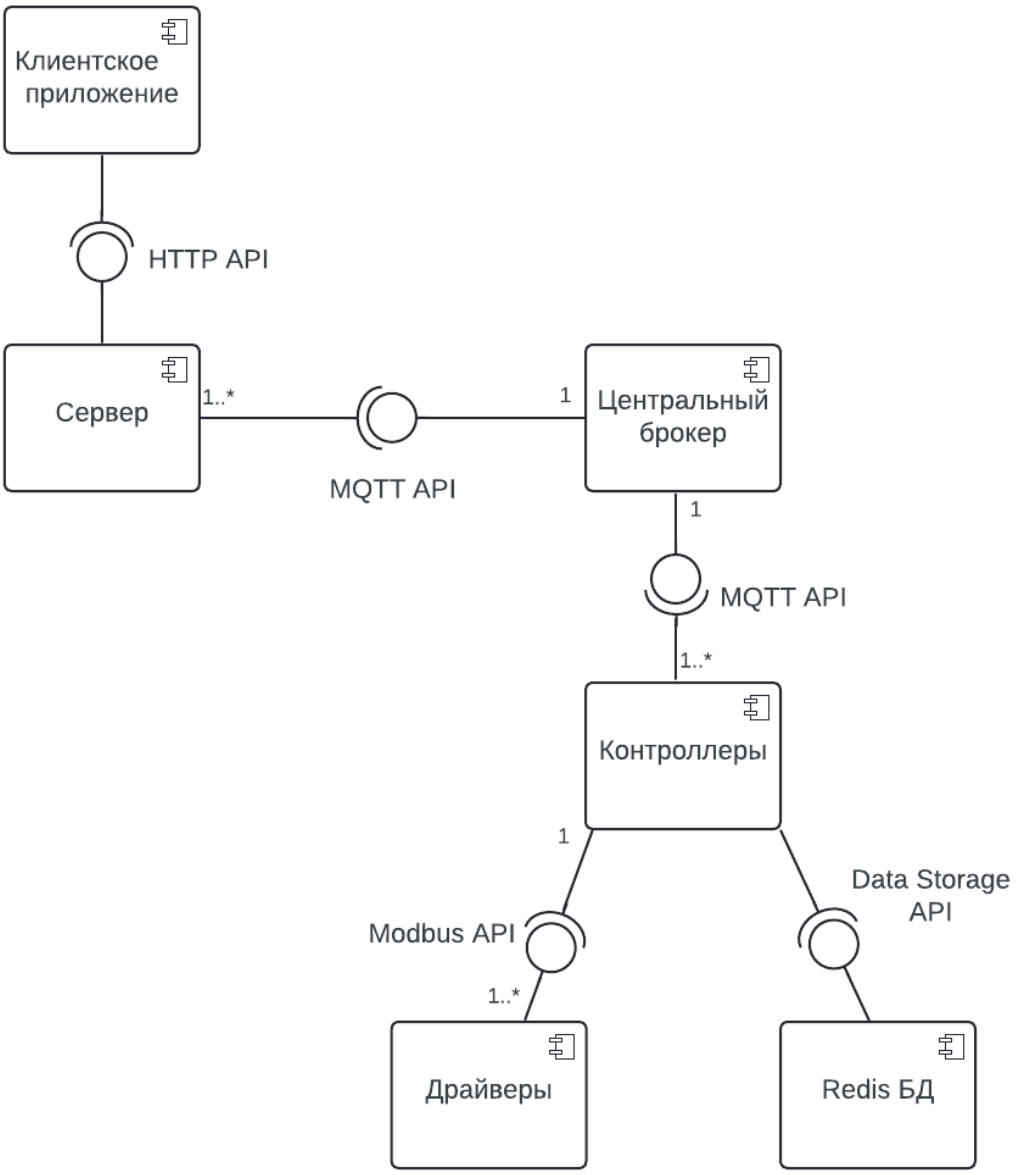

Рисунок 1 — Диаграмма компонентов системы управления

Задачей моей работы является реализация клиентской части и сервера.

3 Обзор платформ создания систем управления

Классическая автоматизированная система управления состоит из трех уровней: [5]

1. Нижний уровень. Контрольно-измерительные приборы и автоматика.
2. Средний уровень. Программируемые логические контроллеры.
3. Верхний уровень. SCADA (Supervisory Control and Data Acquisition) — программный пакет, необходимый для обеспечения или разработки новой системы работы в реальном времени.

SCADA предоставляет пользовательский интерфейс управления, а также взаимодействие между пользовательским интерфейсом и ПЛК. Разрабатываемый автоматизированный интерфейс управления также предоставляет пользовательский интерфейс управления и обеспечивает взаимодействие пользовательского интерфейса с ПЛК через брокер MQTT.

Принципиальным различием SCADA от разрабатываемого автоматизированного интерфейса является процесс создания пользовательского интерфейса для управления оборудованием. В автоматизированном интерфейсе разработчику необходимо вносить изменения в код клиентской части, чтобы настроить его под конкретные требования системы станции. Несмотря на то, что благодаря гибкости кода этот процесс несложен и выполняется достаточно быстро, такой подход не является доступным для широкого круга пользователей и, следовательно, не может считаться платформой для создания пользовательского интерфейса.

В работе рассмотрено несколько популярных SCADA.

Ignition от Inductive Automation является платформой для промышленной автоматизации, которая с 2010 года используется в тысячах мест в более чем 100 странах [11]. Платформа позволяет создавать интуитивно-понятные веб-интерфейсы, обладающие функциональностью, требующейся для управления оборудованием и мониторинга, а также предоставляет взаимодействие операторского интерфейса с ПЛК с помощью OPC-протоколов

[11, 12]. Интеграция данного автоматизированного интерфейса в систему управления является платной.

VTScada предоставляет интуитивно понятную платформу для разработки настраиваемых приложений промышленного мониторинга и управления. Эти приложения характеризуются высокой степенью надежности и простотой использования для конечных пользователей. Множество организаций по всему миру использует VTScada для выполнения задач любой сложности, обеспечивая стабильное и эффективное управление производственными процессами [13]. VTScada является платной платформой.

Rapid SCADA — это платформа для промышленной автоматизации с открытым исходным кодом. Она позволяет быстро создавать системы мониторинга и управления, а открытый исходный код предоставляет прозрачность и безопасность работы программного обеспечения [14]. Однако платформа имеет ограничения в создании пользовательских интерфейсов, что затрудняет создание интуитивно-понятных и настраиваемых интерфейсов для операторов. Для создания более сложных и интерактивных интерфейсов может потребоваться использование сторонних инструментов.

Рассмотренные платформы предоставляют возможности для создания качественных приложений для управления и мониторинга, но они также имеют свои недостатки. Ignition и VTScada предоставляют множество функций, многие из которых могут быть избыточными для задач оператора, что усложняет эксплуатацию и делает покупку лицензии на использование платформы неоправданно дорогой. Rapid SCADA является бесплатной, но имеет ограничения в создании пользовательских интерфейсов. Разработка собственного решения позволяет интегрировать только необходимые функции и элементы, требующиеся для создания пользовательских интерфейсов управления оборудованием станции 1-3 «Быстропротекающие процессы», обеспечивает гибкость в адаптации и масштабировании, исключая недостатки сторонней платформы.

Одним из требованиев к автоматизированной системе управления оборудованием станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ» является создание собственного решения, реализующего компоненты, обеспечивающие взаимодействие с конечным пользователем, и связь компонентов пользователя с ПЛК. В связи с этим собственная реализация автоматизированного интерфейса необходима для предоставления системы управления на станцию. Использование готовой платформы для разработки верхнего уровня автоматизированной системы управления в данном случае не является возможным.

4 Инструментарий для разработки и архитектура приложения

Наиболее распространенными видами архитектуры веб-приложений являются: [15]

- Многостраничные веб-приложения (MPA);
- Одностраничные веб-приложения (SPA).

В многостраничных приложениях, когда происходит пользовательское действие, отправляется запрос на сервер, который возвращает клиентской части полностью новый HTML-документ. Такое поведение приложения сильно влияет на скорость и производительность поскольку большая часть данных каждый раз загружается повторно. MPA подходит для случаев, когда приложение отображает большое количество статичной информации.

Одностраничное приложение содержит динамически обновляемый HTML-документ. При действиях пользователя, загружается и обновляется только определенная часть контента пользовательского интерфейса.

В случаях, когда приложение подразумевает частые действия со стороны пользователя и частое обновление данных на интерфейсе, одностраничные приложения получают существенное преимущество в общей скорости работы над многостраничными.

Обновление данных на пользовательском интерфейсе оператора происходит достаточно часто, так как на нем должна отражаться текущая информация об оборудовании. В связи с этим реализация одностраничного веб-приложения является более подходящей для использования его в качестве составляющей системы управления оборудованием.

Существует множество инструментов для создания пользовательских интерфейсов. Одними из самых популярных инструментов создания одностраничных приложений являются React [16], Vue.js и Angular. React обеспечивает высокую производительность за счет использования Virtual DOM и гибкость в выборе подходов к разработке. Разработчики имеют доступ к низкоуровневым механизмам и компонентам, которые управляют

функциональностью и поведением приложения. Это позволяет более детально и точно управлять процессами и оптимизировать производительность. Vue.js выделяется своей простотой изучения. Vue.js также использует Virtual DOM, но отсутствие низкоуровневого доступа не позволяет разработчикам достаточно хорошо контролировать внутренние процессы и оптимизировать их на уровне, доступном в React. Angular предлагает полную инфраструктуру для разработки с множеством встроенных решений и строгой типизацией с использованием TypeScript. Однако Angular требует большой объем памяти, а также сложен в изучении [17].

Для реализации пользовательского интерфейса выбран React. Высокая производительность является ключевой составляющей для интерфейса оператора станции. Она обеспечивает эффективную визуализацию данных и надежность. Модульная структура React позволяет создавать гибкие интерфейсы, что делает быстрым и удобным процесс изменения и настройки пользовательского интерфейса в зависимости от требований к управлению определенной системы оборудования станции.

При использовании React для разработки веб-интерфейса используются языки программирования Javascript, CSS и язык HTML.

Использование одного языка программирования для разработки как клиентской, так и серверной части веб-приложения значительно упрощает процесс разработки и поддержки. Это способствует более легкому переносу данных и логики между клиентской и серверной частями, улучшая консистентность кода и сокращая время разработки. Использование JavaScript для серверной части автоматизированного интерфейса управления оборудованием станции повышает эффективность его разработки.

Node.js — это программная платформа, которая позволяет выполнять JavaScript-код на стороне сервера. Node.js позволяет обрабатывать большое количество запросов с высокой скоростью. Платформа обладает большой библиотекой пакетов NPM (Node Package Manager), которая предоставляет

доступ к модулям и инструментам, облегчающим процесс разработки. Использование Node.js предоставляет возможность быстрой и несложной реализации кода [18].

В работе взаимодействие сервера и пользовательского интерфейса обеспечивается с помощью HTTP [19], упрощенного протокола прикладного уровня, размещающегося поверх TCP. Сообщение HTTP содержит метаданные о сообщении и информацию о запросе и ответе. Тело сообщения для запроса содержит данные, которые могут быть использованы сервером при обработке запроса, тело сообщения для ответа содержит данные, требующиеся для работы клиентской части. Наиболее распространенными методами HTTP запросов являются GET и POST. Метод POST нужен для отправки сущностей к определённому ресурсу, метод GET запрашивает представление ресурса.

Обмен сообщениями между сервером и брокером MQTT осуществляется с использованием протокола MQTT. При публикации сообщения сервером в теме указывается часть оборудования, к которой относится команда управления, а также название самой команды. В содержимом сообщения хранится описание данных, которые использует команда при выполнении. Например, тема может быть "/driver/calibration/4/command", а сообщение "{\"timestamp\":123123123,\ "cmdname\":\"move_rel\",\"cmdvalue\":10}". Сервер подписывается на темы, в которых указываются названия частей оборудования, информация о которых приходит в сообщениях. Это означает, что по каждой конкретной теме сервер получает информацию, относящуюся именно к указанной в теме части оборудования.

.

5 Разработка пользовательского интерфейса

5.1 Элементы пользовательского интерфейса

Для создания удобного и функционального пользовательского интерфейса оператора пользовательский интерфейс разделен на два основных элемента:

1. Основное окно интерфейса. Окно, на котором располагаются элементы, использующиеся оператором основное время при настройке оборудования.
2. Дополнительное окно интерфейса. Окно интерфейса предназначенное для отображения элементов, предоставляющих информацию и средства управления оборудованием, которые используются оператором реже, например, при начальной настройке оборудования или при выполнении специфических задач.

В качестве элемента, предоставляющего информацию о системе оборудования, используется табличная форма представления данных. Строка таблицы отображает инмформацию о статусе, парметрах отдельного элемента системы, а также предоставляет отправку команд для изменения значений параметров элементов. Также таблица используется для работы с параметрами отдельного элемента системы и для работы с регистрами микроконтроллера. Табличный формат позволяет легко воспринимать данные, предоставляя структурированное отображение информации, что повышает общую эффективность работы оператора, упрощает процесс управления и мониторинга.

Отправка команд осуществляется с помощью использования числовых полей ввода и кнопок.

5.2 Структура проекта

В React структура приложения состоит из компонентов, являющихся самостоятельными частями пользовательского интерфейса, которые можно переиспользовать. Компоненты могут включать дочерние компоненты и взаимодействовать с родительскими компонентами. Основным способом

передачи данных между компонентами является использование props. Props передаются в качестве аргументов дочернему компоненту от родительского и являются неизменяемыми, то есть дочерний компонент не может изменить их значение. Props предоставляют возможность создавать динамические компоненты, которые не зависят от конкретных статических данных.

Код проекта разбит на отдельные директории:

1. main. Компоненты, в которых определяется структура основного окна интерфейса.
2. modal. Компоненты, в которых определяется структура дополнительного окна.
3. ui. Компоненты пользовательского интерфейса. Представляет библиотеку компонентов, которые используются для построения окон.
4. data. Конфигурационные файлы.
5. http. Модули для работы с HTTP-запросами.
6. styles. Файлы стилей на языке CSS, в которых описывается внешний вид интерфейса.

5.3 Компоненты ui

Исходя из анализа элементов пользовательского интерфейса, выделено несколько ключевых компонентов, которые могут использоваться при разработке интерфейсов многократно. К таким элементам относятся: таблица, кнопка, поле ввода и дополнительное окно. Реализация данных компонентов находится в ui.

Компоненты из ui были спроектированы с учетом принципов абстракции и независимости, что позволяет использовать их для различных целей в разных частях приложения, независимо от конкретного содержимого или функциональности, которую они должны выполнять.

5.3.1. Компонент Table

Компонент Table служит для отображения данных в табличной форме. Данные, содержащиеся в ячейках таблицы, а также описание структуры

таблицы передаются от родительского компонента. Описание структуры таблицы представляет из себя объект, в котором определены названия столбцов таблицы и тип содержимого для каждой ячейки столбца. Передача информации, которая содержится в таблице, извне позволяет использовать компонент Table в разных частях пользовательского интерфейса, обеспечивая независимость компонента от конкретного описания структуры таблицы.

Компонент реализован с использованием библиотеки react-table.

В стилях таблицы реализованы чередование цветов строк и яркий цвет заголовков для обеспечения улучшенного восприятия информации. Пример отображения Table при использовании его для создания таблицы, описывающей систему калибровки образцов оборудования станции, представлен на рисунке 2.

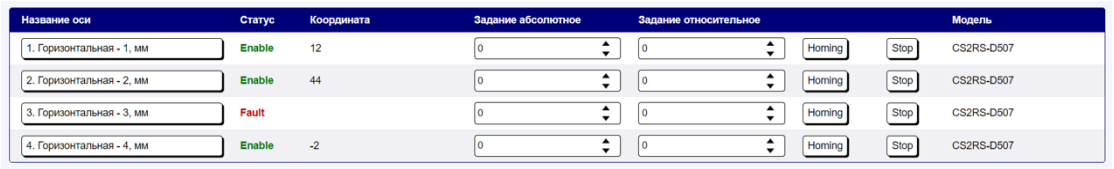

Рисунок 2 — пример использование Table

5.3.2 Компонент Button

Компонент Button является кнопкой. Он принимает от родительского компонента функцию, которая будет вызвана при нажатии кнопки, и текст, который отображается на кнопке. Стили кнопки предоставляют возможность изменения цвета кнопки в зависимости от состояния: кнопка нажата, наведён курсор на кнопку, кнопка не используется. Пример отображения Button при использовании его для создания элементов отправки команд представлен на рисунке 3.

Рисунок 3 — пример использования Button

5.3.3 Компонент Input

Компонент Input нужен для обеспечения возможности пользователю вводить числовые значения. Компонент Input не является базовым элементом

HTML <input> типа “number”, а представляет совокупность текстового поля ввода и двух кнопок для увеличения или уменьшения значения. Использование данного варианта позволяет детально настраивать внешний вид и функциональность компонента. В стилях CSS предусмотрено, чтобы поле ввода и кнопки визуально воспринимались как единый элемент управления.

Компонент Input является числовым полем ввода. Он принимает функцию для обработки ввода пользователя, текущее значение поля ввода и параметр, определяющий наличие кнопок для увеличения или уменьшения значения на единицу при нажатии. Функцией обработки ввода может являться, например, функция, которая отправляет введенное пользователем число на сервер и обновляет данные в таблице.

В компоненте реализована логика проверки ввода, предотвращающая ввод нечисловых значений, что обеспечивает корректность данных. Также в компоненте есть проверка, соответствует ли введенное значение критериям отправки его на сервер. Отправка значения происходит при нажатии пользователем клавиши “Enter” или при изменении значения с помощью кнопок увеличения и уменьшения.

Пример отображения Input при использовании его для создания элементов отправки команд представлен на рисунке 4.

Рисунок 4 — пример использования Input

5.3.4 Компонент Modal

Компонент Modal представляет собой окно, которое отображается при взаимодействии пользователя с основным интерфейсом, например, через нажатие кнопки, ответственной за открытие окна. Компонент принимает параметр, определяющий его видимость, а также компоненты, которые должны быть отображены в этом окне. Такая архитектура обеспечивает независимость компонента от способов управления его видимостью и от содержимого окна.

Пример отображения Modal при использовании его для создания дополнительного окна для системы калибровки образцов представлен на рисунке 5.

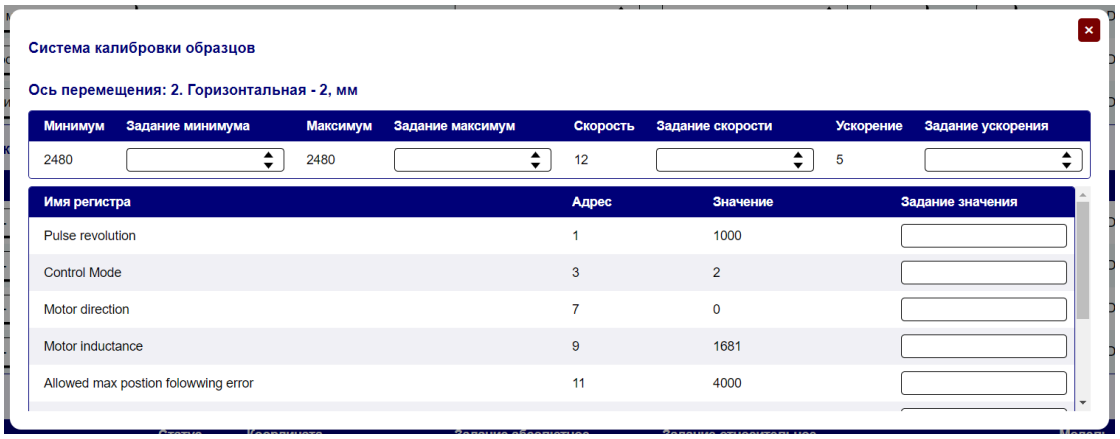

Рисунок 5 — пример использования Modal

5.4 Взаимодействие пользовательского интерфейса с сервером

Модуль, предназначенный для обеспечения взаимодействия пользовательского интерфейса с серверной частью приложения через HTTP-запросы, находится в http. В нем указывается адрес сервера, реализованы функции, которые обеспечивают отправку команд для изменения параметров элементов системы и отправку команд для изменения значений регистров на сервер и получение данных, содержащих информацию об оборудовании и статуса связи пользовательского интерфейса с брокером MQTT. Для получения данных используется метод GET, для отправки команд используется метод POST. Для реализации запросов используется стандартный метод fetch Javascript. Функции инкапсулируют логику сетевых запросов и могут быть использованы в различных частях проекта. В проекте модуль используется в компонентах из modal и main.

5.5 Конфигурационные файлы

Каталог data содержит конфигурационные файлы в формате JSON. Эти файлы описывают отображаемые на пользовательском интерфейсе таблицы. Описание включает в себя имена ячеек таблицы и данные, которые будут отображены в ячейках при начальной загрузке интерфейса. В файлах также

содержатся названия систем оборудования. Конфигурационные файлы позволяют удобно настраивать интерфейс в соответствии с конкретными требованиями системы станции. Конфигурационный файл импортируется в компоненте, в котором реализуется соответствующая таблица.

5.6 Создание интерфейса для конкретной системы оборудования

В ходе работы был разработан пользовательский интерфейс для управления несколькими блоками: система калибровки образцов, система позиционирования детектора, система зарезания рассеянной компоненты пучка. Вид основного окна представлен на рисунке 2, вид дополнительного окна представлен на рисунке 3. На основном окне в таблицах представлена информация о параметрах и статусе механических подвижек систем, а также элементы отправки команд, которые изменяют положение подвижки. Оператор при использовании данного интерфейса чаще всего будет работать именно с изменением координаты положения подвижки, поэтому в дополнительное окно были вынесены задание границ перемещения подвижки, задание скорости и ускорения перемещения подвижки. Работа с регистрами также вынесена на дополнительное окно, так как будет использоваться оператором достаточно редко.

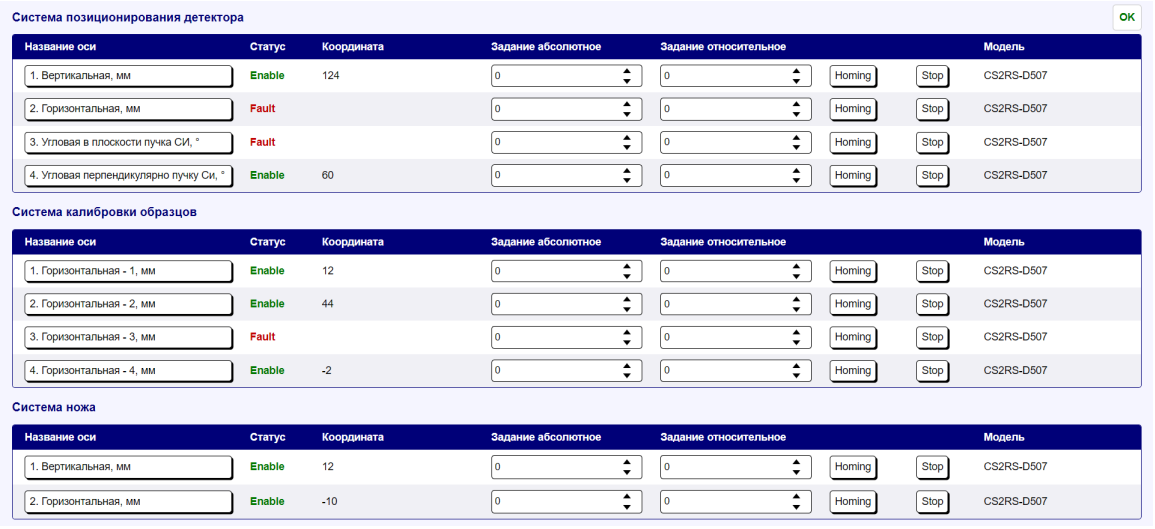

Рисунок 6 - основная страница пользовательского интерфейса

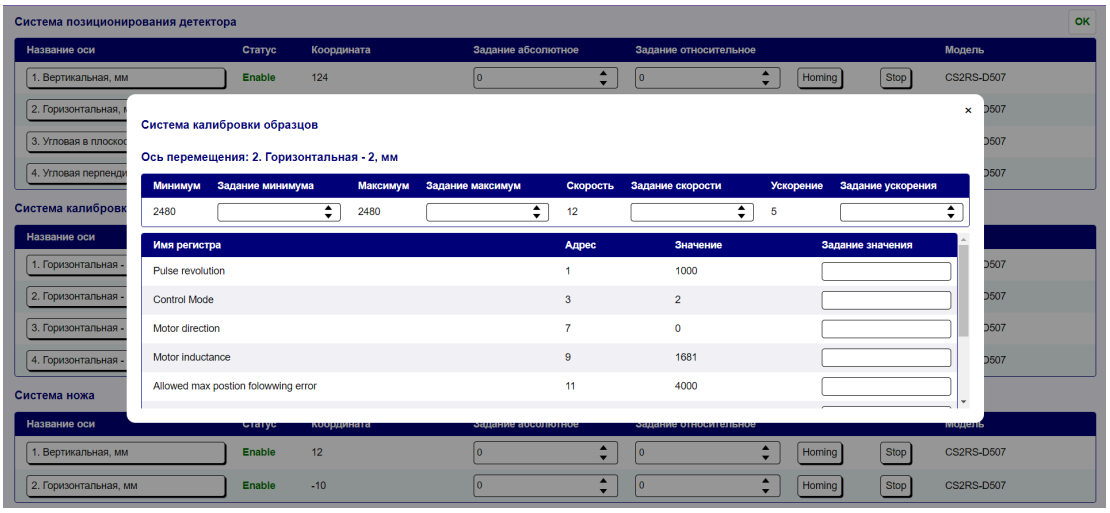

Рисунок 7 - дополнительное окно пользовательского интерфейса

Создание пользовательского интерфейса для управления конкретным блоком оборудования может быть осуществлено быстро и просто. Для этого используется код, разработанный для управления блоками: система калибровки образцов, система позиционирования детектора и система зарезания рассеянной компоненты пучка.

Основные шаги создания пользовательского интерфейса для определенного блока оборудования станции включают:

1. Определение структуры основного окна в main и структуры дополнительного окна в modal. Это включает описание структур таблиц, определение порядка и количества таблиц, подписей над ними. Осуществляется с помощью использования в качестве примера реализованных файлов, содержащихся в main и modal, интерфейса управления: система калибровки образцов, система позиционирования детектора, система зарезания рассеянной компоненты пучка.
2. Настройку конфигурационных файлов в зависимости от требований к отображаемой в таблицах информации.

Базовое поведение, включая открытие и закрытие дополнительного окна, обновление данных в ячейках таблиц, отправку команд и получение данных с сервера, уже определено в файлах, находящихся в каталогах main и modal, а также в App.js, являющимся основным компонентом проекта, и не требует

изменений. Файлы стилей из каталога styles являются адаптивными и не нуждаются в модификации. Файлы index.js, выполняющий инициализацию и запуск приложения в браузере, и index.html, являющийся главной точкой входа в приложение, также не требуют изменений, так как они обеспечивают базовую инфраструктуру приложения.

Разработанный пользовательский интерфейс удовлетворяет изначальным требованиям. Архитектура кода позволяет незатруднительно вносить изменения в проект для реализации интерфейса под определенную систему оборудования.

6 Разработка сервера

Во время разработки серверной части брокер MQTT, являющийся частью автоматизированной системы управления станции быстропротекающие процессы ЦКП СКИФ, не был реализован. Поэтому, помимо взаимодействия с брокером MQTT, в коде сервера для управления и мониторинга состояния оборудования операции чтения и записи данных были реализованы напрямую через базу данных Redis. Такая реализация позволяет при отсутствии брокера проверять работоспособность приложения и оборудования в лабораторных условиях.

Для реализации сервера использовались библиотеки:

1. Express. Библиотека для создания веб-серверов и API. Обеспечивает простоту и гибкость настройки маршрутов, обработки запросов и ответов.
2. Dotenv. Библиотека для управления переменными окружения. Позволяет загружать переменные окружения из файла .env в process.env.
3. Redis. Библиотека для работы с Redis.
4. MQTT. Библиотека для работы с MQTT-протоколом, позволяющая подключаться к MQTT-брокерам, публиковать и подписываться на топики.

Структура серверного кода организована так, чтобы обеспечить модульность и удобство поддержки. Каждая функциональная часть приложения разделена на соответствующие модули.

Структура проекта:

1. index.js.
2. Директория config, в которой находятся конфигурационные файлы.
3. Модуль контроллеров.
4. Модуль клиентов.
5. Модуль маршрутизаторов.
6. Модуль сервисов.
7. Модуль обработки ошибок.

6.1 Основной файл

index.js является основным файлом запуска сервера. Здесь инициализируется сервер, то есть создается экземпляр Express. Подключается Express.json, который отвечает за парсинг тела входящих запросов в формате JSON и делает его доступным в req.body. В index.js реализована загрузка статических файлов, которые являются сборкой пользовательского интерфейса, с которым взаимодействует сервер. Это позволяет предоставить пользовательский интерфейс, когда сервер получает запрос на корневой маршрут. Подключается модуль маршрутизаторов и глобальный обработчик ошибок.

6.2 Конфигурационные файлы

Директория config содержит конфигурационные файлы, которые определяют параметры и настройки, используемые в серверной части приложения. Включение конфигурационных данных в отдельные файлы позволяет изменять параметры приложения без необходимости модифицировать основной код.

Конфигурационные файлы состоят из:

- Файлы в формате JSON, содержащие темы, необходимые для публикации данных на MQTT-брокер, и темы, на которые нужно подписываться для получения данных с брокера.
- Файлы в формате JSON, описывающие ключи, по которым можно получать данные из базы данных Redis, и ключи, по которым осуществляется запись в Redis.
- Файл .env содержит параметры конфигурации, такие как URL базы данных Redis и брокера MQTT, а также порт сервера.

6.3 Модуль контроллеров

Контроллеры содержат обработку запросов, поступающих с клиентской части, и бизнес-логику, определяющую правила обработки данных и взаимодействие с внешними системами.

В модуле есть контроллер для обработки запросов, определяющих информацию о состоянии элементов системы. Если запрос с клиентской части является командой управления оборудованием, контроллер извлекает из тела запроса необходимые параметры, определяет соответствующую тему или ключ из конфигурационных файлов, формирует строку с информацией о команде и вызывает соответствующий метод из модуля клиентов для отправки команды брокеру MQTT или записи в Redis, передавая тему или ключ и строку. Если запрос является запросом на получение данных об оборудовании, контроллер вызывает соответствующий метод из модуля клиента с темами или ключами в качестве параметров и формирует ответ запроса и возвращает его в формате JSON.

В модуле есть контроллер для обработки запросов, определяющих информацию о регистрах микроконтроллеров элементов системы. Если запрос с клиентской части является командой управления регистрами оборудования, контроллер извлекает из тела запроса необходимые параметры, определяет соответствующую тему или ключ из конфигурационных файлов, формирует строку с информацией о команде и вызывает соответствующий метод из модуля клиентов для отправки команды брокеру MQTT или записи в Redis, передавая тему или ключ и строку. Если запрос является запросом на получение данных о регистрах оборудования, контроллер извлекает параметры из запроса, определяет ключ или тему для получения данных о конкретной подвижке и вызывает соответствующий метод из модуля клиента с ключом или топиков, формирует ответ запроса и возвращает его в формате JSON.

6.4 Модуль сервисов

Модуль сервисов отвечает за взаимодействие с внешними системами и выполнением операций, связанных с брокером MQTT и базой данных Redis. Эти сервисы предоставляют необходимые методы для подключения, обмена данными и управления соединениями, которые затем используются в модуле клиентов.

Сервис MQTT обеспечивает логику подключения к брокеру MQTT, подписки на темы и публикации сообщений. Сервис использует библиотеку mqtt.

Сервис Redis обеспечивает логику подключения к базе данных Redis и выполнения операций чтения и записи данных. Сервис использует библиотеку redis.

6.5 Модуль клиентов

Модуль клиентов отвечает за специфическое взаимодействие с внешними системами, такими как брокер MQTT и база данных Redis, в контексте конкретной системы управления. Этот модуль инкапсулирует логику подключения, обмена данными и управления соединениями, предоставляя удобные интерфейсы для использования в контроллерах.

Клиент MQTT обеспечивает подключение к брокеру MQTT и, используя методы сервиса MQTT, осуществляет подписку на темы и публикацию сообщений. В клиенте также хранится последняя полученная информация с брокера по всем топикам, что позволяет контроллеру получать актуальные данные о конкретном топике при вызове метода получения данных.

Клиент Redis обеспечивает подключение к базе данных Redis и выполнение операций чтения и записи данных, используя методы сервиса Redis.

6.6 Модуль маршрутизаторов

Модуль маршрутизаторов определяет эндпоинты, то есть url-адреса, использующихся для запросов, приложения и связывает их с соответствующими контроллерами. В нем определяется, какой метод контроллера должен обрабатывать запрос в зависимости от эндпоинта и типа запроса (GET, POST). Модуль централизует управление эндпоинтами, упрощая их добавление, удаление и модификацию.

6.7 Модуль обработки ошибок

Модуль обработки ошибок отвечает за централизованное управление

ошибками, возникающими при работе приложения. Он перехватывает ошибки, возникшие в процессе обработки запросов, и возвращает клиентской части приложения соответствующие ответы с информацией об ошибке. Это позволяет на пользовательском интерфейсе отображать статус работы приложения.

6.8 Настройка сервера для конкретного блока оборудования станции

Для того чтобы настроить сервер для взаимодействия с конкретной системой оборудования, требуется изменить конфигурационные файлы, определив в них темы и ключи, которые используются для работы с данной системой.

Сервер реализован для управления блоками: система калибровки образцов, система позиционирования детектора и система зарезания рассеянной компоненты пучка. Помимо взаимодействия с брокером MQTT реализован способ передачи данных между контроллером и сервером с помощью базы данных Redis, что упрощает процесс проверки работы системы управления и тестирования оборудования в лабораторных условиях. Архитектура серверного кода предполагает возможность эффективной интеграции дополнительного поведения или изменения существующей логики, потребность в которых может возникнуть при дальнейшей разработке.

7 Тестирование разработанного автоматизированного интерфейса

Во время разработки серверной и клиентской частей для отдельных компонентов и методов были реализованы юнит-тесты. Юнит-тесты обеспечивают надежность и корректное поведение компонентов и методов.

Была проверена работоспособность автоматизированного интерфейса при управлении блоками оборудования: система калибровки образцов, система позиционирования детектора, система зарезания рассеянной компоненты пучка. Эти тесты включали проверку функциональности интерфейса, его взаимодействие со всей цепочкой частей системы управления, корректность выполнения команд управления оборудованием и отображения данных. Фотография, подтверждающая использование автоматизированного интерфейса для управления оборудованием, включающим систему калибровки образцов, систему позиционирования детектора и систему зарезания рассеянной компоненты пучка, представлена на рисунке 8.

Рисунок 8 — использование интерфейса для управления оборудованием

ЗАКЛЮЧЕНИЕ

В результате выполнения научной работы по разработке автоматизированного интерфейса управления оборудованием станций 1-3 «Быстропротекающие процессы» ЦКП «СКИФ», были успешно достигнуты поставленные цель и задача. Работа включала в себя формулировку требований, определение архитектуры системы, обзор существующих решений, выбор подходящих инструментов и технологий для разработки, непосредственно разработку автоматизированного интерфейса и его тестирование.

Разработанный автоматизированный интерфейс включает в себя необходимые элементы для управления системами оборудования станции. Были разработаны программные документы “Описание программы” (Приложение А) и “Руководство оператора” (Приложение Б).

В ходе тестирования автоматизированный интерфейс продемонстрировал способность корректно взаимодействовать с оборудованием, обеспечивая точность и оперативность выполнения команд.

В дальнейшем планируется продолжение работы над созданием автоматизированных интерфейсов для всех систем экспериментального оборудования станции. Это будет включать в себя доработку существующих решений с учетом новых требований, которые могут возникнуть в дальнейшем. В результате автоматизированный интерфейс будет внедрен на станцию 1-3 “Быстропротекающие процессы”, где будет использоваться сотрудниками станции.

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлен с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного

цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

_____ _____

ФИО студента *Подпись студента*

« ____ »_____20 __г.
(заполняется от руки)

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Левичев, Е. Б., Журавлев, А. Н., Золотарев, К. В., Зубавичус, Я. В., Шефер, К. И. Проект создания синхротронного источника поколения 4+ ЦКП «СКИФ» в р. п. Кольцово Новосибирской области: общая информация и статус реализации // Технологическая инфраструктура сибирского кольцевого источника фотонов" СКИФ". – 2022. – С. 8-12.
2. Чернышев В. В., Кузьмичева Г. М. Синхротронное излучение в характеризации новых материалов // ББК 94.3 Р 76. – 2021. – Т. 5. – С. 10.
3. Кондранова А. М. , Гладышева, А. А., Гладышева, А. В., Агафонов, А. П. Использование синхротронного излучения в вирусологии // Здоровье населения и среда обитания. – 2022. – Т. 30. – №. 12. – С. 81-88.
4. Рубцов, И. А., Прууэл, Э. Р., Тен, К. А., Кашкаров, А. О., Аракчеев и др. Концептуальный проект Станции 1-3 «Быстропротекающие процессы» // Технологическая инфраструктура сибирского кольцевого источника фотонов" СКИФ". – 2022. – С. 115-135.
5. Марусин А. Е. Автоматизированные системы управления технологическими процессами, их структура и функции // Международный студенческий научный вестник. – 2022. – №. 3. – С. 17.
6. Benson H., Automation interfaces advance [Электронный ресурс]. – Режим доступа: https://www.controleng.com/articles/automation-interfaces-advance/ (дата обращения: 10.05.2024).
7. Кряжева Е. В., Васина Т. А. Общие подходы к проектированию ВЕБ-приложений // Заметки ученого. – 2021. – №. 9-2. – С.
8. Литвинов М. А., Паньков Д. Н., Бугорский И. А. Программируемый логический контроллер (ПЛК) как основа автоматизации технологических процессов // Поколение будущего: Взгляд молодых ученых. – 2022. – С. 92-96.
9. Дикий Д. И. Анализ протокола MQTT на атаки «отказ в обслуживании» // Научно-технический вестник информационных технологий, механики и

оптики. – 2020. – Т. 20. – №. 2. – С. 223-232.

10. Рубин О. и др. Использование СУБД Redis в качестве промежуточного хранилища данных для POSTGRESQL // StudNet. – 2020. – Т. 3. – №. 9. – С. 1646-1650.
11. Slusariuc R. SCADA systems analysis for industrial procesess // Annals of the University of Petrosani, Electrical Engineering. – 2019. – Т. 21. – С. 17-22.
12. Ignition [Электронный ресурс]. – Режим доступа: https://inductiveautomation.com/ignition/ (дата обращения: 10.05.2024).
13. VTScada [Электронный ресурс]. – Режим доступа: https://www.vtscada.com/what-is-vtscada/ (дата обращения: 10.05.2024).
14. Rapid SCADA [Электронный ресурс]. – Режим доступа: https://rapidscada.ru/ (дата обращения: 10.05.2024).
15. Вершинин Е. В., Хромов А. Е. СРАВНЕНИЕ РАЗЛИЧНЫХ АРХИТЕКТУРНЫХ РЕШЕНИЙ ПРИ РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ // Наука. Исследования. Практика. – 2022. – С. 46-49.
16. React [Электронный ресурс]. – Режим доступа: https://ru.legacy.reactjs.org/docs/getting-started.html (дата обращения: 10.05.2024).
17. Морозов М. А. Сравнение инструментов разработки React, Vue и Angular // Материалы I Международной научно-практической конференции. – 2020. – С. 545.
18. Чепегин И. Д. Серверный JavaScript-преимущества и недостатки node. JS // Вестник науки и образования. – 2020. – №. 12-1 (90). – С. 18-20.
19. Палагин Д. А. HTTP ЗАПРОСЫ И МЕТОДЫ АНАЛИЗА //Новые информационные технологии в научных исследованиях НИТ-2021. – 2021. – С. 14-16.

ПРИЛОЖЕНИЕ А

АВТОМАТИЗИРОВАННЫЙ ИНТЕРФЕЙС УПРАВЛЕНИЯ ОБОРУДОВАНИЕМ СТАНЦИИ 1-3 «БЫСТРОПРОТЕКАЮЩИЕ ПРОЦЕССЫ» ЦКП «СКИФ»

ОПИСАНИЕ ПРОГРАММЫ

Листов 10

Новосибирск, 2024

СОДЕРЖАНИЕ

АННОТАЦИЯ.....38
1. Общие сведение.....39
2. Функциональное назначение..... 40
3. Описание логической структуры.....41
4. Используемые технические средства.....42
5. Вызов и загрузка.....43
6. Входные данные.....44
7. Выходные данные.....45

АННОТАЦИЯ

В данном программном документе приведено описание программы управления оборудованием станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ». Программа представляет из себя веб-приложение. Основным языком программирования, на котором программа написана, является Javascript.

Основной функционал программы — управление оборудованием и мониторинг оборудования станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ».

Оформление программного документа «Описание программы» произведено по требованиям ЕСПД (ГОСТ 19.402-78, ГОСТ 19.105-78).

1. Общие сведение

Наименование программы — “Автоматизированный интерфейс управления оборудованием станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ»”. Сокращенное наименование программы — “Автоматизированный интерфейс управления”.

Для функционирования программы необходимы следующие программные средства:

- Node.js версии 18 и выше;
- Наличие npm;
- Браузер, поддерживающий корректную работу клиентской части приложения, написанной на React 18 версии.

Программа написана на JavaScript с использованием фреймворка React для клиентской части.

2. Функциональное назначение

Программа предназначена для обеспечения взаимодействия пользователя с оборудованием станции 1-3 «Быстропротекающие процессы». Программа обеспечивает возможность отправки команд управления на оборудование и отслеживание состояния оборудования через веб-интерфейс.

Программа предназначена только для использования в качестве составляющей части системы управления станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ».

3. Описание логической структуры

Программа является составляющей частью системы управления станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ». Программа предназначена для обработки изменений конечного пользователя на веб-интерфейсе, формирования соответствующих сообщений и отправки сообщений на брокер MQTT. Также программа предназначена для обработки сообщений, приходящих с брокера MQTT, и отображения соответствующей информации на веб-интерфейсе.

Программа реализует клиент-серверную архитектуру. Клиентская часть взаимодействует с сервером посредством HTTP запросов. Сервер взаимодействует с остальной системой управления через MQTT-протокол.

Структура программы:

1. Клиентская часть. Пользовательский интерфейс, реализованный с помощью React. Предоставляет возможность конечному пользователю отправлять команды управления и отображение информации об оборудовании и статуса работы системы.
2. Серверная часть. Express.js сервер. Предназначен для обработки запросов с клиентской части и взаимодействия с MQTT брокером.
3. Программа взаимодействует с брокером MQTT для обмена данными с оборудованием.

4. Используемые технические средства

Программа может эксплуатироваться на персональном компьютере. Работа пользователя программы происходит в браузере.

5. Вызов и загрузка

Запуск сервера выполняется командой node index.js в терминале из корневой директории проекта.

После запуска сервера клиентская часть доступна через веб-браузер с указанием соответствующего URL-адреса сервера.

6. Входные данные

Входные данные включают параметры, которые пользователь предоставляет с помощью полей ввода и нажатия кнопок на интерфейсе.

На вход сервер от брокера MQTT принимает сообщения, содержащие информацию в виде строки, преобразованной из JSON объекта, об оборудовании.

7. Выходные данные

Данные от клиентской части передаются в формате JSON через HTTP-запросы серверу. Выходные данные включают отображение на интерфейсе текущих параметров состояния оборудования и результатов выполнения команд управления. Сервер отправляет сообщения MQTT, где тема является строкой, описывающей часть оборудования и команду, которая должна быть применена, а тело является строкой, преобразованной из JSON объекта, в которой хранится информация о параметрах, с которыми должна быть применена команда.

ПРИЛОЖЕНИЕ Б

АВТОМАТИЗИРОВАННЫЙ ИНТЕРФЕЙС УПРАВЛЕНИЯ ОБОРУДОВАНИЕМ СТАНЦИИ 1-3 «БЫСТРОПРОТЕКАЮЩИЕ ПРОЦЕССЫ» ЦКП «СКИФ»

РУКОВОДСТВО ОПЕРАТОРА

Листов 7

Новосибирск, 2024

СОДЕРЖАНИЕ

АННОТАЦИЯ.....48
1. Назначение программы.....49
2. Условие выполнения программы.....50
3. Выполнение программы..... 51
4. Сообщения оператору..... 52

АННОТАЦИЯ

В данном программном документе приведено руководство оператора для автоматизированного интерфейса управления оборудованием станции 1-3 «Быстропротекающие процессы» ЦКП «СКИФ». Руководство предоставляет информацию о том, как использовать программу для обеспечения управления оборудованием и мониторинга состояния оборудования.

Оформление программного документа «Руководство оператора» произведено по требованиям ЕСПД (ГОСТ 19.101-77, ГОСТ 19.505-79, ГОСТ 19.105-78).

1. Назначение программы

Программа управления оборудованием станции 1-3 «Быстропротекающие процессы» предназначена для автоматизации управления и мониторинга оборудования через веб-интерфейс. Она обеспечивает дистанционное управление оборудованием, отображение текущего состояния системы оборудования.

2. Условие выполнения программы

Серверный код исполняется на персональном компьютере, на котором установлена платформа Node.js версии 18 и выше и установлен пакетный менеджер npm.

Клиентский код исполняется в браузере, поддерживающим корректную работу клиентской части приложения, написанной на React 18 версии.

Для взаимодействия программы с системой управления должен обеспечиваться доступ к брокеру MQTT.

3. Выполнение программы

Последовательность действий оператора:

1. Запустить сервер, выполнив команду node index.js в терминале из корневой директории проекта.
2. Открыть веб-браузер и ввести адрес сервера. На веб-интерфейсе оператору будут доступны элементы управления для взаимодействия с оборудованием и элементы, отображающие состояние оборудования.
3. Использовать кнопки и поля ввода для отправки команд и изменения параметров оборудования.
4. Ожидать ответа от системы, который будет отображаться на экране.
5. Закрыть вкладку веб-браузера.
6. Остановить сервер, используя сочетание клавиш Ctrl+C в терминале.

4. Сообщения оператору

В текстовом элементе, расположенном на пользовательском интерфейсе отображается статус работы приложения. Там указывается, осуществляется ли связь с сервером и связь с брокером MQTT. В консоли, где запускается сервер, при возникновении ошибки в работе появляется сообщение с описанием, на каком этапе работы приложения и какая ошибка произошла. Данная информация предоставляет возможность оператору определить, какое действие следует совершить для корректной работы приложения.