

РЕФЕРАТ

ПРОГРАММНОЕ СРЕДСТВО СБОРА И ВИЗУАЛИЗАЦИИ ТЕХНОЛОГИЧЕСКИХ ПАРАМЕТРОВ ХИМИЧЕСКОГО ЦЕХА НА ПЛАТФОРМЕ .NET: дипломный проект / Г.Б. Насанович – Минск: БГУИР, 2023, – п.з. – 81 с., чертежей (плакатов) – 6 л. формата А1.

Объектом проектирования является программное средство автоматического сбора технологических параметров.

Цель работы – автоматизация процессов сбора данных с контроллеров omron.

Произведен анализ предметной области и исследованы существующие аналоги. Разработаны функциональные требования.

Произведено проектирование программного средства, построены диаграммы и схемы базы данных и алгоритмов.

Произведена разработка программного средства с использованием языков программирования C# и JavaScript, платформ Asp.Net и Node.JS, системы управления базой данных MySQL.

Использование данного программного средства позволяет упростить способы за контролем технологических параметров и ускорить получение информации на производстве.

В ходе выполнения технико-экономического обоснования были проведены расчеты целесообразности выполнения данного проекта. Были обоснованы затраты на реализацию и была высчитана возможная рентабельность проекта.

СОДЕРЖАНИЕ

| | |
|---|----|
| Введение | 6 |
| 1 Аналитический обзор программных продуктов, методов и подходов по теме дипломного проекта..... | 8 |
| 1.1 Анализ существующих решений по теме дипломного проекта | 8 |
| 1.2 Постановка задач дипломного проектирования..... | 14 |
| 2 Моделирование предметной области, разработка функциональных требований и составление их спецификации | 16 |
| 2.1 Общие сведения и требования к работе программного средства..... | 16 |
| 2.2 Описание функциональности программного средства | 16 |
| 2.3 Разработка протоколов связи | 23 |
| 2.4 Разработка информационной модели | 24 |
| 2.5 Разработка спецификации функциональных требований..... | 25 |
| 3 Проектирование программного средства..... | 27 |
| 3.1 Разработка программной архитектуры | 27 |
| 3.2 Проектирование архитектуры программного средства | 28 |
| 3.3 Проектирование алгоритма соединения пользователя и контроллера .. | 30 |
| 3.4 Проектирование алгоритма добавления новой команды для контроллера..... | 32 |
| 4 разработка программного обеспечения | 34 |
| 4.1 Выбор и обоснование языка и среды разработки программного средства | 34 |
| 4.2 Разработка базы данных | 35 |
| 4.3 Разработка программной архитектуры | 40 |
| 5 Тестирование и проверка работоспособности программного средства | 42 |
| 6 Руководство пользователя | 44 |
| 6.1 Авторизация пользователя | 44 |
| 6.2 Просмотр данных с контроллера пользователем | 44 |
| 6.3 Управление выходами контроллера администратором | 45 |
| 6.4 Работа с возможностями администратора | 46 |
| 7 Техничко– экономическое обоснование разработки программного средства сбора и визуализации технологических параметров химического цеха на платформе .NET | 52 |
| 7.1 Описание функций, назначения и потенциальных пользователей ПО.. | 52 |
| 7.2 Расчет затрат на разработку программного продукта..... | 52 |
| 7.3 Оценка результата (эффекта) от использования ПО | 55 |
| Заключение | 57 |
| Список использованных источников | 58 |
| Приложение А (Обязательное) Исходный код программного средства | 59 |

ВВЕДЕНИЕ

Современные производства являются сложными системами состоящие из большого количества технологических процессов. Каждый процесс состоит из технологических операций. Для правильного выполнения технологической операции необходима контролировать входные и выходные технические параметры. При использовании простых технологических процессов человек может самостоятельно отслеживать параметры и изменять их, но если технологический процесс имеет большое количество параметров, большую скорость изменения параметров, то человек не сможет правильно осуществлять необходимые операции. В это время на место человека приходят автоматизированные системы управления технологических процессов.

Электроэнергетика относится к одной из самых важных отраслей промышленности. Генерация электроэнергии является первым этапом доставки электроэнергии конечному пользователю.

Для генерации электроэнергии используются различные способы генерации. К этим способам относятся:

- ядерная энергетика;
- тепловая энергетика;
- гидроэнергетика;
- альтернативная энергетика.

На территории нашей страны основным производителем электроэнергии являются тепловые электростанции. Для тепловых электростанций вода является одним из важных компонентов для производства.

Вода, взятая из обычных источников, не подходит для ТЭС. В не отчищенной воде содержится большое количество растворимых и не растворимых веществ. Примеси в воде приносят много проблем при использовании. К таким проблема относятся накипь и ускоренная коррозия.

Накипь – это твердые отложения солей, образующиеся в процессе выпаривания воды. Данные отложения приводят к снижению эффективности водонагревательных устройств, вызывать перегрев оборудования, могут приводить к закупориванию труб. Проводить ручную очистку на электростанции невозможно, так как для ручной очистки необходимо остановить полностью производство, что ведет к огромным потерям.

Газы растворимые в воде могут приводить к ускоренному коррозионному металлических деталей, что приводит к необходимости постоянно проводить сложные ремонтные работы.

Очистка воды – один из необходимых технологических процессов. Данный процесс позволяет подготовить воду к поступлению в водонагреватель, что значительно снижает ущерб от примесей в воде.

Для автоматизации технологического процесса водоочистки используются системы, основанные на программируемых логических контроллерах.

Данные устройства позволяют выполнять большинство операций в автоматическом режиме. ПЛК зачастую работают в режиме реального времени, что позволяет отслеживать параметры в жестко заданных временных промежутках. ПЛК имеют большую надежность и отказоустойчивость.

Часть задач на данный момент нет возможности автоматизировать. Человек должен контролировать правильность выполнения процессов, а также человек может принимать решения, которые не могут быть точно запрограммированы. Человек должен нести ответственность за контролируемый процесс.

Целью данного проекта является разработка программного средства, которое сможет облегчить сбор и преобразование данных, полученных с контроллеров Omron corporation, для удобного использования человеком.

1 АНАЛИТИЧЕСКИЙ ОБЗОР ПРОГРАММНЫХ ПРОДУКТОВ, МЕТОДОВ И ПОДХОДОВ ПО ТЕМЕ ДИПЛОМНОГО ПРОЕКТА

1.1 Анализ существующих решений по теме дипломного проекта

Необходимость в автоматизации создало большое количество различных инструментов для создания проектов для ПЛК. Такие инструменты называются SCADA. Они могут поставляться как разработчиком контроллеров, так и сторонними разработчиками. Для контроллеров Omron corporation существуют крупные программные комплексы, так и относительно небольшие программные средства. К большим программным комплексам можно отнести комплекс CX-One, поставляемый непосредственно производителем контроллеров, также можно отнести комплекс Master SCADA производства «МПС софт». К небольшим программным средствам относятся большой набор некоммерческих программ, доступных с открытым программным кодом в интернете.

Основная проблема небольших программных решения является то, что они за частую реализуют небольшую часть функциональности, которой недостаточно для удобного использования данного приложения.

Первым делом рассмотрим программный комплекс от производителей контроллеров Omron. CX-One состоит из большого числа специализированных инструментов, таких как CX-Designer, CX-Supervisor, CX-Programmer, CX-Simulator, CX-Motion и других.

CX-Designer предоставляет возможность создавать графические интерфейсы для человеко-машинных интерфейсов. На рисунке 1.1 представлен пользовательский интерфейс CX-Designer, с помощью которого можно создавать новые проекты для интерфейсов. Данная программа позволяет проектировать много экранные приложения, каждый экран может содержать различные базовые компоненты.

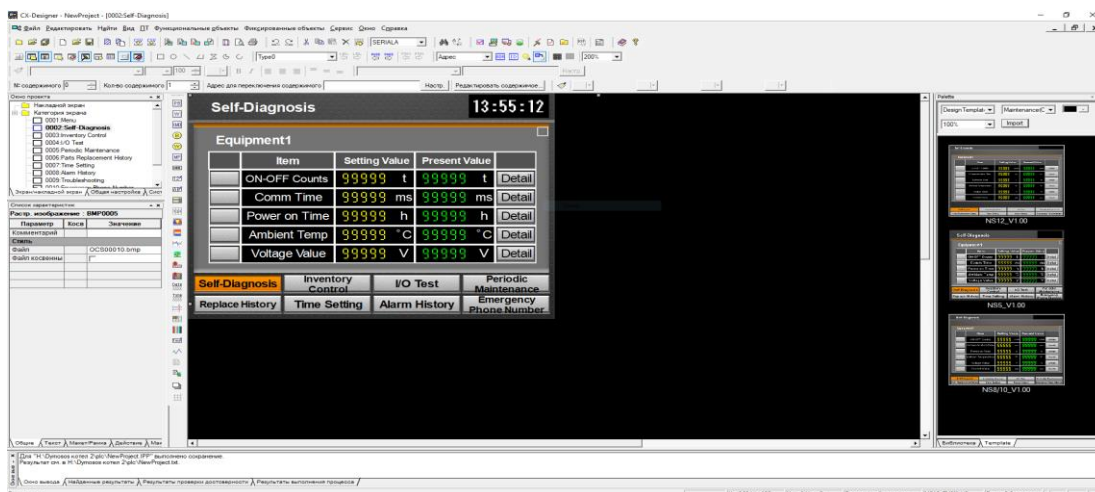


Рисунок 1.1 – Пользовательский интерфейс CX– Designer

Компоненты находятся на панели инструментов, что обеспечивает быстрый доступ к ним. Для размещения компонента необходимо просто перетянуть его из панели инструментов на рабочую область.

Каждый компонент содержит большой список настраиваемых свойств для гибкой настройки. На рисунке 1.2 показано окно с настройкой свойств для числового отображения.

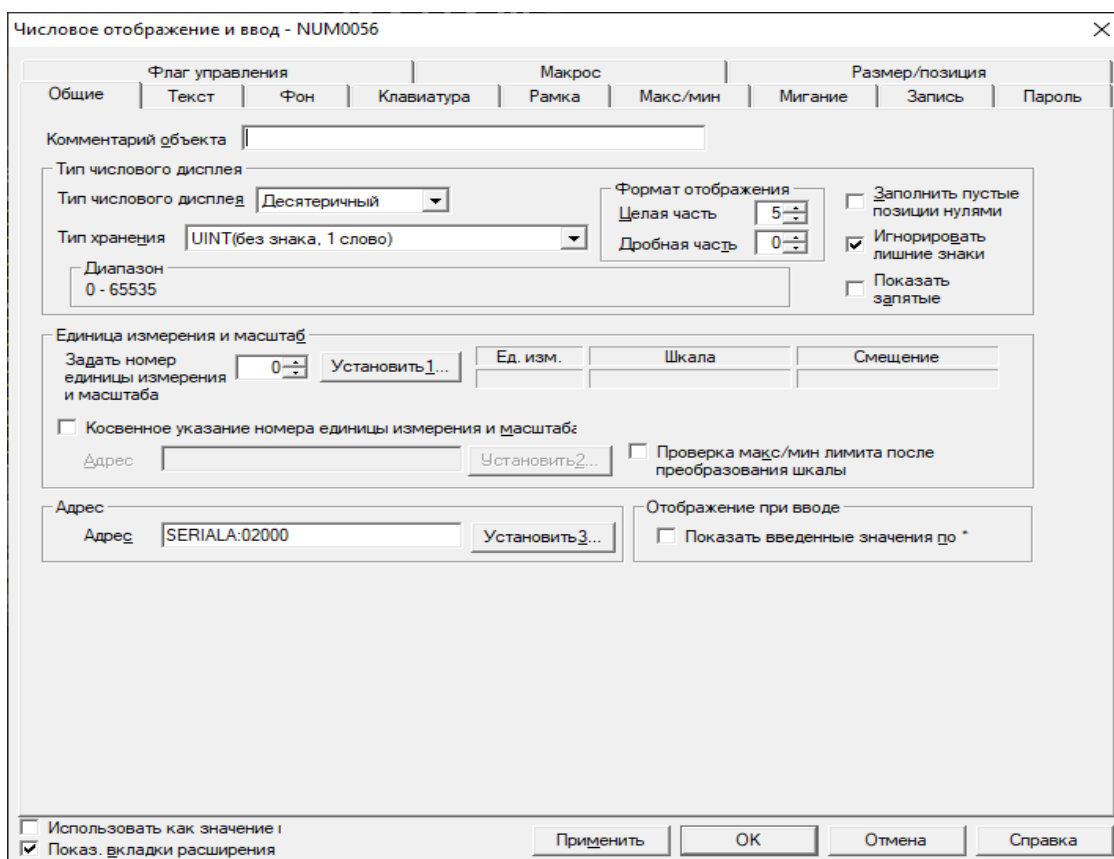


Рисунок 1.2 – Настройка свойств для числового отображения

Для добавления возможности отслеживать параметры необходимо добавить ПЛК в список доступных. ПЛК могут соединяться несколькими способами:

- с помощью последовательных портов;
- по сети Ethernet;
- с помощью Controller Link модуля.

В настройках связи мы можем выбирать и комбинировать разные виды связи. Есть возможность более тонкой настройки запросов:

- настройка задержек для отправления;
- настройка интервалов между сообщениями;
- настройка карт маршрутизации.

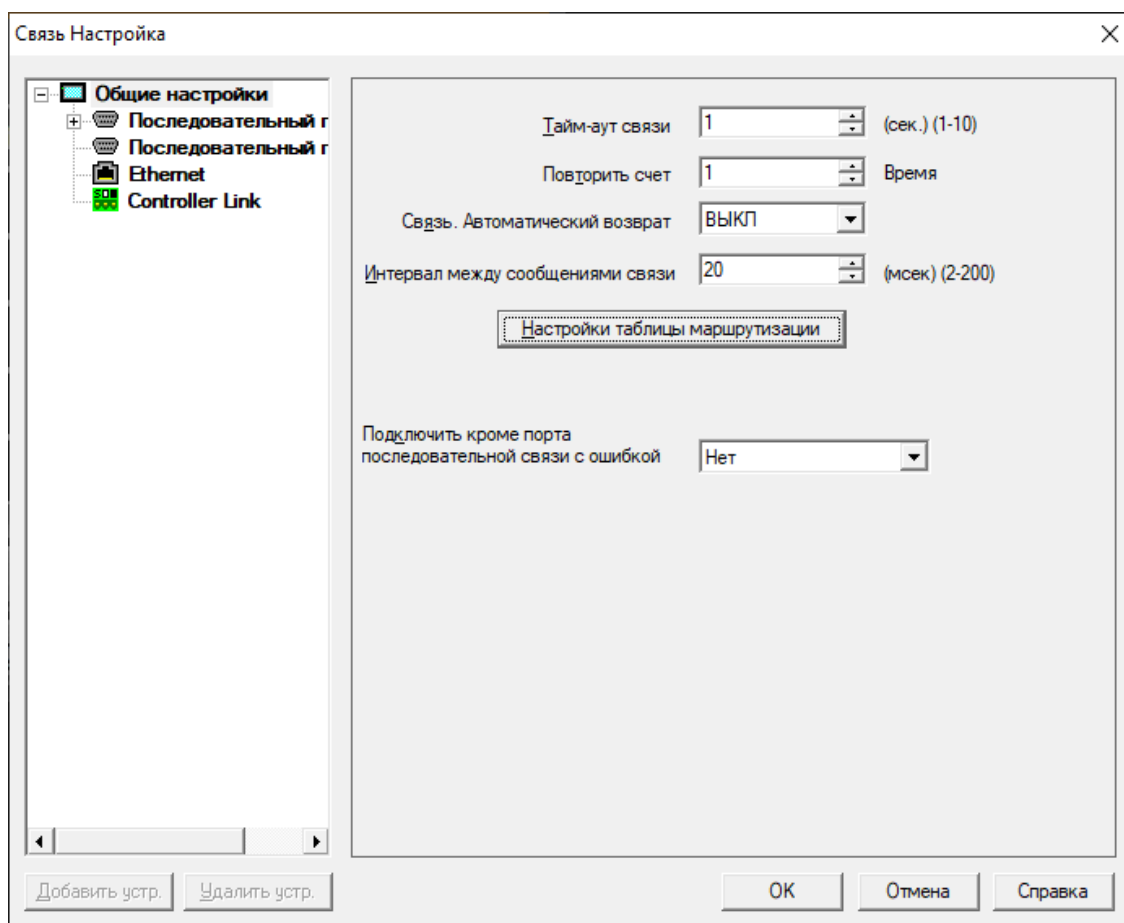


Рисунок 1.3 – Общие настройки связи

Последовательный порт может использоваться только для работы на небольшом расстоянии. Также данный тип связи не позволит соединить большое количество ПЛК.

При использовании сети Ethernet можно связываться с контроллером на большом расстоянии, также можно связываться с большим количеством кон-

троллеров. Для соединения по сети Ethernet необходимо знать адрес контроллера в сети Ethernet и сетевой адрес ПЛК в локальной группе контроллеров.

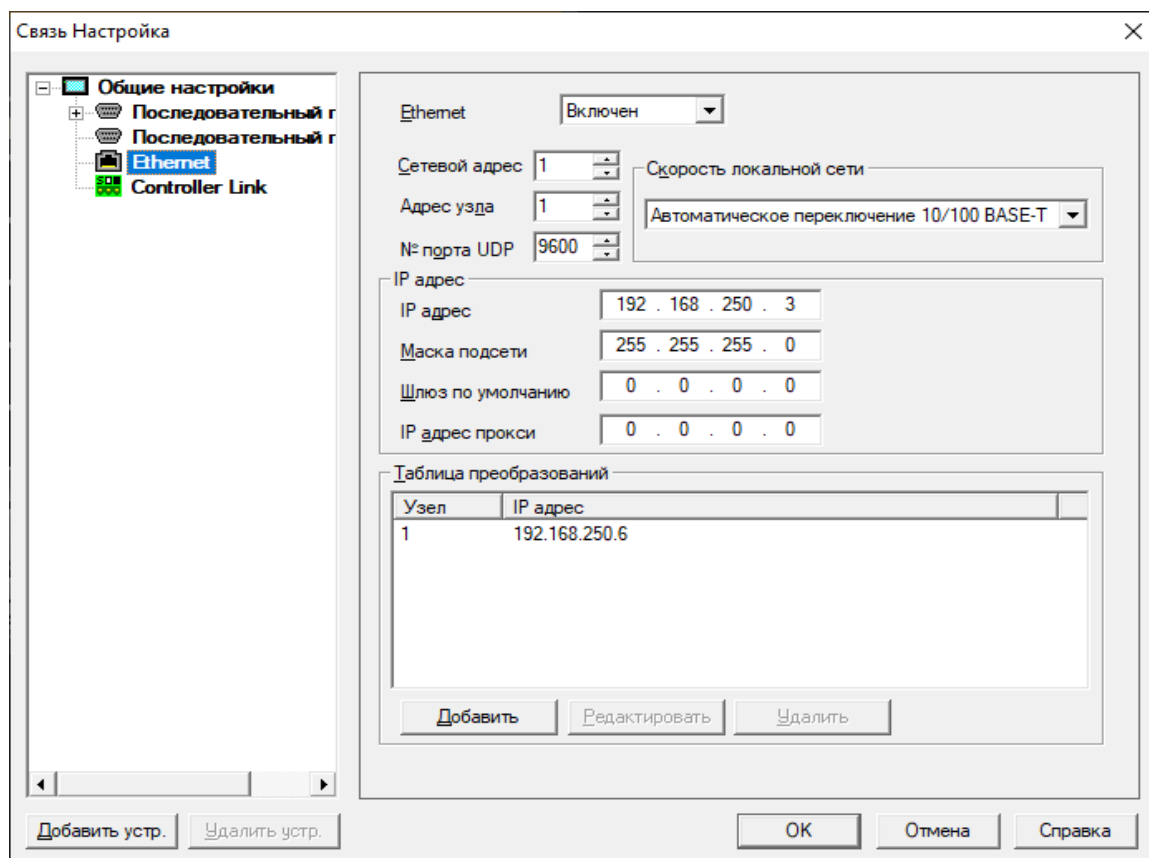


Рисунок 1.4 – Настройки подключения по сети Ethernet

После установления связи с контроллером, можно настраивать передаваемые значения. Для получения значений от контроллера необходимо указать адрес значения в контроллере. Адрес значения состоит из области памяти и сдвига в данной области.

CX– Designer дает много возможностей для создания интерфейсов для контрольных панелей, но также обладает рядом недостатков:

- отсутствие возможности написания интерфейсов для контролируемых панелей других производителей;
- нет возможности соединяться с контроллерами по TCP протоколу;
- отсутствие возможности соединения с контроллерами других производителей;
- нет возможности писать мульти платформенные интерфейсы.

CX– Supervisor – программное средство предназначенное для создания пользовательских интерфейсов для персональных компьютеров. Данное программное средство дает возможность создать приложения для использования на персональных компьютерах на операционной системе Windows.

Разработка пользовательского интерфейса достаточно простая. CX– Supervisor имеет интуитивно понятный интерфейс, обширную документацию и большое количество стандартных объектов для проектирования пользовательского интерфейса. На рисунке 1.5 показан пользовательский интерфейс CX– Supervisor.

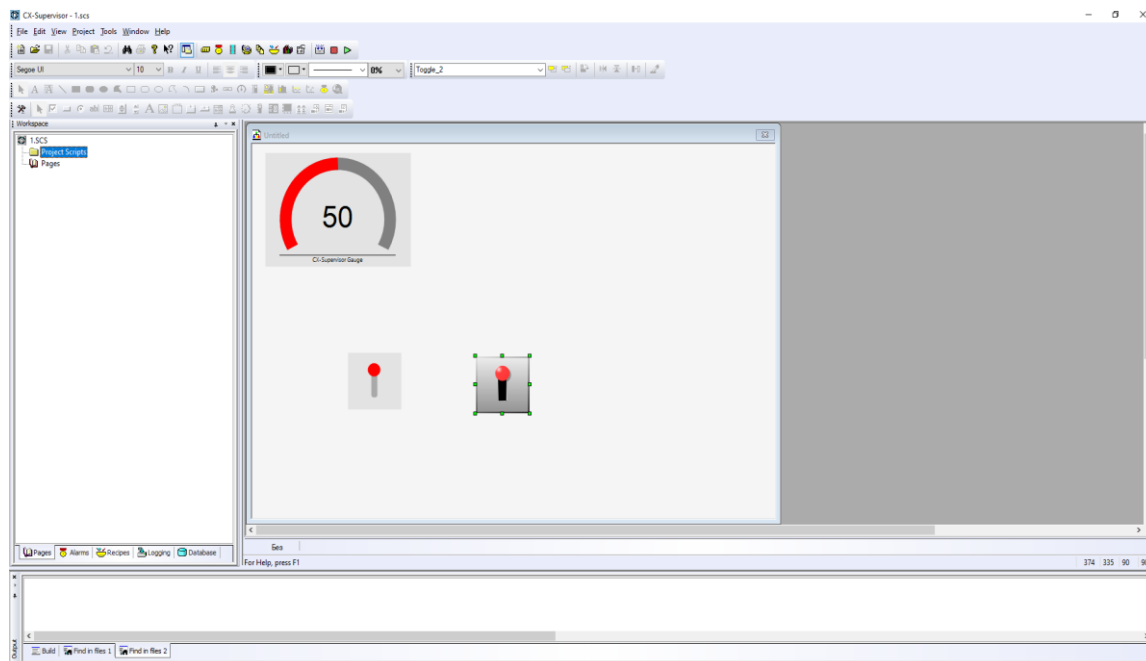


Рисунок 1.5 – Пользовательский интерфейс CX–Supervisor

CX–Supervisor позволяет писать достаточно сложные приложения для пост обработки данных. Приложение поддерживает скрипты, написанные на языках VB script и CX–Supervisor script. Скрипты позволяют соединиться с базой данных и отправлять запросы к базе данных.

Возможность работы с базой данных существенное отличие CX– Supervisor от CX–Designer. Для работы с базой данных необходимо, чтобы на каждом устройстве была установлена СУБД и была копия базы данных, что является не совсем удобным способом соединения.

Основными недостатками данного приложения являются:

- вычисления происходят на каждом устройстве из– за этого необходимая производительность устройства, повышается;
- необходимо создавать несколько вариантов приложения для работы с базой данных, так как в противном случае в базе данных будут копии вычисленных данных;
- приложение возможно запустить только на операционной системе windows 10 и 11, что вносит ограничения на системные требования персонального компьютера.

Создания приложения в CX–Supervisor подходит для тех случаев, когда нет необходимости просматривать информацию на нескольких устройства.

Рассмотрим Master SCADA D4 – это относительно новый проект. При первом запуске можно сразу же отметить информативный и красивый интерфейс приложения, его можно увидеть на рисунке 1.6

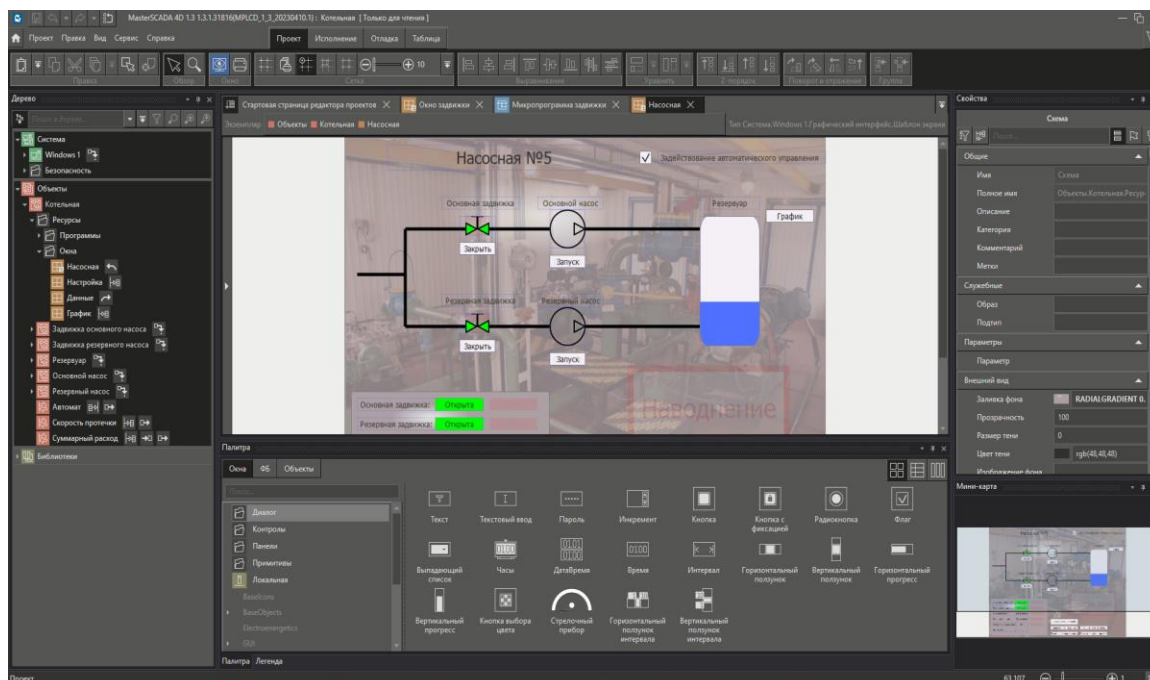


Рисунок 1.6 – Пользовательский интерфейс Master SCADA 4d

Данная SCADA позволяет разрабатывать приложения для контроллеров, серверной части и отдельно пользователей.

Master SCADA можно разделить на две части: на среду разработки и среду исполнения. Среда разработки работает только на ОС Windows. Среда исполнения может исполнять код на различных архитектурах и операционных системах, это позволяет повысить производительность, но в то же время это вносит неудобство: при написании кода необходимо полностью переписывать исполняемый код, даже при незначительных изменениях кода.

Master SCADA позволяет разрабатывать пользовательские веб интерфейсы, что позволяет запустить интерфейс из любого браузера, который поддерживает HTML5, есть возможность писать скрипты для баз данных, что очень упрощает разработку, также данное приложение поддерживает возможность написания функций для вычисления данных на языках:

- C#
- ST
- FBD
- LD
- SFC

Программы, реализованные на языках C# и ST, являются достаточно удобными для реализации специфических задач.

Написание программы в Master SCADA для контроллеров не совсем удобная, в приложении присутствуют уже готовые функциональные блоки, которые соединяются по средствам линий. Функциональные блоки можно размещать без четко выраженной иерархии, что создает достаточно запутанные схемы при разработке.

Master SCADA содержит большое количество библиотечных компонентов, которые упрощают разработку пользовательского интерфейса.

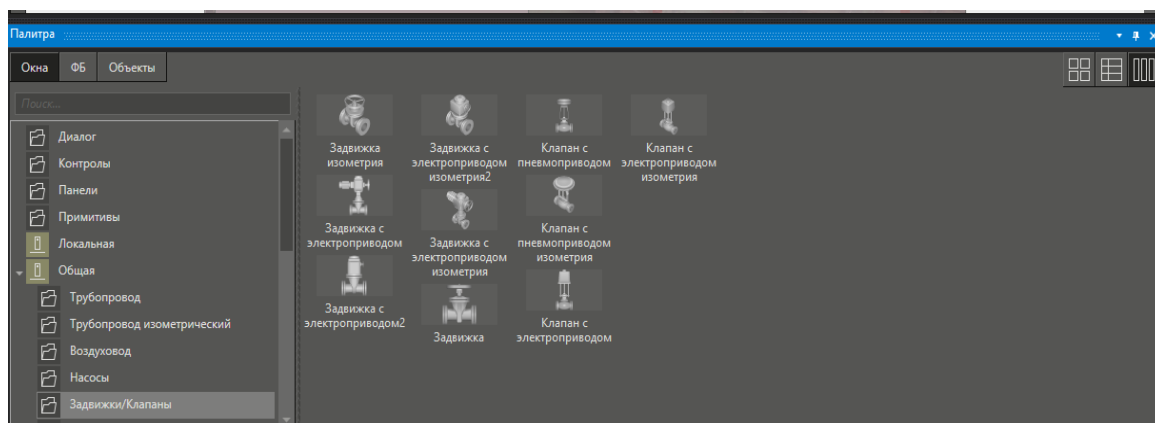


Рисунок 1.7 – Пример пользовательских компонентов

Master SCADA поддерживает большой список различных протоколов связи, которые реализованы производителем:

- OPC;
- UA/DA/HAD;
- Modbus RTU/TCP;
- BACNet;
- Profinet;
- Omron FINS;
- и другие.

Большое количество протоколов связи позволяет создавать сети, состоящие из разных контроллеров, что позволяет выбирать контроллеры для специфических задач.

1.2 Постановка задач дипломного проектирования

В ходе исследования существующих программных решений были выявлены следующие недостатки:

- отсутствие кроссплатформенных сервисов;
- высокая стоимость решений;
- отсутствие возможности изменять запросы во время выполнения программы;

- отсутствие возможности интеграции с другими программными решениями;
- отсутствие возможности изменения топологии сети;
- отсутствие возможности создавать отчеты.

Исходя из перечисленных недостатков, целью дипломного проектирования является разработка программного средства, которое будет способно реализовать функциональность, способную устранить выделенные недостатки

Для достижения поставленной цели выделим основные задачи:

- определение требований к программному средству и составить спецификацию программного средства;
- определить необходимые технологии и языки программирования для реализации программного средства;
- провести проектирование программной архитектуры;
- провести проектирование пользовательского интерфейса;
- разработать модель данных;
- реализовать необходимые алгоритмы и протоколы связи;
- провести тестирования модулей;
- провести тестирование конечного программного продукта.

2 МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ, РАЗРАБОТКА ФУНКЦИОНАЛЬНЫХ ТРЕБОВАНИЙ И СОСТАВЛЕНИЕ ИХ СПЕЦИФИКАЦИИ

2.1 Общие сведения и требования к работе программного средства

Основными задачами данного программного средства являются:

- предоставление пользователю возможность отслеживать технологические параметры;
- возможность изменять список отслеживаемых параметров в режиме реального времени;
- сохранения технологических параметров.

Пользователем данного программного средства может быть только человек, который имеет учетную запись. Для доступа к сервису необходимо иметь устройство с выходом в локальную сеть Ethernet.

Программное средство должно быть рассчитано для использования большим количеством человек, которые могут как просматривать параметры системы, так и изменять некоторые параметры.

2.2 Описание функциональности программного средства

Для описания функциональности программного средства будет использоваться диаграмма вариантов использования. Данный тип UML диаграмм описывает концептуальное взаимодействие между пользователем и системой.

В ходе исследования предметной области и анализа существующих аналогов был определен список пользователей, которые будут пользоваться данной системой. К таким пользователям относятся:

- анонимный пользователь;
- авторизованный пользователь;
- администратор.

Составим функциональные требования для каждой группы пользователей. Начнем с анонимного пользователя, к данному типу пользователей отнесем всех не зарегистрированных пользователей, а также пользователей, которые не прошли процедуру аутентификации. Анонимные пользователи должны получать только возможность аутентификации.

Нельзя позволить любым людям просматривать данные с контроллеров, и тем более изменять состояния системы.

Следующий тип пользователей – аутентифицированный пользователь. Данные пользователи прошли процедуру аутентификации и являются полноценными пользователями системы. Рассчитывается, что данный тип пользователей будет самым многочисленным. Аутентифицированный пользователь, в зависимости от группы, имеет возможность получать данные от контроллера

в режиме реального времени, а также просматривать отчетов по изменению за определенный период. Для аутентификации пользователь обязан ввести логин и пароль, который был получен от администратора.

Необходимо предусмотреть возможность выходить из системы для аутентифицированного пользователя. После выхода из системы пользователь является анонимным и не может больше выполнять никаких действий. На рисунке 2.1 можно увидеть диаграмму использования анонимного пользователя и авторизованного пользователя.

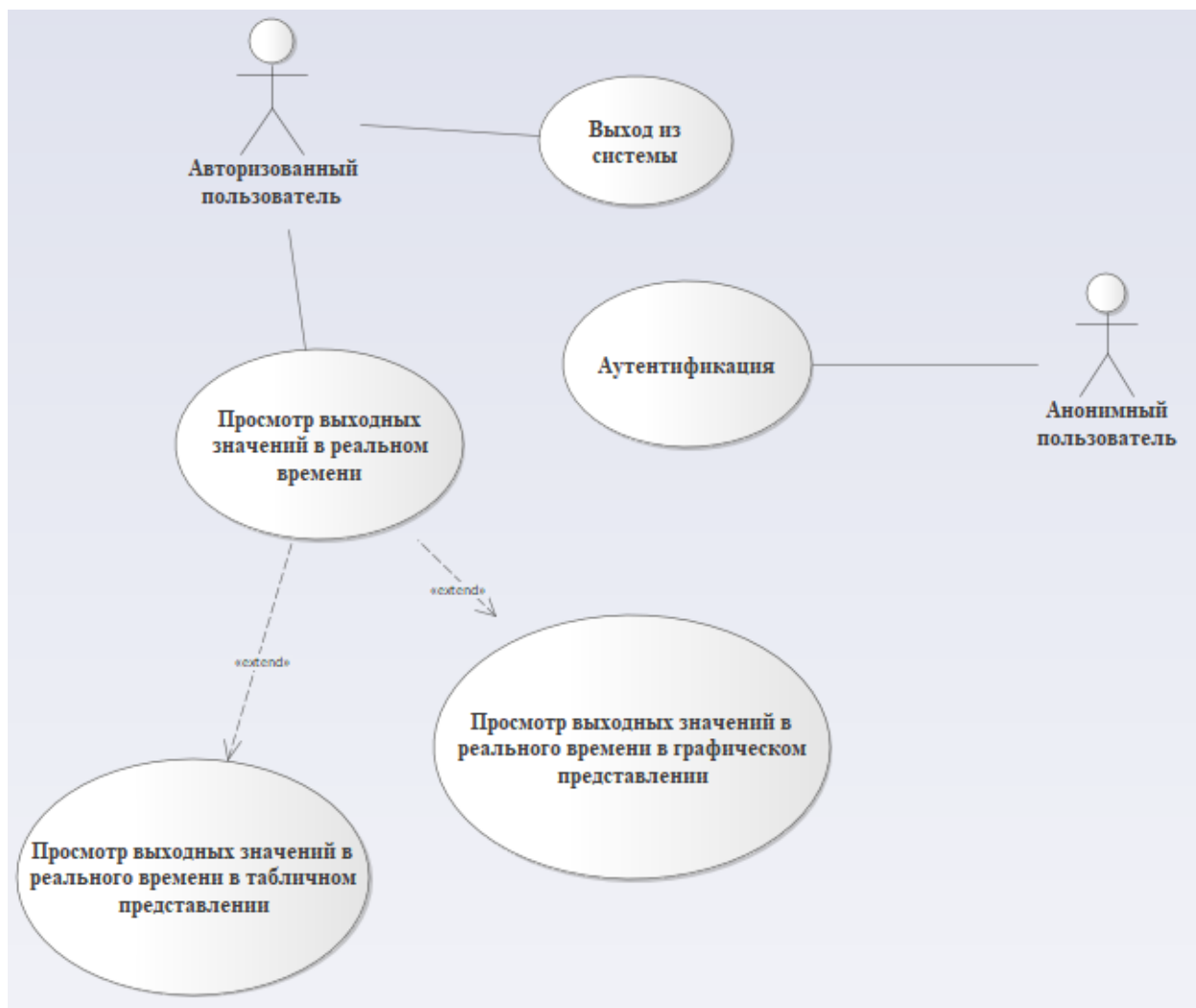


Рисунок 2.1 – Диаграмма использования анонимного и авторизованного пользователя

Отслеживание за изменениями в реальном времени имеют два способа отображения – графический и табличный.

Тип пользователя администратор – является аутентифицированным пользователям с расширенными правами. Данный тип пользователя имеет возможность помимо просмотра данных также вносить изменения в работу системы.

Администратор ответственен за создание новых пользователей. В процессе добавления пользователя, администратор обязан ввести логин и пароль пользователя, при создании нового пользователя администратор может также выбирать в какие группы добавить пользователя. Добавление в группу возможно и после добавления пользователя, также, когда пользователь уже добавлен, администратор может просматривать все доступные этому пользователю группы и по необходимости удалять пользователя из группы. Диаграмма использования для добавления пользователя расположена на рисунке 2.2.

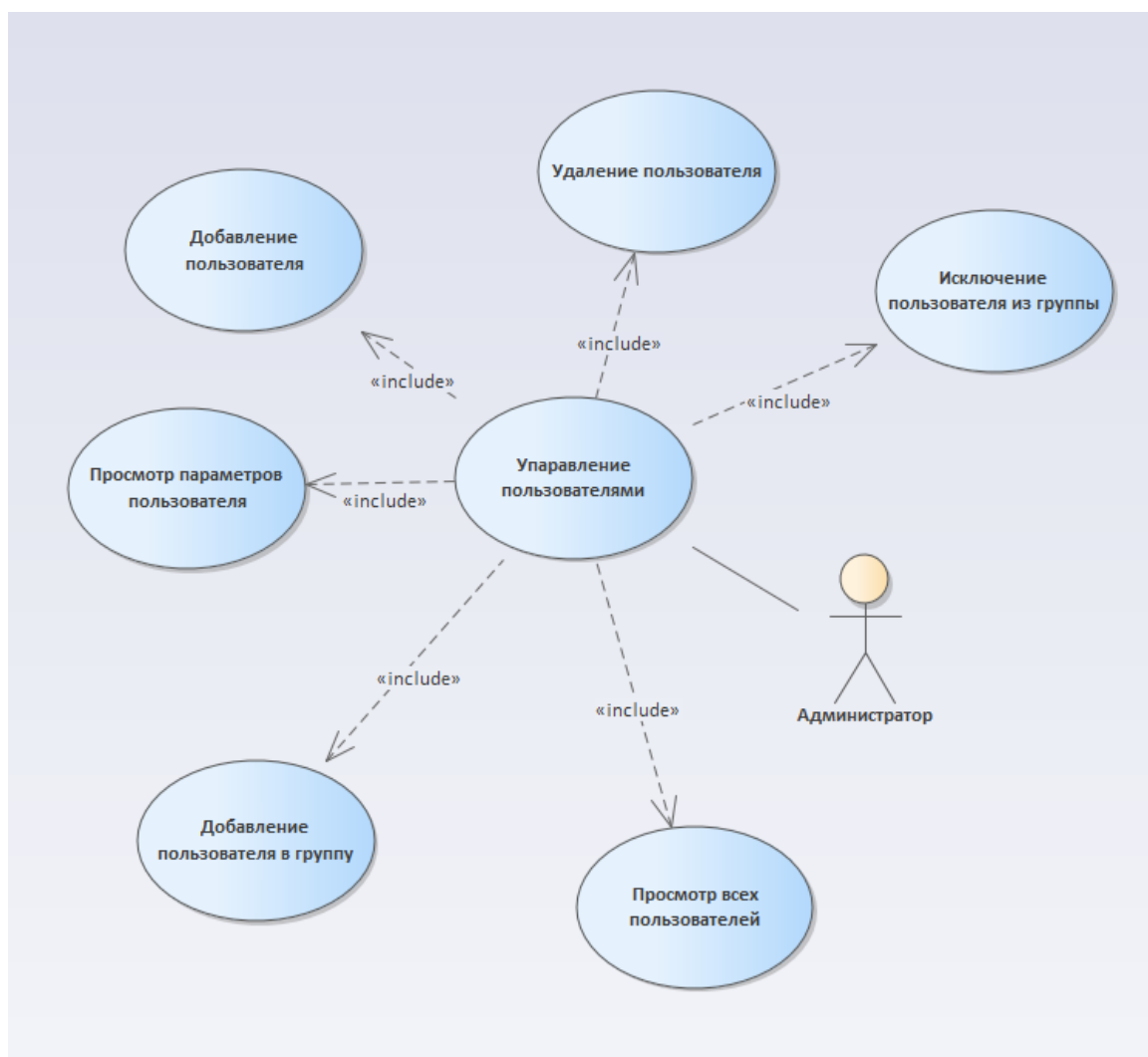


Рисунок 2.2 – Диаграмма вариантов использования для работы с учетными записями пользователей

Для работы с пользовательскими учетными записями администратор может просматривать список всех пользователей. Для администрирования пользователей так же необходимо просматривать параметры учетной записи.

Для полноценной работы с пользовательскими группами для администратора должны быть возможность просматривать все возможные группы пользователей, возможность создавать новые группы пользователей и удалять

уже существующие группы.

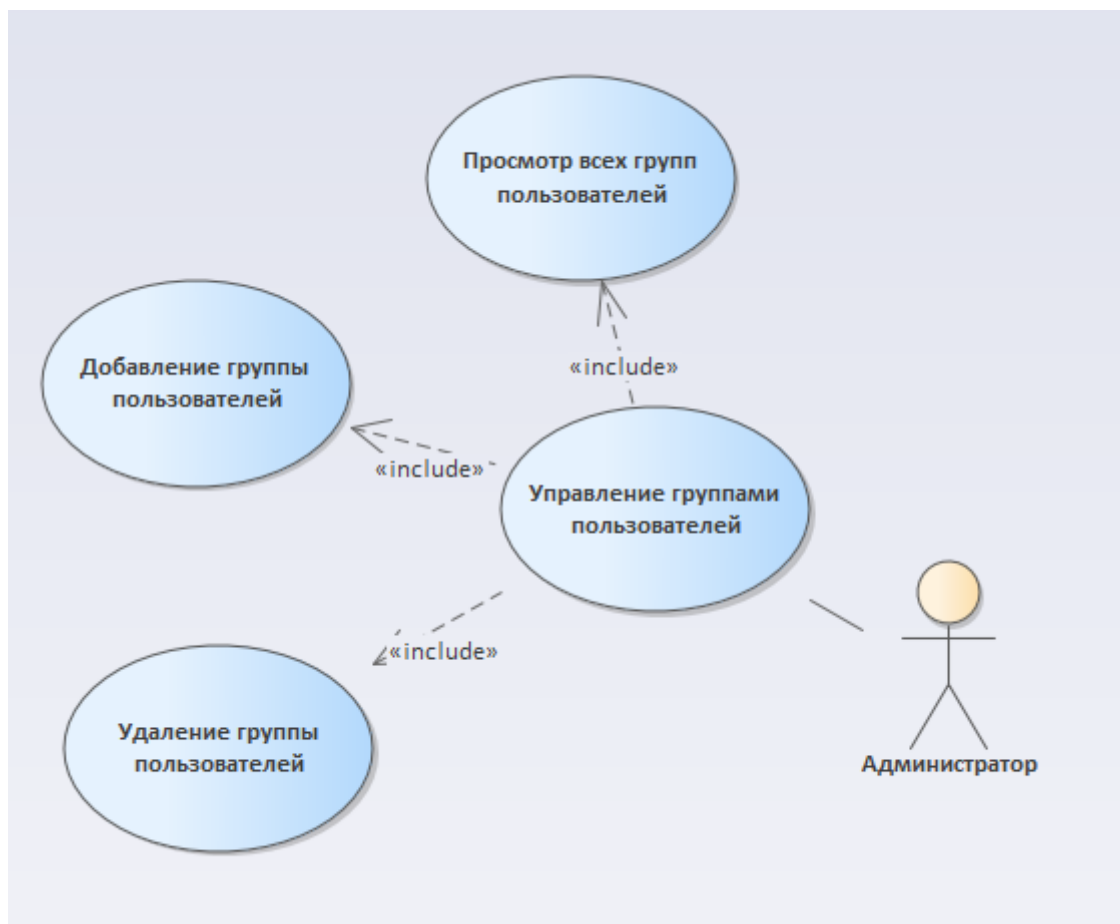


Рисунок 2.3 – Диаграмма вариантов использования для работы с пользовательскими группами

Одной из требований к функциональности для администратора является возможность управления доступными контроллерами. Для реализации управления контроллерами администратор должен иметь возможность добавлять контроллеры и удалять контроллер, просматривать все существующие контроллеры, добавлять контроллеры в группы, а также удалять контроллеры из группы. Для работы так же необходимо отслеживать и изменять состояние контроллера. Администратор должен иметь возможность выбирать состояния. На рисунке 2.4 представлены варианты использования системы для работы с контроллерами

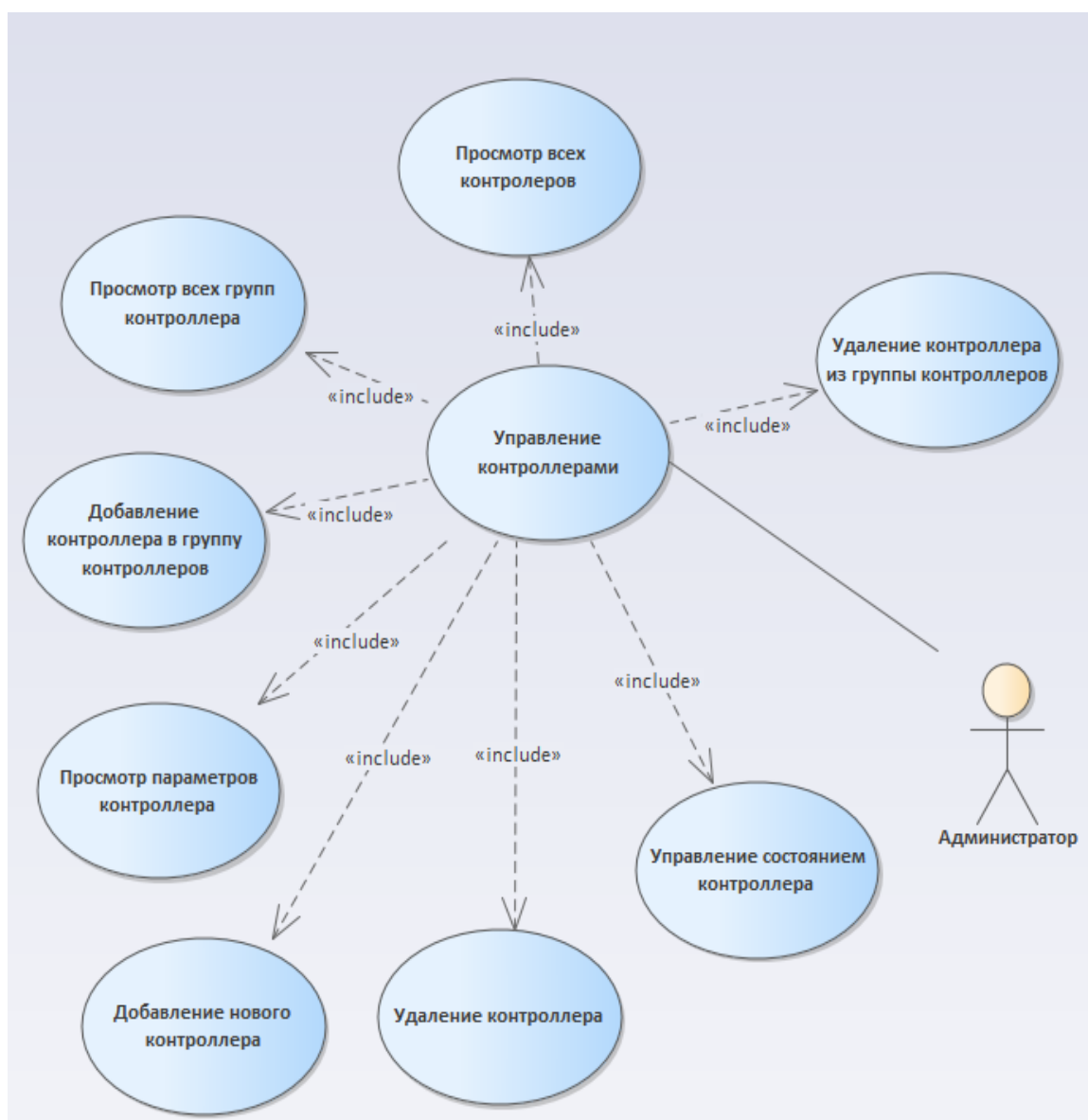


Рисунок 2.4 – Диаграмма вариантов использования для работы с контроллерами

Для получения данных от контроллера администратору необходимо работать с выходами контроллеров, которые представляют собой команды для контроллера. Администратор должен иметь возможность добавлять и удалять команды для каждого контроллера отдельно. Администратор должен объединять выходы контроллеров в группы и удалять выходы из группы. Для работы с выходами контроллера администратору необходимо выбрать контроллер, к которому относиться данный выход. Для корректного отображения данных с выхода контроллера администратору необходимо иметь возможность устанавливать диапазон доступных значений. На рисунке 2.5 представлена диаграмма вариантов использования для работы с выходами контроллеров.

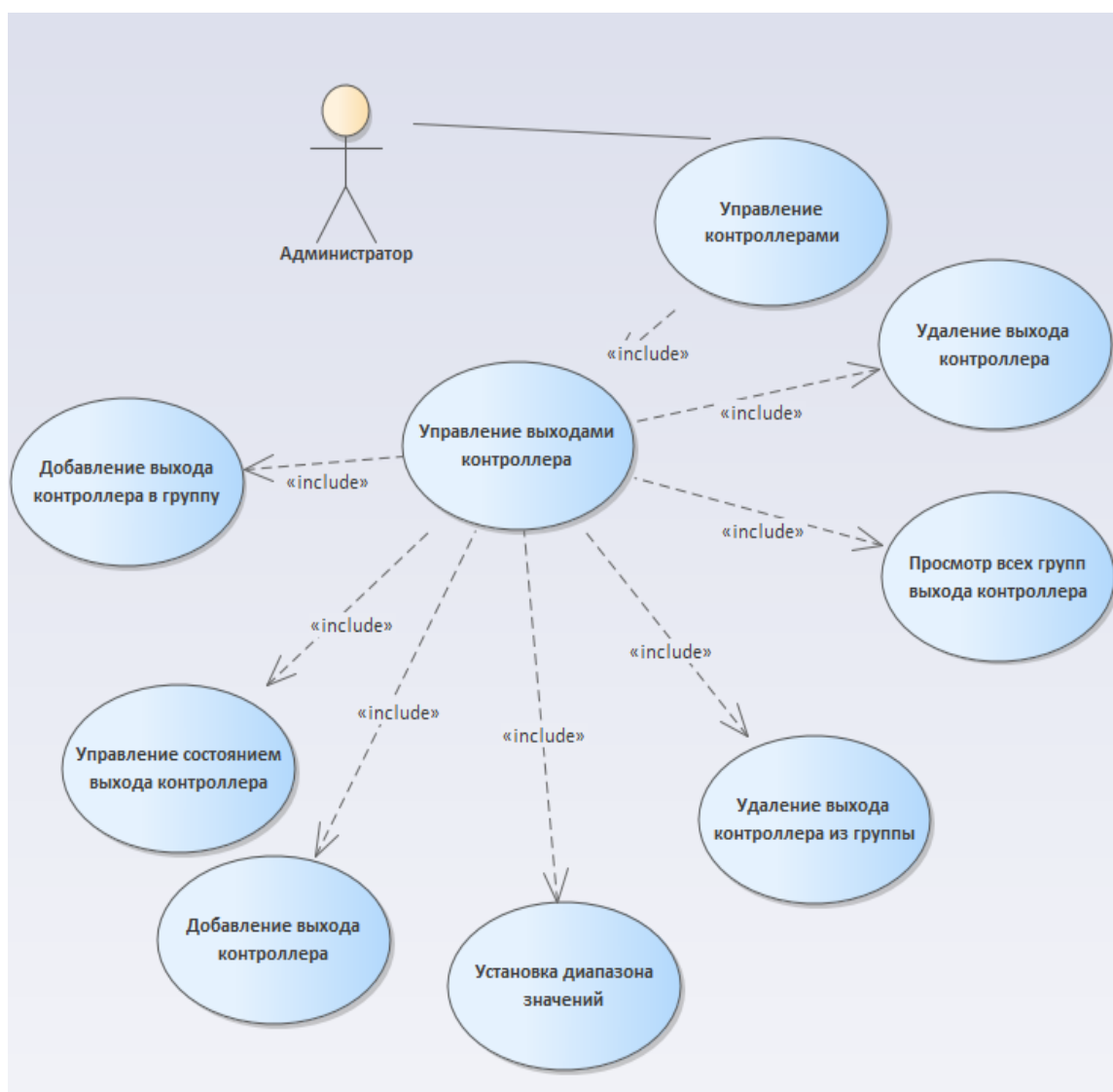


Рисунок 2.5 – Диаграмма вариантов использования для работы с выходами контроллера

Еще одна функциональность, необходимая для корректной работы программного средства, является возможность администратора создавать группы для пользователей, контроллеров и выходов контроллеров. Данный блок позволяет удобно управлять доступом к разным частям приложения для пользователей.

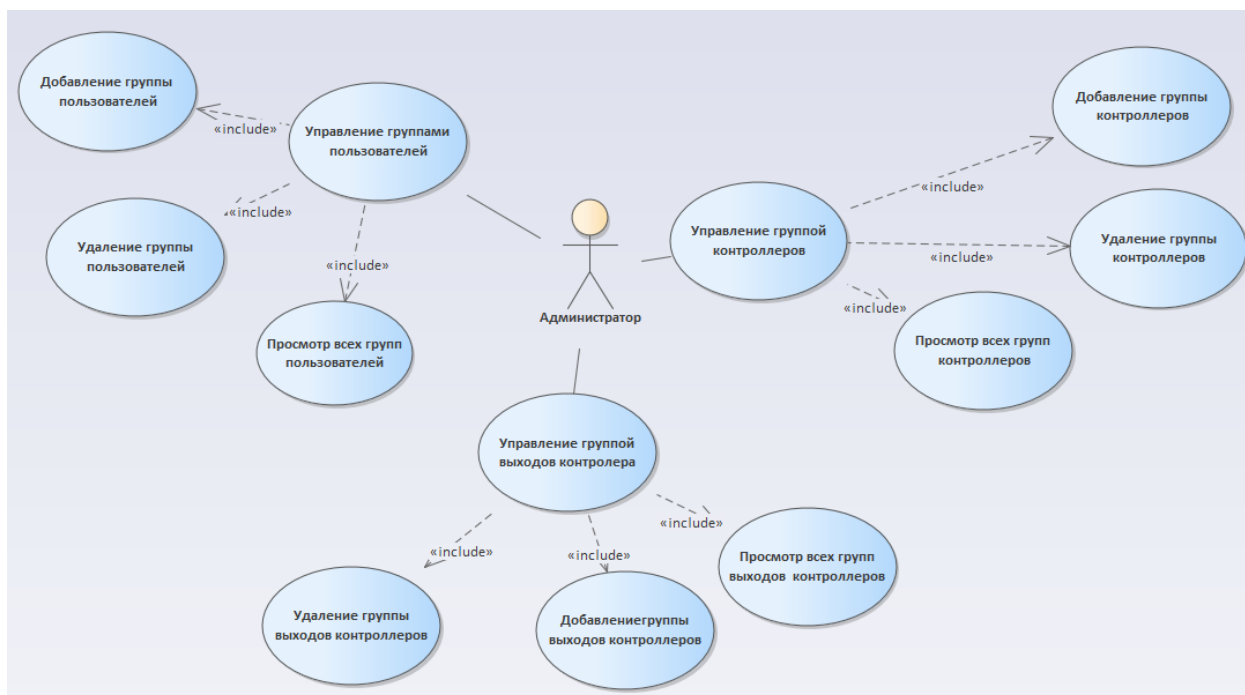


Рисунок 2.6 – Диаграмма вариантов использования для работы с группировкой

На рисунке 2.7 можно увидеть диаграмму вариантов использования в общих чертах. Данная диаграмма представляет упрощенную модель использования системой.

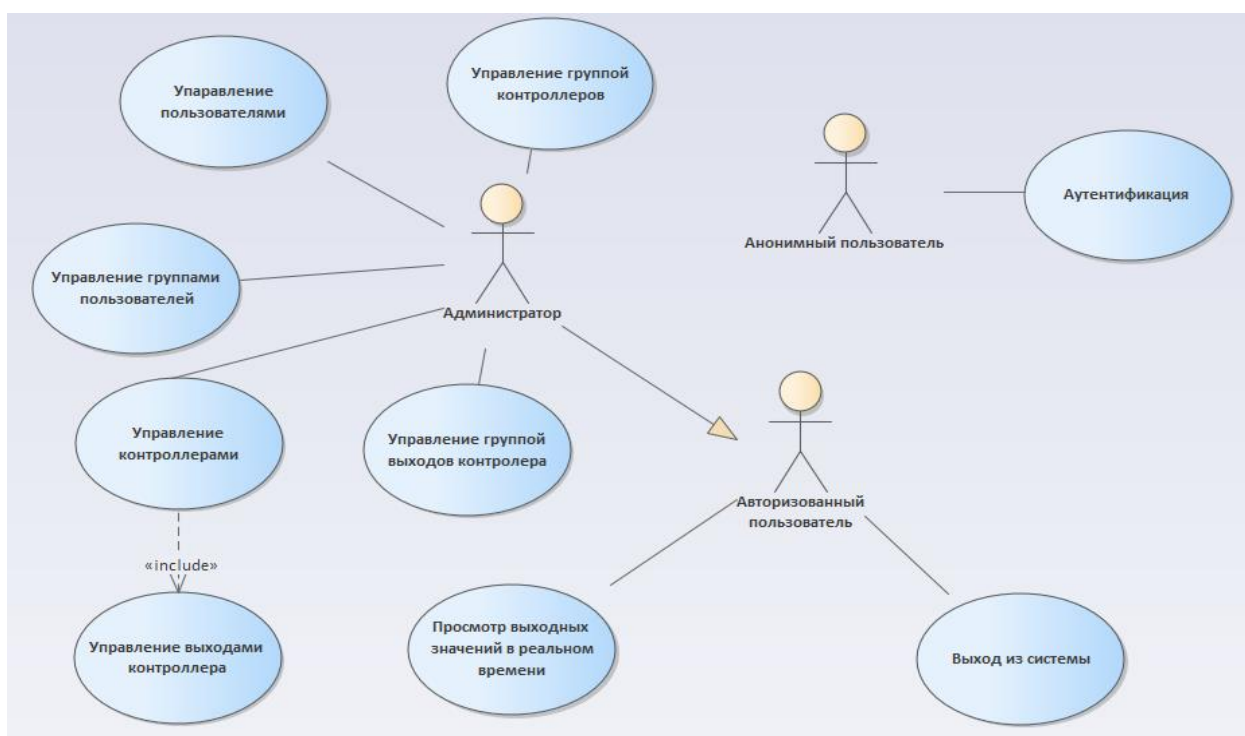


Рисунок 2.7 –Диаграмма вариатов использования системой

2.3 Разработка протоколов связи

Для данного программного обеспечения необходимо работать с промышленными протоколами связи. Для первоочередного протокола связи был выбран FINS – Factory Interface Network Service.

Протокол связи FINS (Factory Interface Network Service) является протоколом, разработанным компанией Omron для обмена данными между устройствами в системах автоматизации производства. Он используется для связи между контроллерами, ПК и другими устройствами в сети.

Протокол FINS поддерживает несколько типов сообщений, включая запросы на чтение и запись данных, запросы на выполнение команд, а также оповещения об изменении состояния устройств. Кроме того, протокол FINS поддерживает защищенную связь с помощью шифрования данных.

Протокол FINS может использоваться для связи с различными устройствами, включая контроллеры Omron, ПК и другие устройства, которые поддерживают этот протокол. Он обеспечивает высокую скорость передачи данных и надежность связи между устройствами.

Протокол FINS является одним из основных протоколов, используемых в системах автоматизации производства, и его использование позволяет улучшить эффективность работы и повысить качество производства.

Протокол FINS поддерживает несколько типов команд, которые могут быть отправлены между устройствами в сети. Некоторые из этих команд включают:

- команда чтения данных: используется для запроса данных из определенного адреса памяти устройства;
- команда записи данных: используется для записи данных в определенный адрес памяти устройства;
- команда выполнения команды: используется для выполнения определенной команды на устройстве;
- команда оповещения об изменении состояния: используется для отправки оповещения другим устройствам в сети об изменении состояния определенного устройства;
- команда проверки связи: используется для проверки связи между устройствами в сети.

Для работы с памятью с помощью FINS запросов нам необходимо знать в какой области данных лежат необходимые для нас данные. ПЛК Omron имеет несколько областей памяти, каждая из которых используется для хранения определенных типов данных и выполнения определенных функций:

- область памяти CIO (Control Input/Output) – используется для хранения входных и выходных сигналов управления. Входные сигналы могут быть физическими входами, такими как кнопки или датчики, а выходные сигналы могут быть реле, контакторы или другие устройства управления;

- область памяти DM (Data Memory) – используется для хранения временных данных, таких как значения переменных, результаты вычислений и другие данные, которые используются в программе управления;
- область памяти EM (Expansion Memory) – используется для расширения памяти DM при необходимости;
- область памяти TR (Timer) – используется для хранения временных данных, необходимых для управления таймерами;
- область памяти CT (Counter) – используется для хранения временных данных, необходимых для управления счетчиками.

Каждая область памяти имеет свой код, адресный диапазон и специальные функции для работы.

Структура FINS команды состоит из следующих частей:

- заголовка сообщения, который содержит информацию об адресе отправителя, адресе получателя, а также содержит необходимые флаги, контрольную сумму и длину всего запроса;
- поле команды – содержит код команды и область памяти;
- поле данных – содержит данные, которые передаются вместе с командой.

2.4 Разработка информационной модели

На основе функциональной модели была разработана информационная модель программного средства. На рисунке 2.2 представлена информационная модель проектируемой системы.

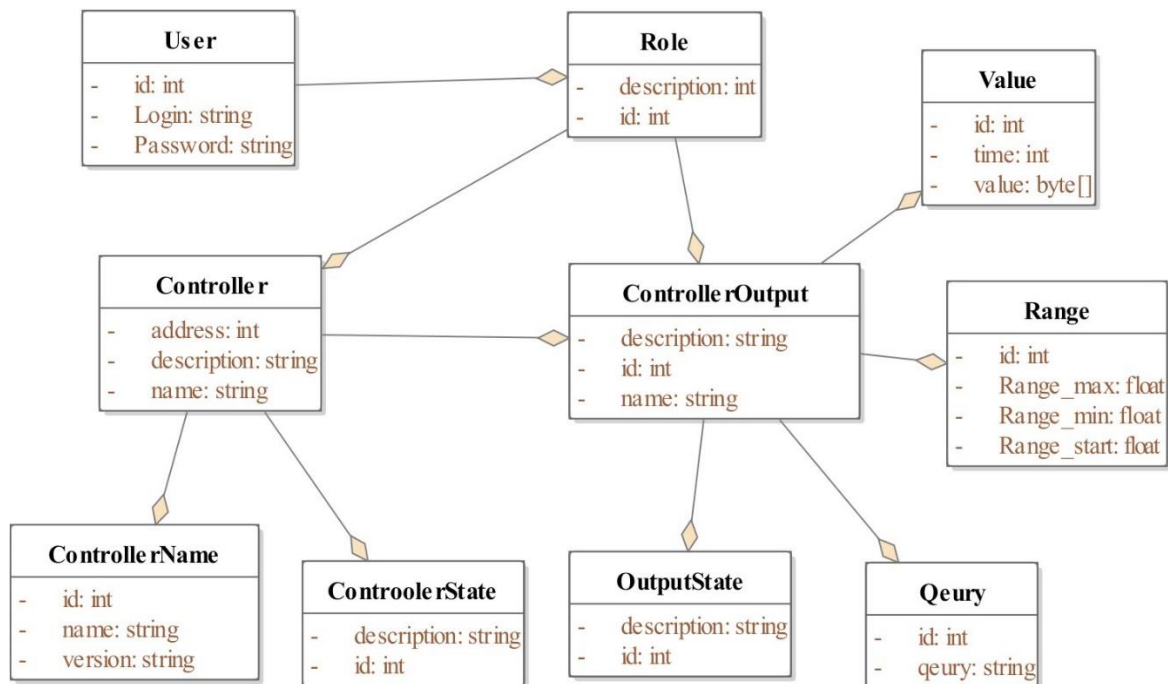


Рисунок 2.2 – Информационная модель программного средства

В процессе анализа предметной области были выделены следующие типы сущностей с атрибутами:

1 User – сущность представляющая собой все типы пользователей системы. Атрибут id – идентификатор пользователя, Login – идентифицирующее имя пользователя, Password – пароль пользователя.

2 Role – сущность содержащая информацию о доступных контроллерах и выходах пользователя, а так же группирующая пользователей. Атрибут id – идентификатор роли, description – информация о роли, удобная для человека.

3 Controller – сущность представляющая контроллер. Атрибут address – идентификатор контроллера, хранящий адрес контроллера, description – описание контроллера, name – имя контроллера.

4 ControllerName – сущность, которая хранит информацию о производителе контроллеров и серии контроллера. Атрибут id – содержит идентификатор сущности, name – содержит информацию о производителе контроллеров, version – содержит серию и версию контроллера.

5 ControllerState – сущность, хранящая информацию о состоянии контроллера. Атрибут id – идентификатор состояния контроллера, description – строковое значение состояния.

6 ControllerOutput – сущность, предоставляющая информацию выходных значениях контроллера. Атрибут id – идентификатор выхода, description – описание выхода, name – название выхода.

7 OutputState – сущность, хранящая информацию о состоянии выхода контроллера. Атрибут id – идентификатор состояния выхода, description – строковое значение состояния.

8 Query – сущность, предоставляющая информацию о запросе к контроллеру. Атрибут id – идентификатор запроса, Query – строковое представление запроса для контроллера.

9 Value – сущность, которая хранит информацию о значениях, полученных от контроллеров. Атрибут id – идентификатор значения, time – информация о времени получения значений, value – значение, полученное от контроллера.

10 Range – сущность, хранящая информацию о диапазоне значений. Атрибут id – идентификатор диапазона, Range_max – максимальное значение диапазона, Range_min – минимальное значение диапазона, Range_start – начальное значение диапазона.

2.5 Разработка спецификации функциональных требований

Исходя из результатов моделирования предметной области можно выделить крупные функциональные возможности, которые можно разделить на следующие группы:

- функциональность управления пользователями;

- функциональность управления выхода контроллера;
- функциональность управления контроллера;
- функциональность просмотра значений, полученных с контроллера;
- функциональность добавления значений;
- функциональность опроса контроллеров;
- функциональность запроса данных у контроллера;
- функциональность по авторизации пользователя.

Функциональность управления контроллера должна предоставлять администратору:

- возможность добавлять контроллер, из поддерживаемых системой;
- возможность удалять контроллер;
- возможность изменять состояние контроллера;
- возможность просмотра доступных состояний;
- возможность выход контроллера;
- возможность группировать контроллеры.

Функциональность управления выхода контроллера должна предоставлять возможность администратору:

- изменять состояние выхода контроллера;
- просмотра доступных состояний выхода контроллера;
- просмотра доступных команд контроллера;
- добавление команды, из доступных, на выход контроллера;
- группировать выходы контроллера в группы;

Функциональность добавления значений должны выполнять сохранение в базе данных значений, полученных с контроллера.

Функциональность просмотра значений должна позволять:

- администратору просматривать все имеющиеся значения;
- пользователю просматривать значений, определенные группой.

Функциональность управления пользователями должна позволять администратору:

- добавлять пользователя;
- группировать пользователей;
- удалять пользователей.

Функциональность по авторизации пользователей должна давать возможности:

- авторизоваться пользователю по паролю и персональному имени;
- авторизоваться в одной учетной записи можно с нескольких устройств одновременно.

Функциональность запроса данных у контроллера должна позволять серверу обращаться к контроллерам Omron при помощи промышленного протокола FINS по сети интернет.

3 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

3.1 Разработка программной архитектуры

После проведения анализа предметной области и формулировки функциональных требований к системе можно определить основные положения организации системы, в которой будет работать разрабатываемое программное средство.

Для разработки данного программного средства была выбрана клиент–серверная архитектура приложения. Данная архитектура программных средств позволяет создавать разнесенные в пространстве системы.

Клиент–серверную архитектуру можно разделить на две части:

- клиентскую;
- серверную.

Клиентская часть предоставляет пользователю удобный интерфейс для работы с данными. Основные задачи клиентской части получать данные от серверной части, оборачивать данные в необходимые стили и отправлять некоторые данные на серверную часть. Взаимодействие клиентской и серверной части происходит путем обмена запросов, в которых передаются необходимые данные.

Серверная часть осуществляет основную обработку данных, отправляет данные клиенту. В данном программном средстве серверная часть должна сохранять данные в базе данных, а также автоматически опрашивать контроллеры, которые находятся в локальной сети.

На рисунке 3.1 представлена диаграмма развертывания программного средства. На диаграмме представлены ключевые элементы, которые необходимы для работы данного программного средства.

Сервер приложения представляет собой устройство, на котором развернуты веб–сервер и серверная часть приложения. В рамках клиент–серверной архитектуры веб–сервер представляет собой посредника между логикой программного средства и внешним миром. Его основная задача: принимать запросы от пользователей и вызывать соответствующие методы обработки данных запросов.

В процессе исследования требований к программному средству было решено реализовывать клиентскую часть в виде SPA(Single Page Application). Данный подход к веб–серверам позволяет обрабатывать действия пользователя без необходимости каждый раз перезагружать страницу. Отражения этого факта на диаграмме заключается в наличии `server.js` файла.

Для хранения данных, необходимых для правильного функционирования программного средства, используется сервер базы данных. Сервер базы данных может быть размещен на устройстве, находящимся в той же локальной сети, так и непосредственно на том же устройстве, что и веб–сервер.

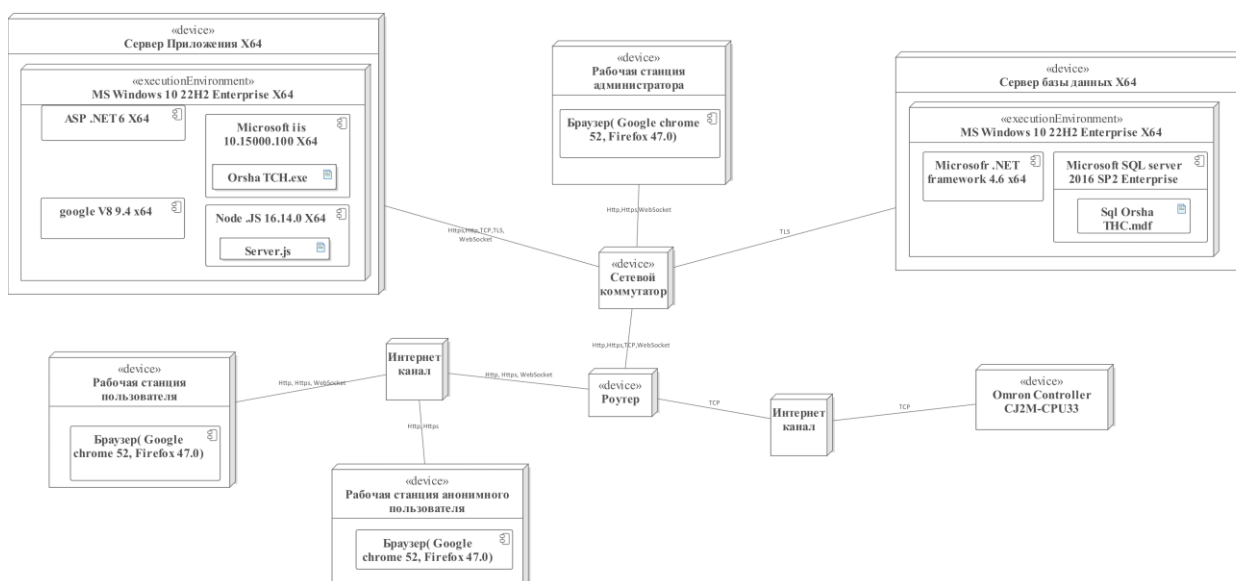


Рисунок 3.1 – Диаграмма развертывания

Важной частью системы являются контроллеры Omron. Для обмена информацией между сервером и контроллером можно использовать протокол связи TCP или UDP. Для разработки данного программного средства было выбрано использовать протокол TCP для обеспечения большей надежности.

3.2 Проектирование архитектуры программного средства

Проектирование программной архитектуры – это процесс создания общего плана структуры программного средства, который позволяет разработчикам легко понимать, как различные компоненты приложения взаимодействуют друг с другом и какие функции они выполняют.

В процессе проектирования программной архитектуры разработчики определяют основные компоненты приложения, такие как модули, классы, функции и объекты, и определяют, как они будут взаимодействовать друг с другом. Они также учитывают требования к производительности, безопасности и масштабируемости приложения.

Для проектирования программной архитектуры используются различные методологии и инструменты, такие как UML (Unified Modeling Language), диаграммы классов, диаграммы последовательностей и диаграммы состояний. Эти инструменты помогают разработчикам создавать понятную и легко читаемую документацию.

В целом, проектирование программной архитектуры является важным этапом разработки программного обеспечения, который позволяет создать структуру приложения, которая легко понимается и поддерживается разработчиками.

Проектирование программной архитектуры стоит начинать как можно раньше. Данный этап создания программного средства позволяет избежать большинства проблем во время разработки. К проблемам относятся:

1 Низкая производительность. Не продуманная архитектура может привести к неэффективному использованию ресурсов, что может снизить производительность программы.

2 Сложность сопровождения. Если архитектура не продумана заранее, то код может быть запутанным и сложным для понимания, что затруднит его сопровождение и поддержку в будущем.

3 Невозможность масштабирования. Не продуманная архитектура может привести к тому, что программа не будет масштабируемой и не сможет поддерживать большое количество пользователей или обрабатывать большие объемы данных.

4 Невозможность расширения. Если архитектура не продумана заранее, то программа может быть неспособной к расширению функционала в будущем, что может привести к необходимости переписывания большей части кода.

5 Низкая надежность. Не продуманная архитектура может привести к ошибкам и сбоям в работе программы, что может повлиять на ее надежность и стабильность.

Модель API Gateway – это модель проектирования, используемая в архитектуре микросервисов для обеспечения единой точки входа для всех клиентских запросов. Он действует как обратный прокси-сервер, который направляет запросы от клиентов к соответствующему микросервису и объединяет ответы от нескольких микросервисов в единый ответ, который возвращается клиенту.

Модель API Gateway обеспечивает ряд преимуществ, включая повышение безопасности за счет централизации аутентификации и авторизации, улучшение масштабируемости за счет разгрузки таких задач, как балансировка нагрузки и кэширование, от отдельных микросервисов, а также повышение гибкости за счет возможности внесения изменений в базовые микросервисы без влияния на клиентов.

API Gateway также может реализовывать дополнительные функции, такие как преобразование запросов/ответов, ограничение скорости и обнаружение сервисов.

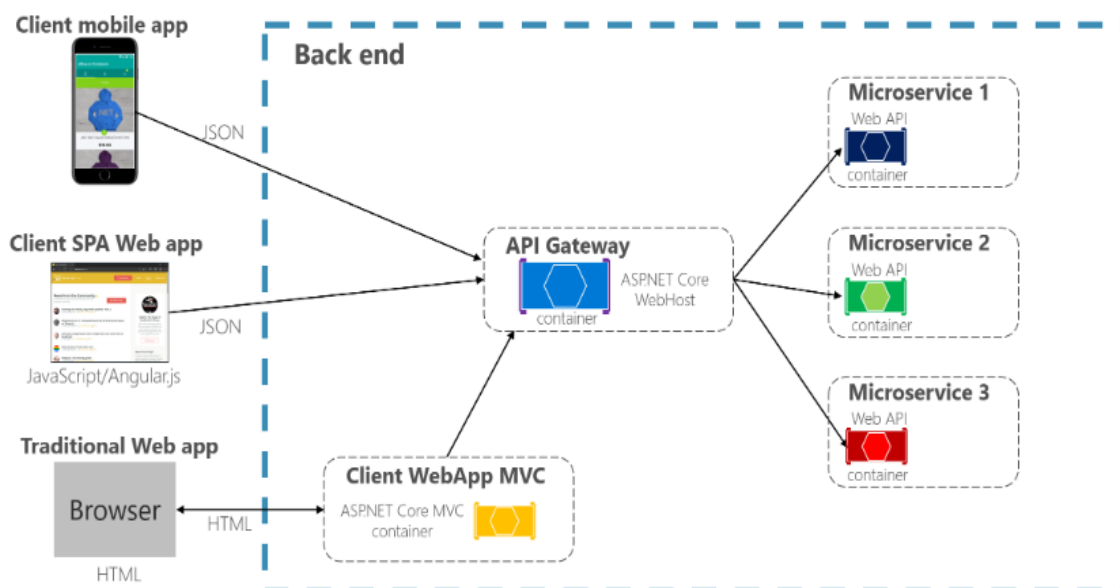


Рисунок 3.2 – API gateway шаблон проектирования.

3.3 Проектирование алгоритма соединения пользователя и контроллера

В процессе проектирования программной архитектуры одной из важнейших частей программного средства было выделено соединения контроллера с пользователем, которое бы позволило передавать большие объемы информации, пришедшие с контроллера, пользователю. Для решения данной задачи был разработан данный алгоритм.

Для реализации правильного соединения пользователя с контроллером нам необходимо объединить пользователи и выходы контроллеров по специальным группам. В ходе проектирования было принято решение, что оптимальным способом уведомления пользовательских контроллеров, о выполненной команде, будет использование функции обратного вызова. Данная функция должна получать результаты запроса и автоматически отправлять их на пользовательское устройство. Алгоритм соединения пользователя и контроллера можно увидеть на рисунке 3.3.



Рисунок 3.3 – Алгоритм соединения пользователя с контроллером

3.4 Проектирование алгоритма добавления новой команды для контроллера

Функциональность добавления новой команды для контроллера является одной из системообразующих. Схема алгоритма, применяемого для реализации данной функциональности приведена на рисунке 3.5.

Из схемы алгоритма можно выделить три основных части данного алгоритма

- ввод пользователем данных о контроллере;
- ввод пользователем типа команды и параметров для данной команды;
- формирование запроса для отправки на контроллер.

Ввод пользователем данных о контроллере начинается с того, что система должна найти все возможные контроллеры доступные для данного пользователя. Контроллеры могут как быть уже объявлены, так и поддерживаться системой. Нам необходимо найти все уже добавленные контроллеры, для формирования списка из названия и версии контроллера. Название и версия контроллера отправляется пользователю для дальнейшего выбора необходимого контроллера.

После ввода пользователем контроллера, с определенным названием и версией. После проверки корректности данных алгоритм должен выбрать поддерживаемые команды на данном контроллере. Структура команд может отличаться в зависимости от реализованного протокола связи. Для решения данной задачи команды должны преобразовываться к строчному виду. После реализации команды передаются пользователю для выборки необходимой команды.

Следующий этап отвечает за корректный вид команд, введенных пользователем. Пользователь должен заполнить необходимые поля и алгоритм преобразует в строковое представление команды к необходимому типу. На данном этапе проверяется корректность команд, если команда не корректна пользователю сообщается об ошибке.

На следующем этапе генерируются основные заголовки для запроса. Каждый контроллер должен содержать обработчик команд, в котором хранятся заголовки запросов. После добавления запроса в обработчик

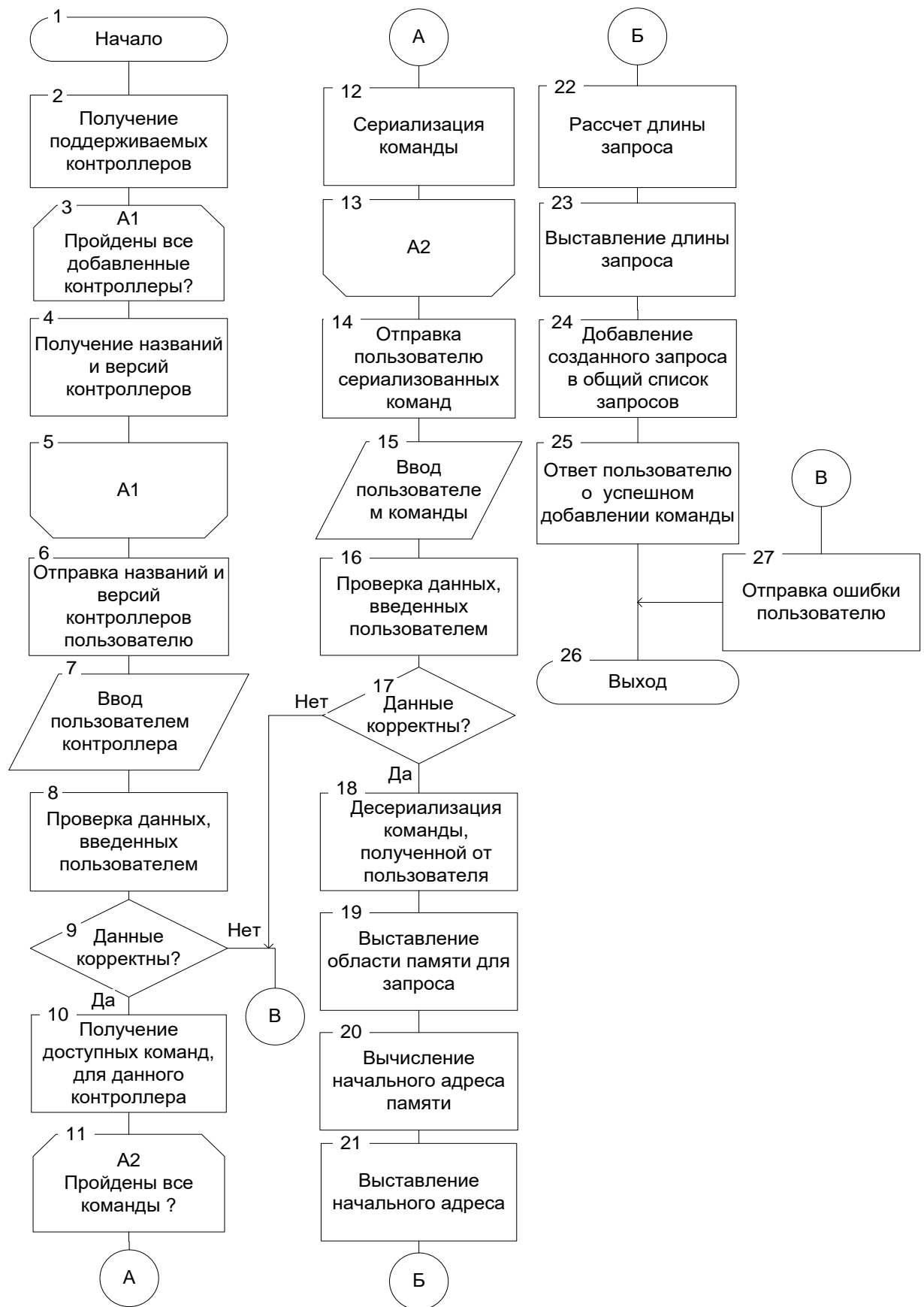


Рисунок 3.5— Алгоритм добавления команды контроллеру

4 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Разработка программного обеспечения начинается на основе требований заказчика, которые были определены на начальных этапах создания проекта. На основе этих требований был разработан план проекта, определен стек технологий и инструментов, были выбраны языки программирования.

4.1 Выбор и обоснование языка и среды разработки программного средства

Данное программное средство можно разделить на две основные части: серверную, которая работает с запросами к контроллерам и информацией, полученной от них, базой данной, в которую сохраняются результаты работы, и отправкой клиенту необходимые данные, и клиентскую часть, которая реализует интерфейс для удобной работы с данными.

.NET 6 – это последняя выпущенная версия платформы .NET с долгой поддержкой, которая была выпущена в ноябре 2021 года. Эта платформа представляет собой современный инструментарий для создания высокопроизводительных приложений для различных платформ и устройств.

Одним из ключевых преимуществ .NET 6 является ее кроссплатформенность. Теперь разработчики могут создавать приложения, которые работают на Windows, Linux и macOS, используя общую кодовую базу. Это значительно упрощает процесс разработки программ.

.NET 6 также обладает высокой производительностью, благодаря оптимизациям в работе с памятью, улучшенной обработке данных и новым инструментам оптимизации кода. Это позволяет создавать приложения, которые работают быстрее и используют меньше ресурсов.

.NET 6 поддерживает различных языков программирования, включая C#, F# и Visual Basic. Для использования в данном программном средстве основным языком разработки был выбран C#. Данный язык имеет следующие качества:

- объектно– ориентированный подход: C# полностью ориентирован на объекты, что делает его более удобным для разработки сложных приложений;
- интеграция с .NET: C# интегрирован с платформой .NET, что облегчает процесс разработки приложения.
- широкая стандартная библиотека: C# имеет обширную стандартную библиотеку, которая предоставляет множество классов и методов для работы с файлами, сетью, базами данных и другими задачами;
- развитая экосистема: C# имеет большое сообщество разработчиков и множество инструментов, библиотек и фреймворков для упрощения процесса разработки.

Для разработки клиентской части программного средства был выбран язык программирования JavaScript со вместо с библиотекой ReactJS.

JavaScript (JS) – это мультипарадигменный язык программирования, который используется для создания интерактивных веб– страниц и веб– приложений. Он является одним из трех основных языков веб– разработки, вместе с HTML и CSS. JS работает на стороне клиента и может быть использован для создания различных функций, таких как анимация, валидация форм, обработка событий и многое другое.

JS имеет множество возможностей и функций, которые позволяют создавать сложные веб– приложения. Он поддерживает объектно– ориентированное программирование, функциональное программирование и асинхронное программирование. JS также предлагает множество инструментов и библиотек, которые упрощают разработку и расширение функциональности приложений.

React – это библиотека JavaScript, которая используется для создания пользовательских интерфейсов веб– приложений. Она была разработана Facebook.

React также использует виртуальный DOM (Document Object Model), который позволяет оптимизировать производительность приложения. Вместо того, чтобы обновлять всю страницу при изменении данных, React обновляет только необходимые элементы, что снижает нагрузку на сервер и ускоряет работу приложения.

4.2 Разработка базы данных

По результатам анализа предметной области была спроектированная физическая модель базы данных. В процессе нормализации информационной модели была создана модель базы данных. Данная база данных содержит следующие таблицы и атрибуты:

Таблица 4.1 – Таблицы базы данных

| Таблица | Поле | Тип данных | Комментарий |
|----------------|------------|--------------|--|
| Users | | | Зарегистрированный пользователь |
| | u_id | Int | Идентификатор пользователя, PK, not null |
| | u_password | varchar(512) | Пароль пользователя, not null |
| | u_login | varchar(50) | Идентификатор пользователя для входа в систему, not null |
| m2m_users_role | | | Сопоставление между пользователем и ролью |

Продолжение таблицы 4.1

| Таблица | Поле | Тип данных | Комментарий |
|-----------------------|----------------|-------------|--|
| | u_id | int | Идентификатор пользователя, PFK, not null |
| | ur_id | int | Идентификатор пользовательской роли, PFK, not null |
| users_role | | | Пользовательские роли |
| | ut_id | int | Идентификатор пользовательской роли, РК, not null |
| | ut_description | varchar(50) | Описание роли |
| m2m_mcg_ut | | | Сопоставление группы контроллеров и ролью |
| | ut_id | int | Идентификатор пользовательской роли, PFK, not null |
| | mc_g_id | int | Идентификатор группы контроллеров, PFK, not null |
| m2m_mcg_ut | | | Сопоставление группы выходов контроллеров и ролью |
| | ut_id | int | Идентификатор пользовательской роли, PFK, not null |
| | mco_g_id | int | Идентификатор группы выходов контроллеров, PFK, not null |
| microcontroller_group | | | Группа контроллеров |
| | mc_g_id | int | Идентификатор группы контроллеров, РК, not null |
| | mc_g_id | varchar(50) | Описание группы |
| m2m_mcg_mc | | | Сопоставление между группой контроллера и контроллером |
| | mc_address | bigint | Идентификатор контроллера, PFK, not null |

Продолжение таблицы 4.1

| Таблица | Поле | Тип данных | Комментарий |
|-----------------------|------------------|--------------|---|
| | mc_g_id | int | Идентификатор группы контроллеров, PFK, not null |
| microcontroller | | | Микроконтроллер |
| | mc_address | bigint | Идентификатор контроллера, РК, not null |
| | mc_description | varchar(100) | Описание контроллера |
| | mc_s_id | int | Идентификатор состояния контроллера, FK, not null |
| | mc_name | varchar(50) | Название контроллера |
| | mcn_id | int | Идентификатор версии контроллера, FK, not null |
| | mc_port | int | Порт для связи с контроллером, not null |
| microcontroller_state | | | Состояние контроллера |
| | mc_s_id | int | Идентификатор состояния контроллера, РК, not null |
| | mc_s_discription | varchar(50) | Описание выхода контроллера |
| microcontroller_name | | | Версия контроллера |
| | mcn_id | int | Идентификатор версии контроллера, РК, not null |
| | mcn_name | varchar(100) | Название производителя контроллера, not null |
| | mcn_version | varchar(50) | Название версии контроллера, not null |
| mco_id | | | Выход контроллера |
| | id | int | Идентификатор выхода контроллера, РК, not null |
| | mc_address | bigint | Идентификатор контроллера, PFK, not null |
| | mco_name | varchar(50) | Имя выхода |
| | mco_description | varchar(50) | Описание выхода |

Продолжение таблицы 4.1

| Таблица | Поле | Тип данных | Комментарий |
|------------------------------|-------------------|--------------|--|
| | mcov_range_id | int | Идентификатор диапазона, FK, |
| | mco_q_id | int | Идентификатор запроса, FK |
| | mco_s_id | int | Идентификатор состояния выхода, FK, not null |
| microcontroller_output_group | | | Группа выходов контроллеров |
| | mco_g_id | Int | Идентификатор группы выходов, PK, not null |
| | mco_g_description | varchar(50) | Описание группы выходов |
| m2m_mco_mcog | | | Сопоставление между группой выходов контроллера и выходами контроллера |
| | mco_g_id | Int | Идентификатор группы выходов, PFK, not null |
| | Id | Int | Идентификатор выхода контроллера, PFK, not null |
| | mc_address | bigint | Идентификатор контроллера, PFK, not null |
| mco_query | | | Запрос к контроллеру |
| | mco_query_request | varchar(300) | Описание запроса, not null |
| | mco_q_id | Int | Идентификатор запроса, PK, not null |
| microcontroller_output_state | | | Состояние выхода контроллера |
| | mco_s_id | int | Идентификатор состояния выхода контроллера, PK, not null |
| | mco_s_description | varchar(50) | Описание выхода контроллера |

Продолжение таблицы 4.1

| Таблица | Поле | Тип данных | Комментарий |
|------------|----------------------|--------------|---|
| mco_v | | | Значение выхода контроллера |
| | mco_v_time | datetime | Время получения значения, РК, not null |
| | mco_id | int | Идентификатор выхода контроллера, PFK, not null |
| | mc_addres | bigint | Идентификатор контроллера, PFK, not null |
| | mco_v_value | varbinary(4) | Значение выхода |
| mcov_range | | | Диапазон значений |
| | mcov_range_id | int | Идентификатор диапазона значений, РК, not null |
| | mcov_range_min_value | real | Минимальное значение диапазона, not null |
| | mcov_range_max_value | real | Максимальное значение диапазона, not null |
| | mcov_range_start | real | Начальное значение диапазона, not null |



Рисунок 4.2 – Физическая модель базы данных

4.3 Разработка программной архитектуры

Для облегчения понимания программной архитектуры была разработана диаграмма классов. На рисунке 4.1 показаны основные классы, которые необходимы для реализации соединения сервера программного средства с контроллерами в сети интернет.

5 ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Таблица 5.1 – Результаты тестирования приложения

| Тестовая ситуация | Ожидаемый результат | Полученный результат |
|--|---|---|
| 1 Авторизация администратора | Переход на страницу администратора, изменение меню администратора | Переход на страницу администратора, изменение меню администратора |
| 2 Авторизация пользователя | Переход на страницу пользователя, изменение меню пользователя | Переход на страницу пользователя, изменение меню пользователя |
| 3 Выход из системы | Переход на начальную страницу | Переход на начальную страницу |
| 4 Просмотра пользователей | Открылась страница пользователей | Открылась страница пользователей |
| 5 Просмотр доступных пользовательских групп | Отображение списка доступных групп для добавления для пользователя | Список доступных групп для пользователя |
| 6 Добавление пользователя в группу | Добавление пользователя в группу, обновление таблицы с пользователями | Пользователь добавлен в группу, обновилась таблица с группами |
| 7 Удаление пользователя из группы | Удаление пользователя из группы, обновление таблицы с пользователями | Пользователь удален из группы, таблица с пользователями удалась |
| 8 Удаление пользователя из группы администраторов | Удаление пользователя из группы администраторов, обновление таблицы пользователей | Пользователь удален из группы администраторов, таблица с пользователями удалась |
| 9 Удаление последнего пользователя из группы администраторов | Возврат ошибки, о невозможности удалить пользователя из группы | Возврат ошибки, о невозможности удалить пользователя из группы |
| 10 Удаление администратора | Администратор не удален | Администратор не удален |
| 11 Просмотр пользовательских групп | Отображение списка всех пользовательских групп | Отображение списка всех пользовательских групп |

Продолжение таблицы 5.1

| Тестовая ситуация | Ожидаемый результат | Полученный результат |
|------------------------------------|---|---|
| 12 Добавление группы пользователей | Добавление группы пользователей, обновление таблицы групп | Добавление группы пользователей, обновление таблицы групп |
| 13 Удаление группы пользователей | Удаление группы пользователей, обновление таблицы групп | Удаление группы пользователей, обновление таблицы групп |
| 14 Добавление контроллера | Добавление контроллера, обновление таблицы контроллеров | Добавление контроллера, обновление таблицы контроллеров |
| 15 Удаление контроллера | Удаление контроллера, обновление таблицы контроллеров | Удаление контроллера, обновление таблицы контроллеров |

По результатам тестирования можно сделать вывод, что разработанное программное средство полностью работоспособно и удовлетворяет всем функциональным требованиям.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

6.1 Авторизация пользователя

При обращении к системе пользователя, не прошедшего авторизацию, будет отображено форма для входа в систему (рисунок 6.1).

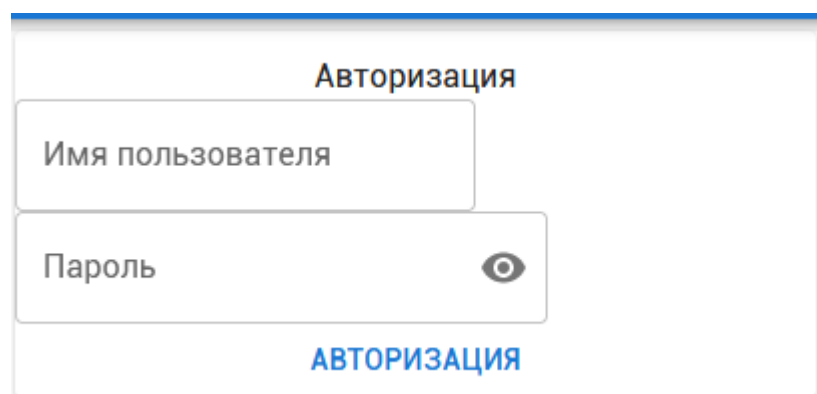
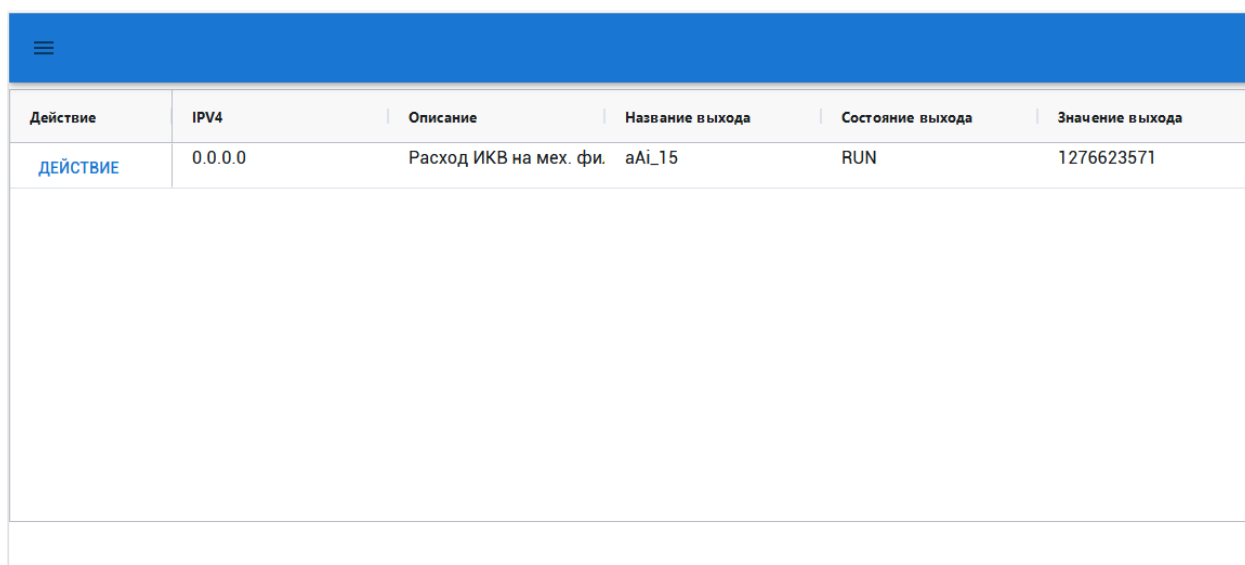


Рисунок 6.1 – Форма входа в систему

Для входа в систему пользователю необходимо ввести имя пользователя и пароль, полученные от администратора.

6.2 Просмотр данных с контроллера пользователем

При входе с помощью пользовательского аккаунта будет отображена страница с доступными для показа пользователю значениями (рисунок 6.2).



| Действие | IPV4 | Описание | Название выхода | Состояние выхода | Значение выхода |
|----------|---------|------------------------|-----------------|------------------|-----------------|
| ДЕЙСТВИЕ | 0.0.0.0 | Расход ИКВ на мех. фи. | aAi_15 | RUN | 1276623571 |

Рисунок 6.2 – Страница авторизованного пользователя

Авторизованный пользователь может вывести данные, полученные от сервера, с помощью графика. Для этого ему необходимо выбрать пункт «Действие» → «Вывести график» для интересующего его параметра. На рисунке 6.3 представлен пример графика.

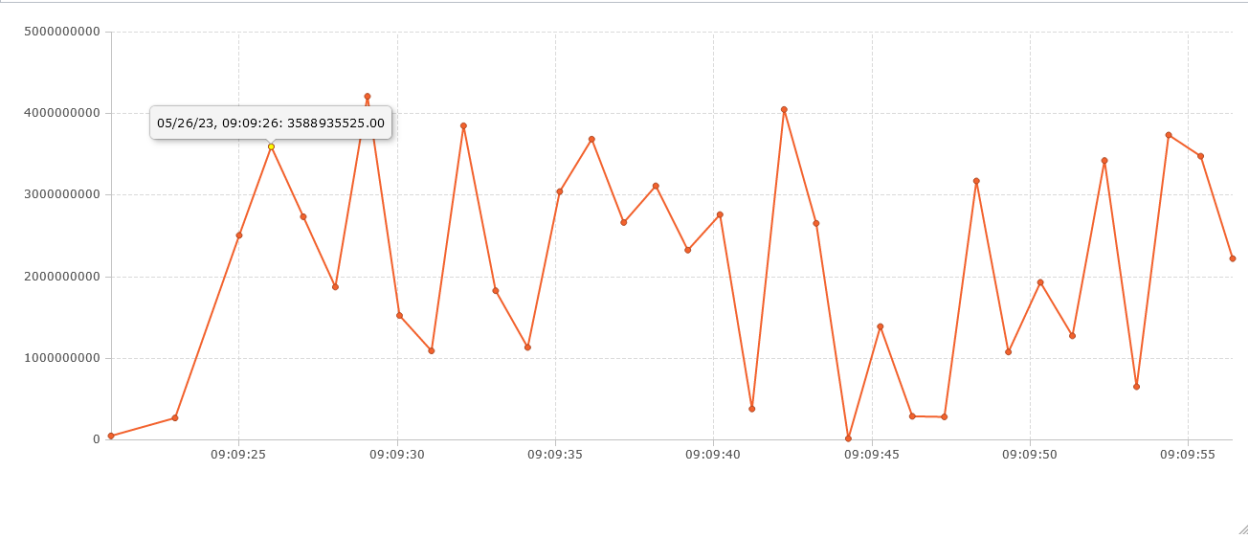


Рисунок 6.3 – График параметра

6.3 Управление выходами контроллера администратором

После входа в систему при помощи аккаунта администратора будет отображены все выходы, отслеживаемые системой. В данном списке администратор сможет изменять состояние выхода, а также администратор может работать с данными, как и пользователь.

| Действие | IPv4 | Описание | Название выхода | Состояние выхода | Значение выхода |
|----------------|---------|------------------------------|-----------------|------------------|-----------------|
| ДЕЙСТВИЕ | 0.0.0.0 | string | string | STOP ▾ | |
| ДЕЙСТВИЕ | 0.0.0.0 | a | a | STOP ▾ | |
| ДЕЙСТВИЕ | 0.0.0.0 | Расход ИКВ на мех. фильтры т | aAi_15 | RUN ▾ | 2164591050 |
| ВЫВЕСТИ ГРАФИК | | | | | |

Рисунок 6.4 – Начальная страница администратора

6.4 Работа с возможностями администратора

Администратор обладает рядом возможностей, которые отличают его среди остальных пользователей системы. Администратор может добавлять новых пользователей в систему и изменять роли пользователей. Для работы с пользователями администратор должен открыть боковую панель и выбрать пункт «Пользователи».

| Действие | ID | Логин | Пароль | Роль |
|----------|----|-------|--------|-------|
| ДЕЙСТВИЕ | 1 | Admin | Admin | ADMIN |
| ДЕЙСТВИЕ | 4 | User | User | USER |

ДОБАВИТЬ РОЛЬ

УДАЛИТЬ ПОЛЬЗОВАТЕЛЯ

Имя пользователя

Пароль

роль

ПОДТВЕРДИТЬ

Рисунок 6.5 – Работа с пользователями

Для добавления нового пользователя администратор должен ввести логин и пароль нового пользователя, он также может выбрать необходимые группы пользователей при создании. Администратор может изменять созданных пользователей с помощью выпадающего меню.

Для добавления новой роли администратор должен выбрать пункт Действие для интересующего пользователя и перейти в подпункт «Добавить роль». При добавлении роли будет выведен список всех доступных для данного пользователя ролей.

Администратор может удалить пользователя с помощью пункта «Действие» → «Удалить пользователя». Для удаления пользователя из группы достаточно нажать на выбранную роль для удаления и в выпадающем меню выбрать пункт «Удалить роль». Администратор не может создать такой ситуации, когда не один пользователь не будет иметь роль Admin.

Администратор может контролировать роли пользователей. Для возможности создания и изменения пользовательских ролей необходимо перейти в боковое меню и выбрать пункт «Роли пользователей». После перехода по данной ссылке откроется страница с пользовательскими ролями показанная на рисунке 6.6.

| Все роли | | | | |
|----------|------|----------|-----------------|---------------------------------------|
| Действие | ID | Описание | Группы контр... | Группы выходов |
| ДЕЙСТВИЕ | 1 | Admin | Группы контр... | Группы выходов string × Выберит... |
| ДЕЙСТВИЕ | 1002 | User | Группы контр... | Группы выходов a × Выберите гр... |

Добавить роль

Название роли

ПОДТВЕРДИТЬ

Рисунок 6.6 – Страница изменения пользовательских ролей.

Для добавления роли администратору необходимо ввести желаемое название роли в форму и нажать на кнопку «Подтвердить». Данная роль добавит в список всех ролей после успешного добавления.

Пользовательские роли можно объединять с группами контроллеров и группами выходов. Для добавления группы выходов к роли необходимо выбрать соответствующую роль и в столбце «Группы выходов» открыть выпадающее меню. В данном меню можно искать группу выходов вводом имени данной группы. После нажатия на необходимую группу, она будет добавлена в список доступных групп для данной роли. Для удаления группы необходимо повторно нажать на группу. Для предоставления доступа к группам контроллеров алгоритм работы аналогичен.

Для удаления пользовательской роли, необходимо выбрать соответствующую роль и выбрать пункт «Действие» → »Удалить роль».

Основной возможностью администратора является возможность добавления новых контроллеров и выходов для них. Для начала работы с контроллерами необходимо открыть боковое меню и выбрать пункт «Контроллеры». После выполнения данного действия откроется окно работы с контроллерами.

| Действие | IPV4 | Описание | Название контроллера... | Состояние | Группы контроллеров... |
|--------------------------|---------|----------------|-------------------------|-----------|--------------------------------------|
| ДЕЙСТВИЕ | 0.0.0.0 | контроллер XBO | OMRON CJ2M-CPU3 | RUN | Группы выходов So... Выбери... |

Добавить контроллер

Адрес

Имя

Описание

Производитель контроллера

OMRON

Версия контроллера

Рисунок 6.7 – Страница управлением контроллерами

На данной странице есть возможность просмотреть все возможные контроллеры и их состояния. Контроллеры можно добавлять в группы контроллеров и удалять из них.

Для создания контроллера на странице присутствует форма, показанная на рисунке 6.8.

Добавить контроллер

Адрес

Имя

Описание

Производитель контроллера

OMRON

Версия контроллера

CJ2M-CPU33

Состояние контролл...

Группы

ДОБАВИТЬ КОНТРОЛЛЕР

Рисунок 6.8 – Форма добавления контроллера

Для добавления контроллера необходимо ввести ip адрес контроллера, в виде четырех десятичных чисел с точкой между ними. Также необходимо ввести в форму имя контроллера и его описание. Обязательным полем для ввода является «Производитель контроллера», в данном поле необходимо выбрать контроллер, поддерживаемый системой. Следующей пункт «Версия контроллера» тоже является обязательной, данный пункт выбирается на основании выбора «Производителя контроллера». Для возможности добавить контроллер необходимо заполнить поле «Состояние контроллера». «Группы» является не обязательным полем для заполнения. Для фиксации нового контроллера необходимо нажать на кнопку «Добавить контроллер».

При необходимости удалить контроллер необходимо выбрать пункт

«Действие» → «Удалить контроллер» в соответствии с выбранным контроллером.

Для управления выходами контроллера необходимо выбрать пункт «Действие» → «Просмотр запросов». Данная страница предоставляет информацию о выходах контроллера.

Выходы контроллера: 0.0.0.0

| Действие | ID | Описание выхода | Название выхода | Состояние выхода | Группы выходов |
|----------|----|----------------------|-----------------|------------------|--|
| ДЕЙСТВИЕ | 1 | string | string | STOP ▾ | Группы выходов string × string × ▾ a × B... |
| ДЕЙСТВИЕ | 2 | a | a | STOP ▾ | Группы выходов string × ▾ Выберите... |
| ДЕЙСТВИЕ | 3 | Расход ИКВ на мех. ф | aAi_15 | STOP ▾ | Группы выходов a × B... |

Добавить новый выход

Название выхода

Описание выхода

Состояние контролл. ▾

Группы выходов ▾

ПОДТВЕРДИТЬ

Добавить группу выходов

Название группы

ПОДТВЕРДИТЬ

Рисунок 6.9 – Рисунок страницы выходов контроллера

На данной страницы администратор может добавить новый выход для контроллера. Для этого необходимо заполнить форму «Добавить новый выход». В данной форме можно ввести название и описание выхода. Необходимо выбрать начальное состояние выхода контроллера. Есть возможность добавить выход к группам во время создания.

При необходимости есть возможность создать новую группу выходов. Для этого необходимо заполнить название в форме «Добавить группу выходов» и подтвердить действие.

На данной странице есть возможность добавить выход в группу или удалить выход из группы. Для выполнения данных действий необходимо выбрать соответствующий выход и в пункте «Группы выходов» изменять настройки выхода.

Для изменения состояния выхода необходимо определить необходимый

выход контроллера и в пункте «Состояние контроллера» есть возможность выбрать необходимое состояние выхода.

Для удаления выхода контроллера необходимо выбрать нужный выход контроллера и в пункте « Действие» → « Удалить выход» .

Для просмотра команды, выполняемой на выходе контроллера, необходимо выбрать выход контроллера и пункт «Действие» → «Изменить запрос» . При выполнении данного действия откроется диалоговое окно с информацией о запросе (рисунок 6.10)

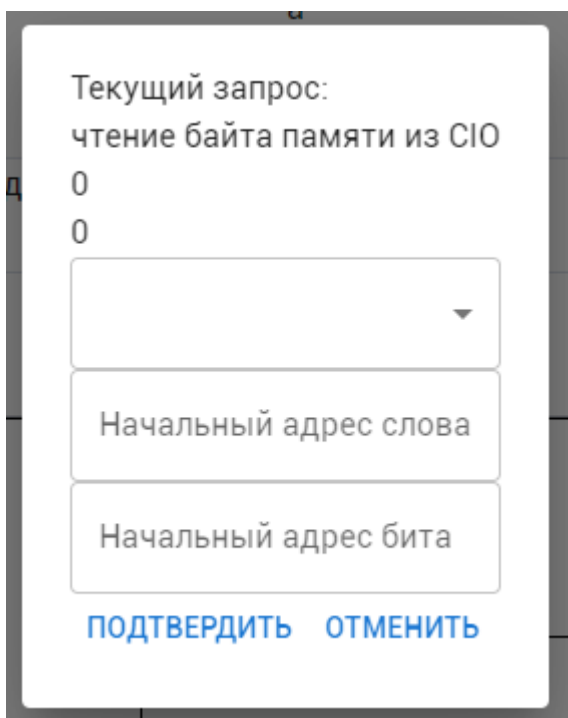


Рисунок 6.10 – Окно с информацией о запросе

В данном окне моно не только просматривать выполняемую команду, но и изменять ее. Для изменения команды необходимо выбрать команду, какая будет выполнять, а также начальный адрес и сдвиг.

7 ТЕХНИКО– ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ПРОГРАММНОГО СРЕДСТВА СБОРА И ВИЗУАЛИЗАЦИИ ТЕХНОЛОГИЧЕСКИХ ПАРАМЕТРОВ ХИМИЧЕСКОГО ЦЕХА НА ПЛАТФОРМЕ .NET

7.1 Описание функций, назначения и потенциальных пользователей ПО

Данное программное средство предоставляет возможность сбора, хранения и визуализации технологических параметров химического цеха. Данное программное средство позволит упростит контроль за технологическими параметрами.

Потенциальными пользователями данного продукта являются компании использующие на производстве контроллеры OMRON Corporation.

Данное программное средства имеет следующие преимущества:

- возможность гибкого расширения функциональности;
- возможность просмотра параметров через веб– браузер.

7.2 Расчет затрат на разработку программного продукта

Для разработки данного программного продукта нам понадобится следующие специалисты: инженер– программист, проектный менеджер, инженер отдела качества. Предполагаемое время работы инженера– программиста составляет 300 часов, дизайнер – 20 часов. проектного менеджера – 50 часов, инженера отдела качества – 70.

Для расчета основной заработной платы использовалась формула 7.1

$$Z_o = K_{\text{пр}} \sum_{i=1}^n Z_{\text{ч.}i} t_i \quad (7.1)$$

где n – количество работников, занятых выполнением работы;

$K_{\text{пр}}$ – коэффициент, учитывающий процент премий;

$Z_{\text{ч.}i}$ – часовая заработная плата i – го исполнителя, р.;

t_i – трудоёмкость работ, выполняемых i – м исполнителем, ч.

Примем количество рабочих часов в месяце равным 168 часам. Расчет затрат на основную заработную плату разработчикам приведен в таблице 7.1. Оклады приведены исходя из сложившегося на рынке труда размера заработной платы данной категории специалистов.

Таблица 7.1 – Расчет затрат на основную заработную плату разработчиков

| Наименование должности разработчика | Вид выполняемой работы | Месячная заработная плата, р. | Часовая заработная плата, р. | Трудоёмкость работ, ч | Сумма, р. |
|--|--|-------------------------------|------------------------------|-----------------------|-----------|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1. Инженер–программист | Разработка логики программного средства, кодирование программы | 2400 | 14,3 | 300 | 4290 |
| 2. Дизайнер | Создание шаблона пользовательского интерфейса | 2200 | 13,1 | 20 | 262 |
| 3. Специалист по тестированию программного обеспечения | Ручное и автоматизированное тестирование программного средства | 1800 | 10,7 | 70 | 749 |
| 4. Менеджер по информационным технологиям | Формирование требований и сроков разработки | 2700 | 16.1 | 50 | 805 |
| Итого | | | | | 6106 |
| Премия (25%) | | | | | 1526.5 |
| Всего основная заработная плата | | | | | 7632.5 |

Расчет затрат на дополнительную заработную плату разработчиков осуществляется по формуле:

$$З_{д} = \frac{З_{о} \cdot Н_{д}}{100\%}, \quad (7.2)$$

где $З_{о}$ – затраты на основную заработную плату, р.;

$Н_{д}$ – норматив дополнительной заработной платы ($Н_{д} = 10\%$).

При нормативе дополнительной заработной платы равном 10%, получаем:

$$З_{д} = \frac{7632,5 \cdot 10}{100} = 763,25$$

Отчисления на социальные нужды определяются в соответствии с действующими законодательными актами по формуле:

$$P_{\text{соц}} = \frac{(Z_o + Z_d) \cdot H_{\text{соц}}}{100}, \quad (7.3)$$

где $H_{\text{соц}}$ – норматив отчислений от фонда оплаты труда ($H_{\text{соц}} = 34,6\%$).
В результате расчетов получаем следующее:

$$P_{\text{соц}} = \frac{(7632,5 + 763,25) \cdot 34,6}{100} = 2904,93$$

Расчет прочих затрат осуществляется по формуле:

$$P_{\text{пз}} = \frac{Z_o \cdot H_{\text{пз}}}{100}, \quad (7.4)$$

где $H_{\text{пз}}$ – норматив прочих затрат ($H_{\text{пз}} = 110\%$).
При нормативе прочих затрат равном 110%, получаем:

$$P_{\text{пр}} = \frac{7632,5 \cdot 110}{100} = 8395,75$$

Полная сумма затрат на разработку программного обеспечения находится путём суммирования всех рассчитанных статей затрат. Результаты расчетов представлены в таблице 7.2.

Таблица 7.2 – Затраты на разработку программного обеспечения

| Наименование статьи затрат | Значение, р. |
|--|--------------|
| 1. Основная заработная плата разработчиков | 7632,5 |
| 2. Дополнительная заработная плата разработчиков | 763,25 |
| 3. Отчисления на социальные нужды | 2904,93 |
| 4. Прочие затраты | 8395,75 |
| Общая сумма инвестиций в разработку | 19696,43 |

7.3 Оценка результата (эффекта) от использования ПО

Экономический эффект организации– разработчика программного обеспечения представляет собой прибыль от его продажи множеству потребителей.

Прибыль, полученная разработчиком от реализации ПО на рынке.

$$\Pi = C \cdot N - \text{НДС} - I_{\text{разр}}, \quad (7.5)$$

где C – цена реализации ПО заказчику, р.;

N – ожидаемое количество копий (лицензий) программного обеспечения, которое будет приобретено пользователями;

НДС – сумма налога на добавленную стоимость, р;

$I_{\text{разр}}$ – затраты на разработку программного обеспечения, р.

Если организация является плательщиком налога на прибыль, то экономический эффект рассчитывается по формуле:

$$\Pi_{\text{ч}} = \Pi \cdot \left(1 - \frac{H_{\text{п}}}{100}\right), \quad (7.6)$$

где $H_{\text{п}}$ – ставка налога на прибыль, согласно действующему законодательству, ($H_{\text{п}} = 20\%$).

Налог на добавленную стоимость определяется по формуле:

$$\text{НДС} = \frac{C \cdot N \cdot H_{\text{дс}}}{100 + H_{\text{дс}}}, \quad (7.7)$$

где $H_{\text{дс}}$ – ставка налога на добавленную стоимость согласно действующему законодательству, ($H_{\text{дс}} = 20\%$).

Программное обеспечение разрабатывается для распространения на информационном рынке. По данным Omron corporation за 2021 год 5000 компаний купили решения для автоматизации технологических процессов на основе их контроллеров. Минимальная цена на программное обеспечение составляет 2783 рублей. Возьмем пессимистичную оценку количества проданных лицензий, которая составляет 25 единиц. Установим цену реализации 2000 рублей.

Налог на добавочную стоимость будет составлять:

$$\text{НДС} = \frac{2000 \cdot 25 \cdot 20}{100 + 20} = 8333,33$$

$$\Pi = 2000 \cdot 25 - 8333,33 - 19696,43 = 21970,24$$

С учетом налога на прибыль:

$$\Pi_{\text{ч}} = 21970,24 \cdot \left(1 - \frac{20}{100}\right) = 17576,192$$

Уровень рентабельности рассчитывается по формуле:

$$y_p = \frac{\Pi(\Pi_{\text{ч}})}{I_{\text{разр}}} \cdot 100, \quad (7.8)$$

где Π ($\Pi_{\text{ч}}$) – чистая прибыль, полученная от реализации ПО на рынке, р;

$I_{\text{разр}}$ – затраты на разработку программного обеспечения, р.

Подставив значения в формулу, получим:

$$y_p = \frac{17576,192}{19696,43} \cdot 100 = 89\%$$

Итоговые затраты на разработку ПО составят 19696,43 р., чистая прибыль составит 17576,192 р., рентабельность составляет 89%.

Проект является экономически эффективным, поскольку рентабельность затрат на разработку ПО существенно выше процентной ставки по банковским депозитным вкладам.

ЗАКЛЮЧЕНИЕ

В результате выполнения дипломного проекта была изучена предметная область, проведен анализ существующих аналогов, было разработано программное средство сбора и визуализации технологических параметров химического цеха на платформе .NET.

В процессе изучения предметной области были изучены промышленные контроллеры Omron, был изучен протокол связи FINS и способы связи контроллеров и прикладных программных средств по сети Ethernet.

В ходе моделирования предметной области была разработана функциональная и информационная модели программного обеспечения. На основе разработанных моделей сформулирована функциональная спецификация к программному обеспечению.

Исходя из полученных на этапе моделирования данных, была спроектирована архитектура программного решения, которая смогла бы позволить точно реализовать требования, указанные в функциональной спецификации. Также на данном этапе были определены архитектурные принципы, в соответствии с которыми должна производиться разработка программного решения.

Весомую часть работы над дипломным проектом заняло изучение протоколов связи, а также моделирование архитектуры программного средства.

Таким образом в ходе выполнения дипломного проекта все поставленные задачи были выполнены. Навыки и знания полученные в процессе создания дипломного проекта будут полезны в последующих проектах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] FINS Commands: reference manual / Omron corporation – Kyoto, 2001
- [2] CJ2 CPU Unit Software: User's Manual/ Omron corporation – Kyoto, 2020
- [3] SYSMAC CS Series CS1G/H– CPU□□– EV1 CS1G/H– CPU□□H CS1D– CPU□□HA CS1D– CPUSA CS1D– CPU□□P CS1D– CPU□□H CS1D– CPU□□S SYSMAC CJ Series CJ1H– CPU□□H– R CJ1G– CPU□□ CJ1G/H– CPU□□H CJ1G– CPU□□P CJ1M– CPU□□ SYSMAC One NSJ Series Programmable Controllers: Programming Manual/ Omron corporation – Kyoto, 2018.
- [4] JSON Web Token (JWT) [Электронный ресурс] – Режим доступа: <https://www.rfc-editor.org/rfc/rfc7519.html>
- [5] C# documentation [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/en-us/dotnet/csharp/>
- [6] React [Электронный ресурс] – Режим доступа: <https://react.dev/learn>
- [7] MUI [Электронный ресурс] – Режим доступа: <https://mui.com/>
- [8] AG Grid [Электронный ресурс] – Режим доступа: <https://www.ag-grid.com/>
- [9] SQL server technical documentation [Электронный ресурс] – Режим доступа: <https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16>
- [10] Javascript [Электронный ресурс] – Режим доступа: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
- [11] Баланс электрической энергии [Электронный ресурс] – Режим доступа: https://www.belstat.gov.by/upload-belstat/upload-belstat-excel/Oficial_statistika/Godovwe/Str_TEB-20.xlsx
- [12] Use case diagram [Электронный ресурс] – Режим доступа: https://en.wikipedia.org/wiki/Use_case_diagram

Приложение А
(Обязательное)
Исходный код программного средства

Файл ConnectionController.cs

```
using System.Collections.Generic;
using backend_.Connection.ControllerConnection;
using System.Collections.Concurrent;
using backend_.DataBase.ControllerDB;
using backend_.Models.controller;
using backend_.Connection.ControllerConnection.OmronController;
using Microsoft.AspNetCore.SignalR;
using Newtonsoft.Json;
using System.Text;

namespace backend_.Connection
{
    public interface IControllerManufactory
    {
        public IControllerConnection Create(
            UserController controller);
        public IControllerConnection Create();
        public static ControllerVersion GetControllerName
            { get; }
        public List<State> states { get; }
    }

    public class OmronManufactory : IControllerManufactory
    {
        public IControllerConnection Create(UserController
            userController)
        {
            return new OmronConnectionController(
                userController.IpAddress,
                userController.IpPort);
        }

        public IControllerConnection Create()
        {
            return new OmronConnectionController();
        }

        public static ControllerVersion GetControllerName { get; }
        = new ControllerVersion()
        {
            Name = OmronConnectionController.GetName,
```

```

        version = OmronConnectionController.GetVersion
    };

    public List<State> states { get; } =
        OmronConnectionController.AllowedState;
}

public class ControllerVersion
{
    public string Name { get; set; }
    public List<string> version { get; set; }
}

public static class ControllerFactory
{
    public static Dictionary<ControllerVersion,
        IControllerManufactory> some { get; } =
        new Dictionary<ControllerVersion,
            IControllerManufactory>()
        {
            { OmronManufactory.GetControllerName,
                new OmronManufactory() },
        };

    public static IControllerConnection Create(UserController
        controller)
    {
        var controllers = some.Where(x => x.Key.Name ==
            controller.controllerName.Name).ToList();
        if(controllers[0].Key.version.Contains(
            controller.controllerName.version))
        {
            return controllers[0].Value.Create(controller);
        }
        return null;
    }
}

public class NameAndVersion
{
    public string name { get; set; }
    public List<string> version { get; set; } =
        new List<string>();
}

public class ConnectionController : BackgroundService,
    IDisposable
{

```

```

private ConcurrentDictionary<UInt32,
    IControllerConnection> _taskManager = new
    ConcurrentDictionary<UInt32,
    IControllerConnection>();
private readonly IServiceScopeFactory
    _serviceScopeFactory;
private async Task Initialize()
{
    ControllerDBContext controllerDB;
    using (var scope = _serviceScopeFactory.Cre
        ateScope())
    {
        controllerDB = scope.ServiceProvider.GetService
            <ControllerDBContext>();
        var controllers = await controllerDB
            .GetAllControllers();
        foreach (var controller in controllers)
        {
            var controllerConnection = ControllerFactory
                .Create(new UserController()
            {
                controllerName = new UserControllerName()
                { Name = controller.controllerName.name,
                    version = controller.controllerName.ver
                        sion },
                IPAddress = controller.IPAddress,
                IPPort = controller.IPPort
            });

            var res = _taskManager.TryAdd(controller.
                IPAddress, controllerConnection);
            if(res)
            {
                foreach (var command in controller.
                    outputs)
                {
                    controllerConnection.AddCommand(
                        command.id.ToString(), com
                            mand.Query!=null?com
                                mand.Query.query:null);

                    var ControllerCommand =
                        controllerConnection.GetCom
                            mand(command.id.ToString());
                    var CommandState = ControllerCommand.
                        GetAllowedState().FirstOrDe
                            fault(x=>x.description==com
                                mand.outputState.description);
                    if(CommandState != null)
                    {
                        ControllerCommand.SetState(

```

```

        CommandState);
        ControllerCommand.
            SetAnswerListener(this.Answer
                Listener);
    }
}
var states = controllerConnection.
    GetAllowedState();
var ControllerState = states.
    FirstOrDefault(x => x.description ==
        controller.ControllerState.
            Description);
if (ControllerState != null)
{
    controllerConnection.SetState(
        ControllerState);
}
controllerConnection.Start();
}
}
}

protected override async Task ExecuteAsync(
    CancellationToken stoppingToken)
{
    await Initialize();
}

public bool SetOutputState(string state, UInt32
    address, int id)
{
    _taskManager.TryGetValue(address, out var
        controller);
    if (controller == null)
        return false;
    var command = controller.GetCommand(id.ToString());
    if (command == null)
        return false;
    var states = command.GetAllowedState();
    var commandState = states.FirstOrDefault(x => x.
        description == state);
    if (commandState == null)
        return false;
    command.SetState(commandState);
    return true;
}

public List<State> GetAllowedState(string ControllerName,

```

```

        string controllerVersion)
    {
        var NamesAndVersions = ControllerFactory.some;
        var value = NamesAndVersions.FirstOrDefault(x =>
            x.Key.Name == ControllerName && x.Key.version.
            Contains(controllerVersion));
        var states = value.Value.states;
        return states;
    }
    public List<State> GetAllowedOutputState(string
        ControllerName, string controllerVersion)
    {
        var NamesAndVersions = ControllerFactory.some;
        var controller = NamesAndVersions.FirstOrDefault(x =>
            x.Key.Name == ControllerName && x.Key.version.
            Contains(controllerVersion));
        var states = controller.Value.Create().
            GetAllowedState();

        return states;
    }

    private class OutValue
    {
        public UInt32 value { get; set; }
        public UInt32 controllerAddress { get; set; }
        public int controllerOutputId { get; set; }
        public DateTime DateTime { get; set; }

        public OutValue(OutputValue value)
        {
            this.value = (UInt32)(value.value[0] +
                ((int)value.value[1] << 8) +
                ((int)value.value[2] << 16) +
                ((int)value.value[3] << 24));
            controllerAddress = value.controllerAddress;
            controllerOutputId = value.controllerOutputId;
            DateTime = value.DateTime;
        }
    }

    private async Task AnswerListener(OutputValue value)
    {
        var res = new OutValue(value);

        await hub.Clients.Group(new backend_.Controllers.
            ValueControllers.ValueControll1.OutputId() {
                ip = value.controllerAddress, outputId =
                value.controllerOutputId }.ToString())
            .SendAsync("Receive",

```

```

        JsonConvert.SerializeObject(res));
using (var scope =
    _serviceScopeFactory.CreateScope())
{
    var dbContext = scope.ServiceProvider.Get
        Service<ControllerDbContext>();
    dbContext.AddOutputValue(value);
}
}

private readonly IHubContext<backend_.Controllers.Value.
    Controllers.ValueControll> hub;

public ConnectionController(IServiceScopeFactory
    serviceScopeFactory, IHubContext<backend_.Control
    lers.ValueControllers.ValueControll> hubContext)
{
    _serviceScopeFactory = serviceScopeFactory;
    hub = hubContext;
}

public void Start(UInt32 id)
{
    _taskManager.TryGetValue(id, out var controller);
    if (controller == null)
        return;
    controller.Start();
}

public void Stop(UInt32 id)
{
    _taskManager.TryGetValue(id, out var controller);
    if (controller == null)
        return;
    controller.Stop();
}

public bool AddController(IControllerConnection
    controller)
{
    _taskManager.TryGetValue(controller.id, out var
        OldController);
    if (OldController != null)
        return false;
    _taskManager.TryAdd(controller.id, controller);
    return true;
}

public bool RemoveController(UInt32 id)
{
    _taskManager.TryGetValue(id, out var Oldcontroller);

```



```

        if (Oldcontroller == null)
            return true;
        _taskManager.TryRemove(id, out Oldcontroller);
        if (Oldcontroller == null)
            return false;
        return true;
    }

    public List<string> GetAllCommands(UInt32 address)
    {
        var controller = _taskManager.Where(x=>x.Value.id==
            address).FirstOrDefault();
        var comand = controller.Value.AllowedCommand;
        return comand;
    }

    public Dictionary<string,List<string>>
        GetAllowedController()
    {
        var version = ControllerFactory.some.ToList();
        var Dict = new Dictionary<string, List<string>>();
        foreach( var item in version)
        {
            Dict.Add(item.Key.Name, item.Key.version);
        }
        return Dict;
    }

    public bool RemoveCommand(UInt32 address,int id)
    {
        _taskManager.TryGetValue(address, out var
            Controller);
        if (Controller == null)
            return false;
        var res = Controller.DeleteComand(id.ToString());
        return res;
    }

    public bool SetCommand(UInt32 address,int id,string
        command)
    {
        _taskManager.TryGetValue(address, out var
            Controller);
        if (Controller == null)
            return false;
        var res = Controller.AddCommand(id.ToString(),
            command);
        if(res == false)
        {
            _taskManager.TryRemove(address, out var
                Oldcontroller);
        }
    }

```

```

        }
        return res;
    }
    public void Dispose()
    {

    }

}
}

```

Файл FinsCommand.cs

```

using backend_.Connection.ControllerConnection.OmronController.
    TransportLayer;
using Newtonsoft.Json;
using System.Text;
using backend_.Models.controller;

namespace backend_.Connection.ControllerConnection.
    OmronController.FinsCmd
{
    [Serializable]
    public class FinsRequest
    {
        public string code { get; set; }

        public string memoryArea { get; set; }
        public string descriptions { get; set; }

        public int startAddress { get; set; }

        public byte bitshift { get; set; }

        public int length { get; set; }
    }
    public class FinsComand : IControllerCommandImplementation
    {
        public int id { get; set; }
        public int Address { get; set; }

        public void SetState(State state)
        {
            lock (this.IsRun)
            {
                this.IsRun.IsRun = state.IsRun;
                this.IsRun.description = state.description;
            }
        }
        public static List<State> AllowedState { get; } =
            new List<State>()

```

```

    {
        new State() { IsRun = true, description = "RUN" },
        new State(){ IsRun = false, description = "STOP" }
    };

    public List<State> GetAllowedState()
    {
        return AllowedState;
    }

    public State IsRun { get; set; }
    #region
    private event CommandListener Answer;
    private event CommandListener Errore;
    public void SetAnswerListener(CommandListener Delegate)
    {
        Answer += Delegate;
    }
    public void DeleteAnswerListener(CommandListener
        Delegate)
    {
        Answer -= Delegate;
    }

    public FinsComand(byte[] comand)
    {
        IsRun = new State();
    }
    public FinsComand(UInt32 address,int id)
    {
        IsRun = new State();
        this.Address = Address;
        this.id = id;
    }

    public void SetCommand(string comand)
    {
        var command = JsonConvert.DeserializeObject
            <FinsRequest>(comand);
        var code = int.Parse(command.code);
        var memoryArea = (MemoryArea)byte.Parse
            (command.memoryArea);
        if (code == (ushort)FinsComandCode.MemoryAreaRead)
        {
            MemoryAreaRead((MemoryArea)byte.Parse(command.
                memoryArea), (UInt16)command.startAddress,
                (UInt16)command.length,
                command.bitshift,null);
        }
    }

```

[illegible]

```

        0x00,
        0x00,
    };

    byte[] FinsTCPHeadeer = new byte[]
    {
        0x46, 0x49, 0x4E, 0x53,          // 'F' 'I' 'N' 'S'
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x02,
        0x00, 0x00, 0x00, 0x00
    };

    UInt16 finsCommandLen = 0;

    const int F_PARAM = 12;

    byte[] ResponseHeader = new byte[]
    {
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00,
        0x00, 0x00, 0x00, 0x00
    };

    Byte[] respFins = new Byte[2048];

    Byte[] respFinsData = new Byte[2048];

    UInt16 finsResponseLen = 0;

    public TCPClient Client
    {
        get { return Client; }
        set { Client = value; }
    }

    public byte[] ResponseData
    {
        get { return respFinsData; }
    }

    public byte[] ResponceAddress
    {
        get { return new byte[10]; }
    }

    public byte ICF
    {

```

```

        get { return cmdFins[0]; }
        set { cmdFins[0] = value; }
    }
    public byte RSV
    {
        get { return this.cmdFins[1]; }
        set { this.cmdFins[1] = value; }
    }
    public byte GCT
    {
        get { return this.cmdFins[2]; }
        set { this.cmdFins[2] = value; }
    }

    public byte DNA
    {
        get { return this.cmdFins[3]; }
        set { this.cmdFins[3] = value; }
    }


    public byte MC
    {
        get { return this.cmdFins[10]; }
        set { this.cmdFins[10] = value; }
    }

    public byte SC
    {
        get { return this.cmdFins[11]; }
        set { this.cmdFins[11] = value; }
    }

    public UInt16 Comand
    {
        get { return (UInt16)(cmdFins[10] >> 8 +
            cmdFins[11]); }
        set { cmdFins[10] = (byte)(value >> 8 & 0xff);
            cmdFins[11] = (byte)(value & 0xff); }
    }

    public byte SA1
    {
        get { return cmdFins[7]; }
        set { cmdFins[7] = value; }
    }
    public byte DA1
    {
        get { return cmdFins[4]; }
        set { cmdFins[4] = value; }
    }

```

```

    }

    public int FinsHeaderErrore
    {
        get { return ResponseHeader[8] << 24 +
            ResponseHeader[9] << 16 + ResponseHeader[10]
            << 8 + ResponseHeader[11]; }
    }

    public byte FinsResponceMainErroreCode
    {
        get { return respFins[12]; }
    }

    public byte FinsResponceSubErroreCode
    {
        get { return respFins[13]; }
    }

    public UInt16 FinsHeaderLenght
    {
        get { return (UInt16)(FinsTCPHeadeer[6] >> 8 & 0xff +
            FinsTCPHeadeer[7] & 0xff); }
        set
        {
            this.FinsTCPHeadeer[6] = (byte)((value >> 8) &
                0xFF);
            this.FinsTCPHeadeer[7] = (byte)(value & 0xFF);
        }
    }

    public UInt16 FinsResponceLenght
    {
        get
        {
            return (UInt16)(ResponseHeader[6] << 8 +
                ResponseHeader[7] & 0xff);
        }
    }
#endregion

#region FinsComand

public void MemoryAreaRead(MemoryArea memoryArea, UInt16
    StartAddress, UInt16 Count, byte? StartBitPosition,
    byte[]? data)
{
    this.Comand = (UInt16)FinsComandCode.
        MemoryAreaRead;
}

```

```

        this.cmdFins[F_PARAM] = (byte) (memoryArea);

        //memory address
        this.cmdFins[F_PARAM+1] = (byte) (StartAddress >> 8 &
            0xff);
        this.cmdFins[F_PARAM+2] = (byte) (StartAddress &
            0xff);

        //adress of start bit

        this.cmdFins[F_PARAM + 3] = StartBitPosition!=null
            ? (byte)StartBitPosition:(byte)0;

        //set count of recive param
        this.cmdFins[F_PARAM +4] = (byte) (Count >> 8 & 0xff);
        this.cmdFins[F_PARAM +5] = (byte) (Count & 0xff);

        this.finsCommandLen = F_PARAM + 6;
    }

    public void Close()
    {
        Transport.Disconnect();
    }

#endregion

#region DataSend

    public void ConnectToPLC(FinsComand fins)
    {

        this.SA1 = fins.SA1;
        this.DA1 = fins.DA1;

    }

    private bool ItWasConnect = false;

    public async Task<bool> ConnectToPLC()
    {
        byte[] cmdNADS = new byte[]
        {
            0x46, 0x49, 0x4E, 0x53, // 'F' 'I' 'N' 'S'
            0x00, 0x00, 0x00, 0x0C, // 12 Bytes expected
            0x00, 0x00, 0x00, 0x00,
            0x00, 0x00, 0x00, 0x00,
            0x00, 0x00, 0x00, 0x00
        };

```



```

        Await Transport.WriteData(cmdNADS,
                                cmdNADS.Length);

        var resposNADS = new byte[24];

        resposNADS = await Transport.ReadData(24);

        if (resposNADS[15] != 0)
        {
            if (!FinsErrorCodes.ErrorCodes.TryGet
                Value(resposNADS[15], out string errorDe
                    scription))
                errorDescription = "Unknown error";
            this._lastError = "NADS command error: " +
                resposNADS[15] + "(" + errorDescription +
                    ")";

            ItWasConnect = false;
            return false;
        }
        if (resposNADS[8] != 0 || resposNADS[9] != 0 ||
            resposNADS[10] != 0 || resposNADS[11] != 1)
        {
            this._lastError = "Error sending NADS command.
                "+ resposNADS[8].ToString() + " " + re-
                    sposNADS[9].ToString() + " " + respos-
                        NADS[10].ToString() + " " + respos-
                            NADS[11].ToString();
            ItWasConnect = false;
            return false;
        }

        this.SA1 = resposNADS[19];
        this.DA1 = resposNADS[23];

        ItWasConnect = true;
        return true;
    }

    public async Task<bool> ExecuteCommand()
    {
        var run = false;
        lock(this.IsRun)
        {
            if(this.IsRun.IsRun)
            {
                run = true;
            }
        }
    }

```

```

        if (run)
        {
            if (this.ItWasConnect)
                return await SendFrames(null);
            else
            {
                await this.ConnectToPLC();
            }
        }
        return false;
    }

    protected async Task<bool> SendFrames(byte[]? data)
    {
        Array.Fill(ResponseHeader, (byte)0, 0,
            ResponseHeader.Length);

        var FinsLength = this.finsCommandLen + 8;

        if(data!=null)
        {
            FinsLength+= data.Length;
        }

        this.FinsHeaderLenght=(UInt16)FinsLength;

        await Transport.WriteData(FinsTCPHeadeer,
            FinsTCPHeadeer.Length);

        await Transport.WriteData(cmdFins,
            finsCommandLen);

        if (data != null)
            await Transport.WriteData(data, data.Length);

        this.ResponseHeader = await Transport.ReadData
            (ResponseHeader.Length);

        if(this.FinsHeaderErrore!=2)
        {
            _lastError = "FRAME SEND error: " +
                FinsHeaderErrore;
            return false;
        }

        if (this.ResponseHeader[15] != 0)
        {

```

```

        this._lastError = "Error receving FS command:
        " + this.ResponseHeader[15];
        return false;
    }

    this.finsResponseLen = this.FinsResponseLenght;
    this.finsResponseLen - = 8;

    this.respFins = await Transport.ReadData(this.
        finsResponseLen);

    if(finsResponseLen > 14)
    {
        this.respFinsData = await Transport.
            ReadData(finsResponseLen - 14);
        var controllerData = new OutputValue() {
            controllerAddress = (UInt32)this.
                Address, controllerOutputId = this.id,
            value = this.respFins, DateTime =
                DateTime.Now };
        await Answer.Invoke(controllerData);
    }

    if (this.FinsResponseMainErrorcode != 0 || this.
        FinsResponseSubErrorcode != 0)
    {
        this._lastError += string.Format("Response
            Code error: (Code: {0} Subcode: {1})",
                this.FinsResponseMainErrorcode, this.
                    FinsResponseSubErrorcode);
        return false;
    }

    return true;
}

#endregion
}
}

```

Файл OmronConnectionController.cs

```

using backend_.Connection.ControllerConnection.OmronController.
    FinsCmd;
using backend_.Connection.ControllerConnection.OmronController.
    TransportLayer;
using System.Collections.Generic;
using System.Collections.Concurrent;

```

```

using backend_.Connection.ControllerConnection;
using System.Net;
using System.Threading;

namespace backend_.Connection.ControllerConnection.
    OmronController
{
    public class OmronConnectionController :
        IControllerConnection
    {
        public UInt32 id { get; }

        public static List<State> AllowedState { get; } =
            new List<State>()
            {
                new State(){IsRun = true,description = "RUN"},
                new State(){IsRun = false, description = "STOP"}
            };

        public List<State> GetAllowedState()
        {
            return AllowedState;
        }
        public static List<string> GetVersion { get; } =
            new List<string>()
            {
                "CJ2M- CPU33",
                "CJ2M- CPU32",
            };
        public static string GetName { get; } = "OMRON";

        private IControllerConnect connect;

        private ConcurrentDictionary<string,
            IControllerCommandImplementation> controllerCommand =
            new ConcurrentDictionary<string,
            IControllerCommandImplementation>();

        public List<string> AllowedCommand { get; private set; }
            = new List<string>();
        public State IsRun { get; set; }

        public OmronConnectionController()
        {
        }

        public OmronConnectionController(models.controller.
            Controller controller)
        {

```

```

        id = controller.IpAddress;
        this.IsRun = new State();
    }

    public OmronConnectionController(UInt32 address,int port)
    {
        this.id = address;
        this.IsRun = new State();
        AllowedCommand = FinsCommand.allowedCommand;

        connect = new TCPClient();
        connect.SetIpAddress(address, port);
    }

    public void SetState(State state)
    {
        lock(this.IsRun)
        {
            this.IsRun.description = state.description;
            this.IsRun.IsRun = state.IsRun;
            Monitor.PulseAll(this.IsRun);
        }
    }

    public IControllerCommand? GetCommand(string OutputId)
    {
        controllerCommand.TryGetValue(OutputId, out var cmd);
        return cmd;
    }

    public async void Stop()
    {
        lock(this.IsRun)
        {
            this.IsRun.IsRun = false;
            Monitor.PulseAll(this.IsRun);
        }
    }

    public async Task Start()
    {
        while (true)
        {
            lock (this.IsRun)
            {
                while (!this.IsRun.IsRun)
                    Monitor.Wait(this.IsRun);
            }
        }
    }

```

```

    }

    foreach (var cmd in controllerCommand)
    {
        var res = await cmd.Value.ExecuteCommand();
    }
    Thread.Sleep(1000);
}

}

#region FinsComand
public bool AddCommand(string OutputId, string? command)
{
    var FinsCommand = new FinsCommand
        (this.id, int.Parse(OutputId));
    if (command != null)
        FinsCommand.SetCommand(command);
    FinsCommand.SetTransportLayer(this.connect);
    controllerCommand.TryAdd(OutputId, FinsCommand);
    return true;
}

public bool DeleteComand(string OutputId)
{
    controllerCommand.TryRemove(OutputId, out var cmd);
    return true;
}

#endregion

}

}

```

Файл IFinscommand.cs

```

namespace backend_.Connection.ControllerConnection.OmronController.FinsCmd
{
    public enum MemoryArea : byte
    {
        CIO_Bit = 0x30,
        WR_Bit = 0x31,
        HR_Bit = 0x32,
        AR_Bit = 0x33,
        CIO_Bit_FS = 0x70,
        WR_Bit_FS = 0x71,
    }
}

```

```

    HR_Bit_FS = 0x72,
    CIO = 0xB0,
    WR = 0xB1,
    HR = 0xB2,
    AR = 0xB3,
    CIO_FS = 0xF0,
    WR_FS = 0xF1,
    HR_FS = 0xF2,
    TIM = 0x09,
    CNT = 0x09,
    TIM_FS = 0x49,
    CNT_FS = 0x49,
    TIM_PV = 0x89,
    CNT_PV = 0x89,
    DM_Bit = 0x02,
    DM = 0x82,
    TK_Bit = 0x06,
    TK = 0x46
};

public enum FinsComandCode: UInt16
{
    MemoryAreaRead = 0x0101,
    MemoryAreaWrite = 0x0102,
}

class FinsErrorCodes
{
    public static IDictionary<byte, string> ErrorCodes
    {
        get; } = new Dictionary<byte, string> {
        { 0x0, "No Error" },
        { 0x1, "Invalid Memory Address Parameter" },
        { 0x2, "Invalid or Illegal Command Param" },
        { 0x3, "Response SID did not match" },
        { 0x4, "NSB did not respond to send req" },
        { 0x5, "Timed Out - No Response" },
        { 0x6, "Timed Out Waiting For Response" },
        { 0x7, "Bad Received CRC" },
        { 0x8, "Unmatched Message IDs" },
        { 0x9, "Unmatched Command/Response" },
        { 0xa, "Unknown Error" },
        { 0xb, "Network Address Out Of Range" },
        { 0xc, "Node Address Out Of Range" },
        { 0xd, "Unit Address Out Of Range" },
        { 0xe, "Invalid Address Parameter" },
        { 0xf, "Timed Out waiting for echo" },
        { 0x10, "Bad Received FCS" },
        { 0x11, "Response from different Host Link Unit" },
        { 0x12, "No Valid Response Code" },
        { 0x13, "No FINS Response Packet" },
        { 0x14, "Unknown Error Code" },
    }
}

```

```

{ 0x15, "Local Node not part of Network" },
{ 0x16, "Token Timeout, Node # too High" },
{ 0x17, "Number of Transmit Retries Exceeded" },
{ 0x18, "Max # of Frames Exceeded" },
{ 0x19, "Node # Setting Error" },
{ 0x1a, "Node # Duplication Error" },
{ 0x1b, "Dest Node not part of Network" },
{ 0x1c, "No Node with Node # Specified" },
{ 0x1d, "Third Node not part of Network" },
{ 0x1e, "Busy Error, Dest Node Busy" },
{ 0x1f, "Response Timeout, Noise or Watchdog" },
{ 0x20, "Error in Comm Controller" },
{ 0x21, "PLC Error in Dest Node" },
{ 0x23, "Undefined Command Used" },
{ 0x24, "Cannot Process Command" },
{ 0x25, "Routing Error" },
{ 0x26, "Command is too Long" },
{ 0x27, "Command is too Short" },
{ 0x28, "Specified Data Items != Actual" },
{ 0x29, "Incorrect Command Format" },
{ 0x2a, "Incorrect Header" },
{ 0x2b, "Memory Area Code Error" },
{ 0x2c, "Access Size Specified is Wrong" },
{ 0x2d, "First Address is Inaccessible" },
{ 0x2e, "Address Range Exceeded" },
{ 0x2f, "Unknown Error Code" },
{ 0x30, "Non-existent Program # Specified" },
{ 0x31, "Unknown Error Code" },
{ 0x32, "Unknown Error Code" },
{ 0x33, "Data Size in Command is Wrong" },
{ 0x34, "Unknown Error Code" },
{ 0x35, "Response Block too Long" },
{ 0x36, "Incorrect Parameter Code" },
{ 0x37, "Program Area Protected" },
{ 0x38, "Registered Table Error" },
{ 0x39, "Area Read- Only or Write Protected" },
{ 0x3b, "Mode is Wrong" },
{ 0x3c, "Mode is Wrong (Running)" },
{ 0x3d, "PLC is in Program Mode" },
{ 0x3e, "PLC is in Debug Mode" },
{ 0x3f, "PLC is in Monitor Mode" },
{ 0x40, "PLC is in Run Mode" },
{ 0x41, "Specified Node is not Control Node" },
{ 0x42, "Specified Memory does not Exist" },
{ 0x43, "No Clock Exists" },
{ 0x44, "Data Link Table Error" },
{ 0x45, "Unit Error" },
{ 0x46, "Command Error" },
{ 0x47, "Destination Address Setting Error" },
{ 0x48, "No Routing Tables" },
{ 0x49, "Routing Table Error" },

```



```

    { 0x4a, "Too Many Relays" },
    { 0x4b, "The Header is not FINS" },
    { 0x4c, "The Data Length is too Long" },
    { 0x4d, "The Command is not Supported" },
    { 0x50, "Timed Out waiting for Port Semaphore" },
    { 0x5e, "All Connections are in Use" },
    { 0x5f, "The Specified Node is Already Connected" },
    { 0x60, "Attempt to Access Protected Node from
        Unspecified IP" },
    { 0x61, "The Client FINS Node Address is OOR" },
    { 0x62, "Same FINS Node Address is being used
        by Client and Server" },
    { 0x63, "No Node Addresses are Available to
        Allocate" },
};
}

}

```