Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

*К защите допустить*:

Заведующая кафедрой ПОИТ

                          Н. В. Лапицкая

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему

**Программное средство сбора и визуализации технологических параметров химического цеха на платформе .NET**

БГУИР ДП 1-40 01 01 125 ПЗ

|  |  |
| --- | --- |
| Студент | Г.Б. Насанович |
|  |  |
| Руководитель | В.Л. Богомаз |
|  |  |
| Консультанты: |  |
| *от кафедры ПОИТ* | В.А. Леванцевич |
| *по экономической части* | А.А. Горюшкин |
|  |  |
| Нормоконтролёр | П.Н. Красковский |
|  |  |
| Рецензент |  |

Минск 2023

Содержание

[Введение 5](#_Toc135661731)

[1 Аналитический обзор программных продуктов, методов и подходов по теме дипломного проекта 7](#_Toc135661732)

[1.1 Анализ существующих решений по теме дипломного проекта 7](#_Toc135661733)

[1.2 Постановка задач дипломного проектирования 13](#_Toc135661734)

[2 Моделирование предметной области, Разработка функциональных требований и составление их спецификации 15](#_Toc135661735)

[2.1 Общие сведения и требования к работе программного средства 15](#_Toc135661736)

[2.2 Описание функциональности программного средства 15](#_Toc135661737)

[2.3 Разработка протоколов связи 22](#_Toc135661738)

[2.4 Разработка информационной модели 23](#_Toc135661739)

[2.5 Разработка спецификации функциональных требований 24](#_Toc135661740)

[3 Проектирование программного средства 26](#_Toc135661741)

[3.1 Разработка программной архитектуры 26](#_Toc135661742)

[3.2 Проектирование архитектуры программного средства 27](#_Toc135661743)

[3.3 Проектирование базы данных 29](#_Toc135661744)

[3.4 Проектирование алгоритма соединения пользователя и контроллера 30](#_Toc135661745)

[3.5 Проектирование алгоритма добавления новой команды для контроллера 32](#_Toc135661746)

[4 Разработка программного обеспечения 34](#_Toc135661747)

[4.1 Выбор и обоснование языка и среды разработки программного средства 34](#_Toc135661748)

[4.2 Разработка программной архитектуры 35](#_Toc135661749)

[5 Тестирование и проверка работоспособности программного средства 36](#_Toc135661750)

[6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 38](#_Toc135661751)

[7 Технико-экономическое обоснование разработки программного средства сбора и визуализации технологических параметров химического цеха на платформе .NET 39](#_Toc135661752)

[7.1 Описание функций, назначения и потенциальных пользователей ПО 39](#_Toc135661753)

[7.2 Расчет затрат на разработку программного продукта 39](#_Toc135661754)

[7.3 Оценка результата (эффекта) от использования ПО 41](#_Toc135661755)

[Заключение 43](#_Toc135661756)

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 относиться накипь и ускоренная коррозия.

Накипь – это твердые отложения солей, образующиеся в процессе выкипания 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 коррозированною 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 использование человеком.

# Аналитический обзор программных продуктов, методов и подходов по теме дипломного проекта

## 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.

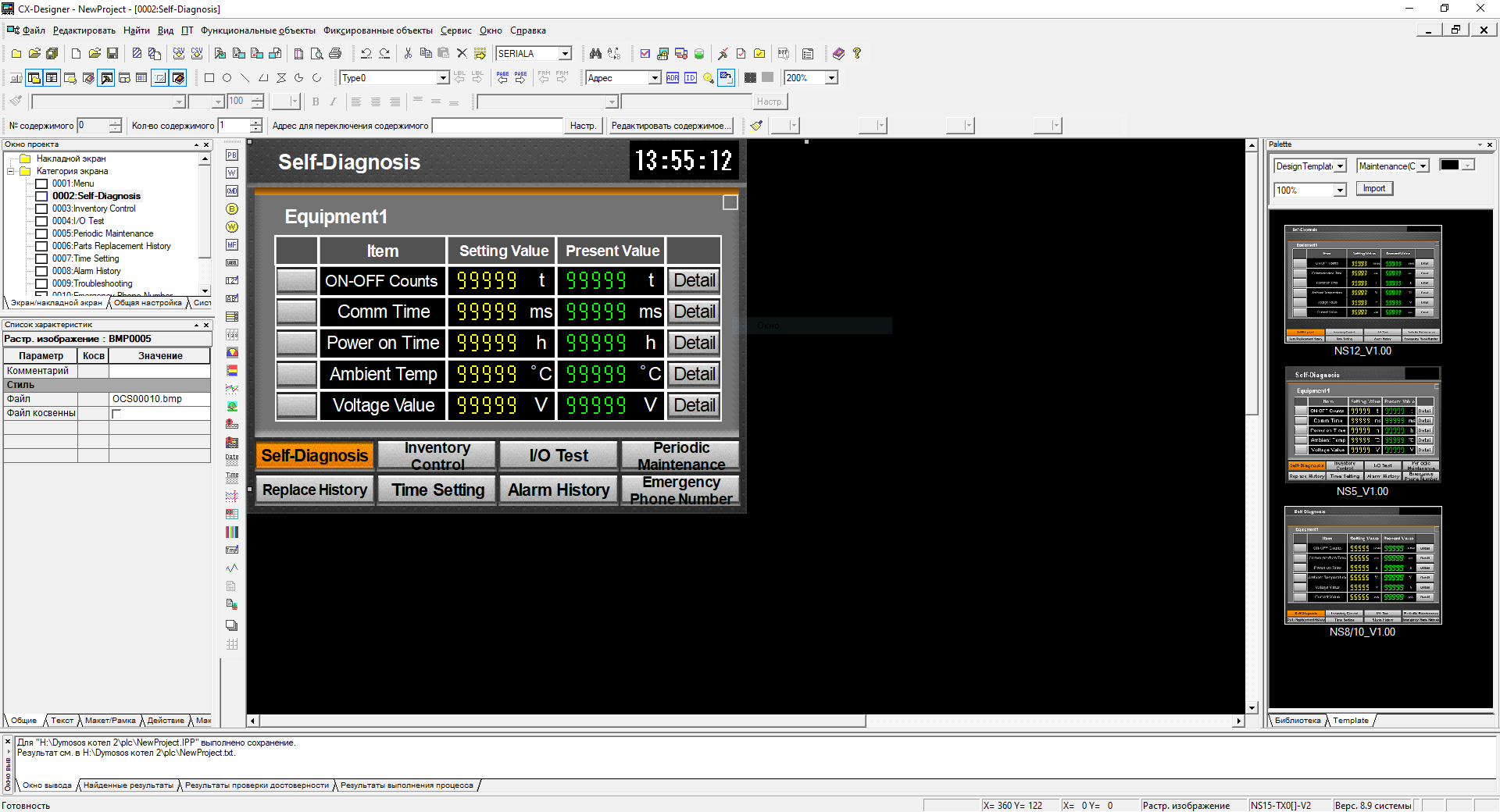


Рисунок 1.1 – Пользовательский интерфейс CX-Designer

Компоненты находиться на панели инструментов, что обеспечивает быстрый доступ к ним. Для размещения компонента необходимо просто перетащить 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.

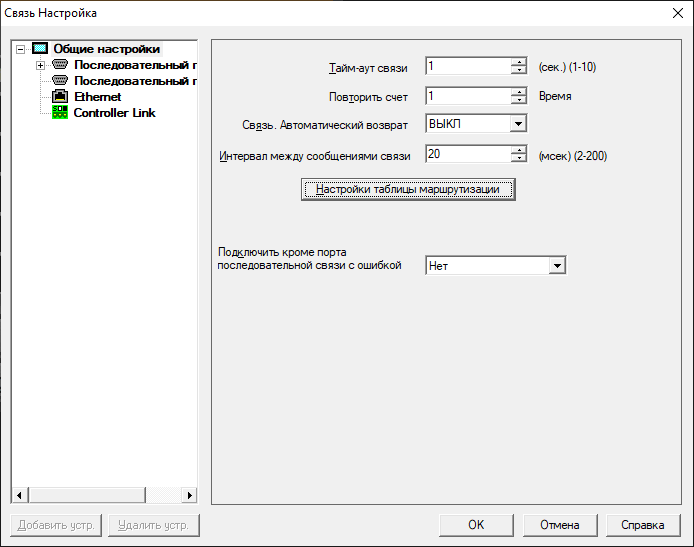


Рисунок 1.3 – Общие настройки связи

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

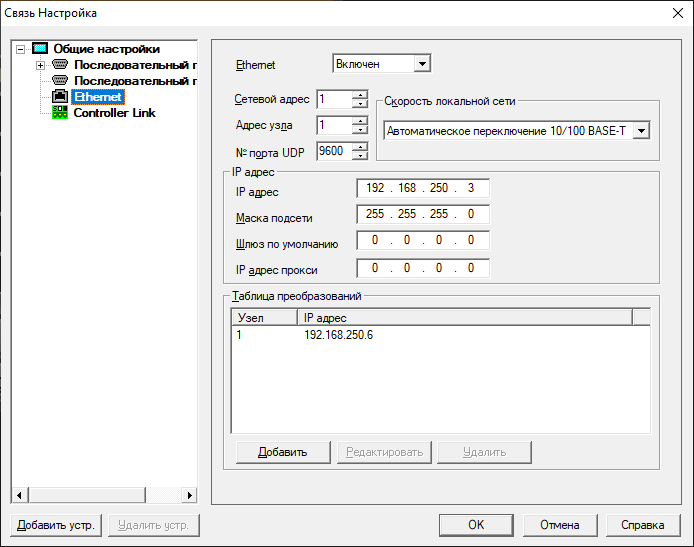


Рисунок 1.4 – Настройки подключения по сети Ethernet

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 так же обладает рядом недостатков:

* отсутствие возможности написание интерфейсов для контролирующих 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.

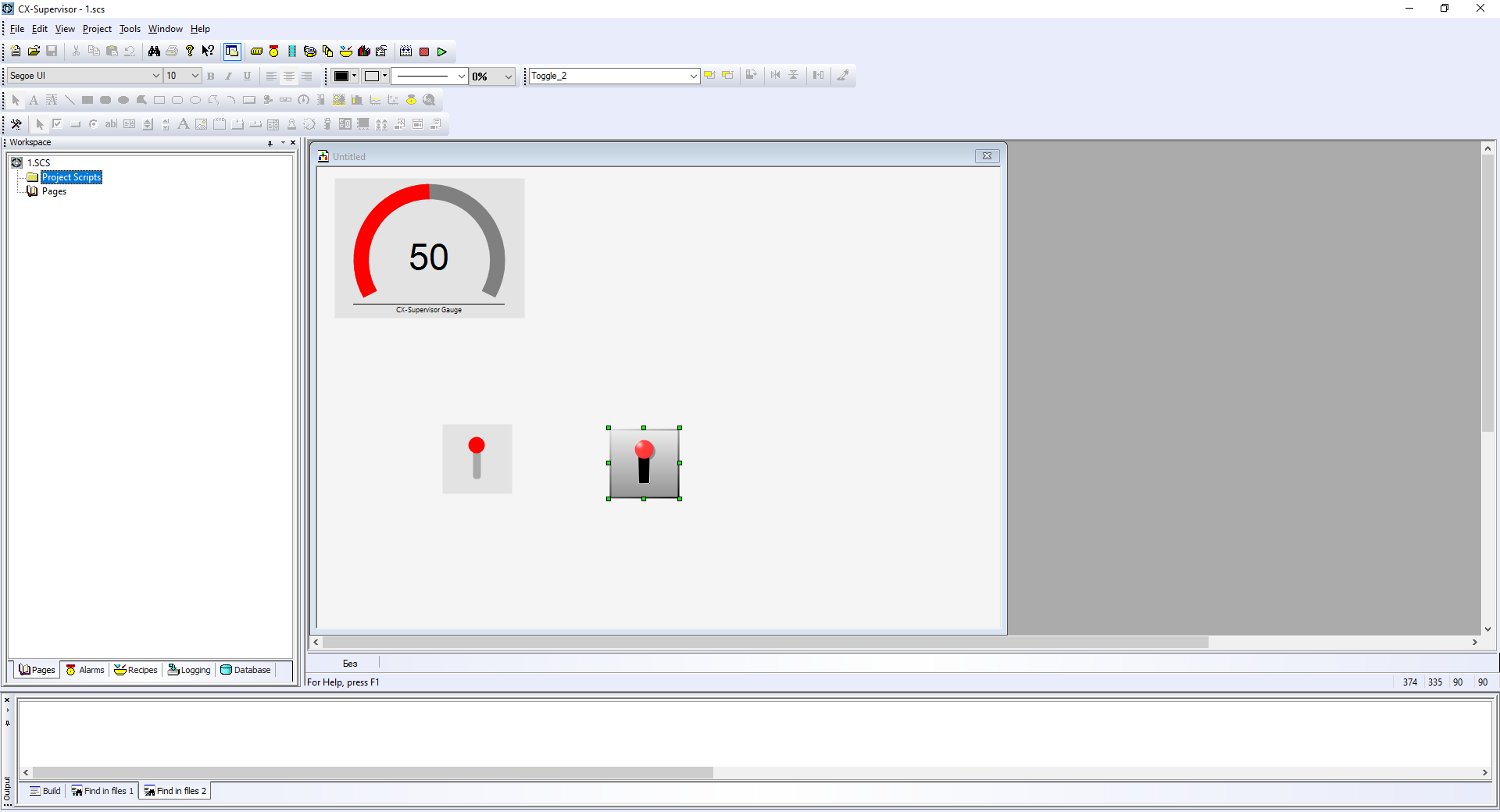


Рисунок 1.5 – Пользовательский интерфейс CX–Supervisor

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 соединятся 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

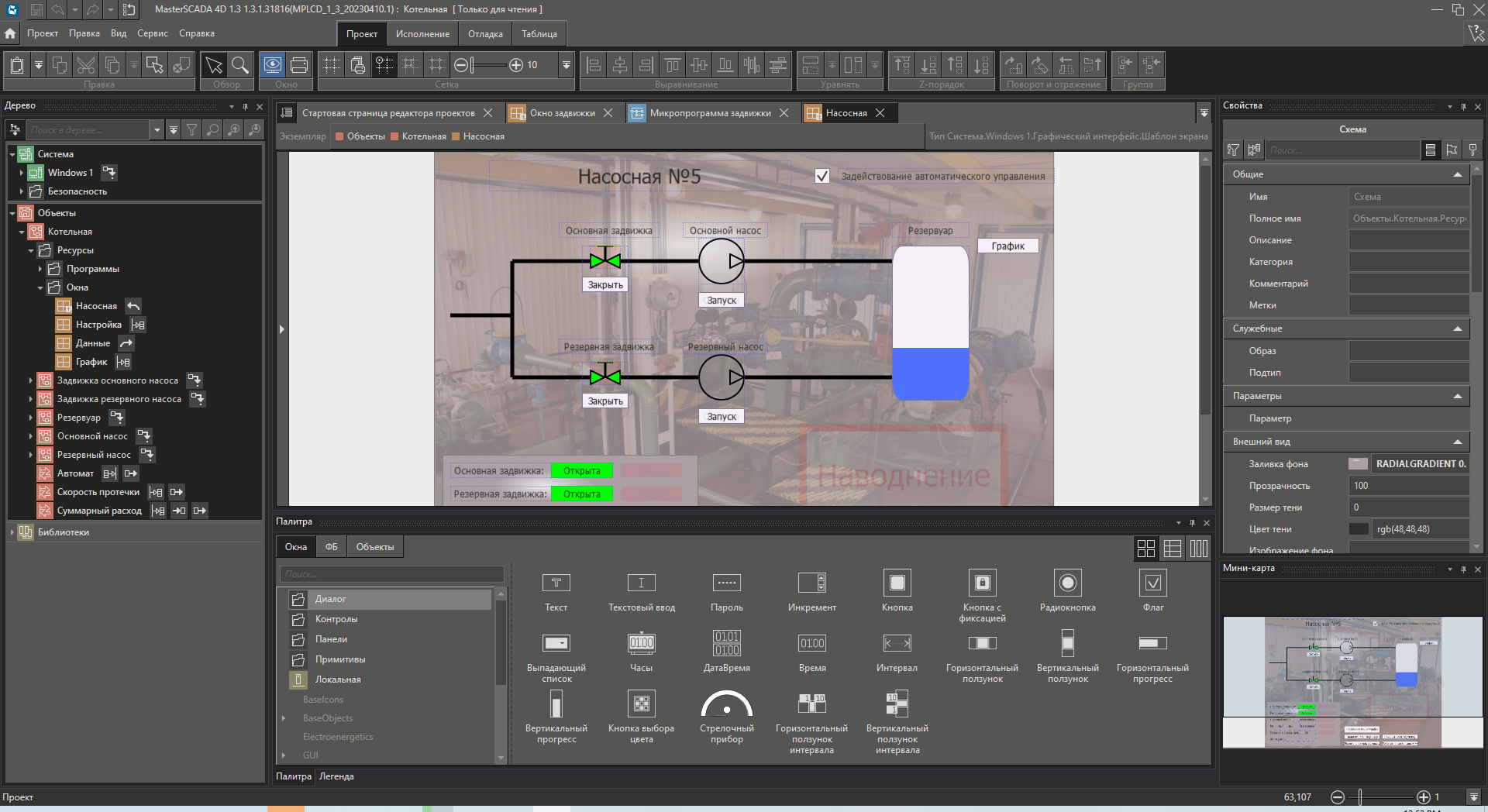


Рисунок 1.6 – Пользовательский интерфейс Master SCADA 4d

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 OC 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 изменения 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.

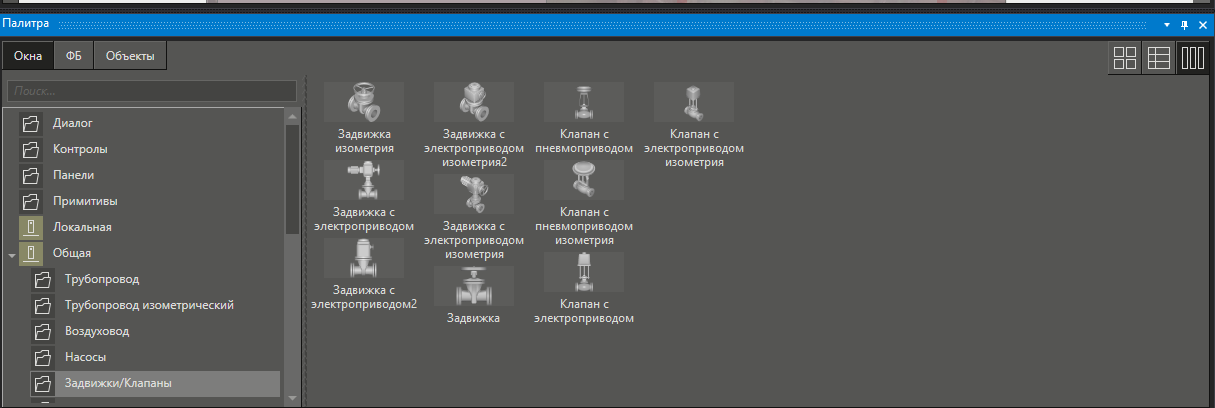


Рисунок 1.7 – Пример пользовательских компонентов

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.

## 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.

# Моделирование предметной области, Разработка функциональных требований и составление их спецификации

## 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.

## 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 авторизированного пользователя.

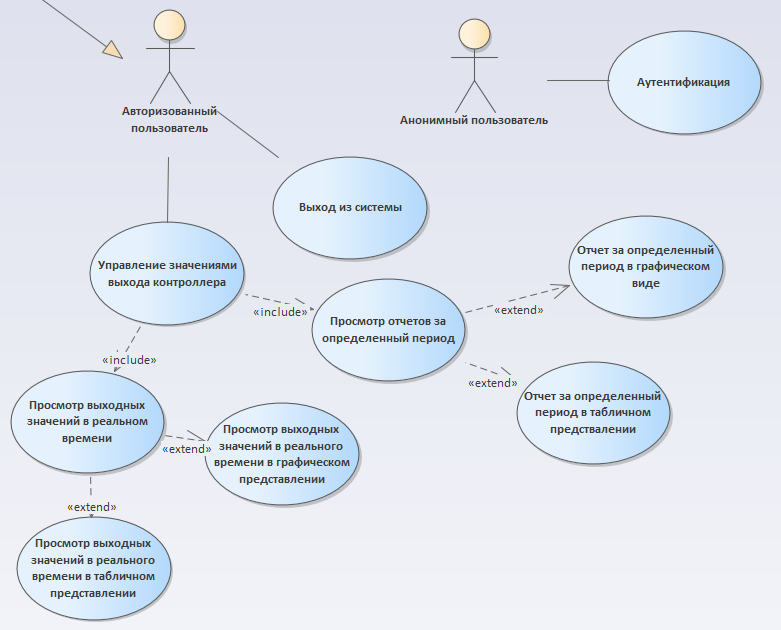


Рисунок 2.1 – Диаграмма использования анонимного и авторизованного пользователя

В зависимости от пожелания пользователя отчеты за определенный период времени могут быть представлены как в графическом виде, так и в табличном виде. Отслеживание за изменениями в реальном времени тоже 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.

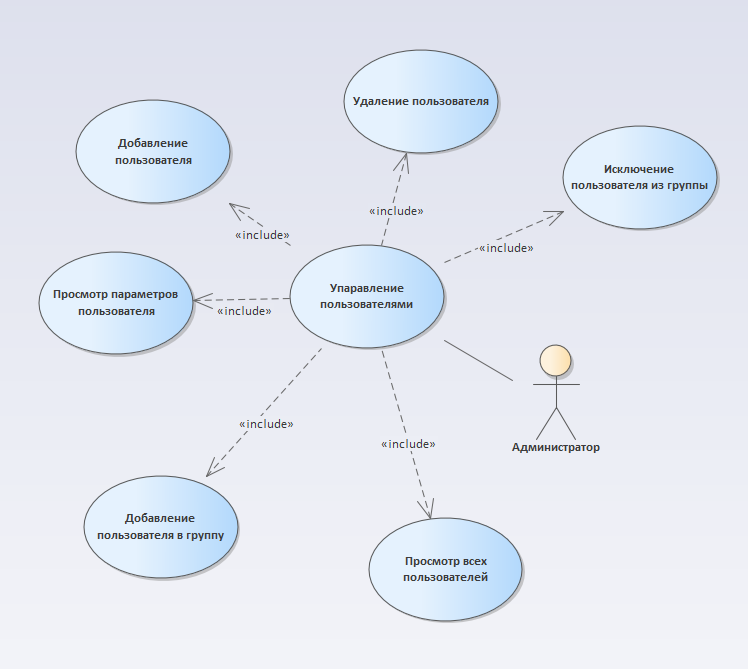


Рисунок 2.2 – Диаграмма вариантов использования для работы с учетными записями пользователей

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.

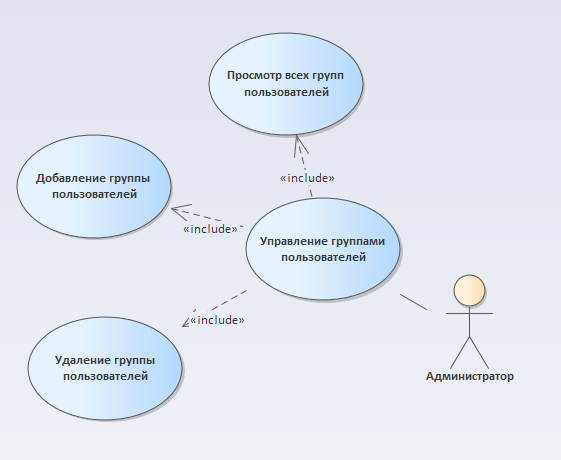


Рисунок 2.3 – Диаграмма вариантов использования для работы с пользовательскими группами

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

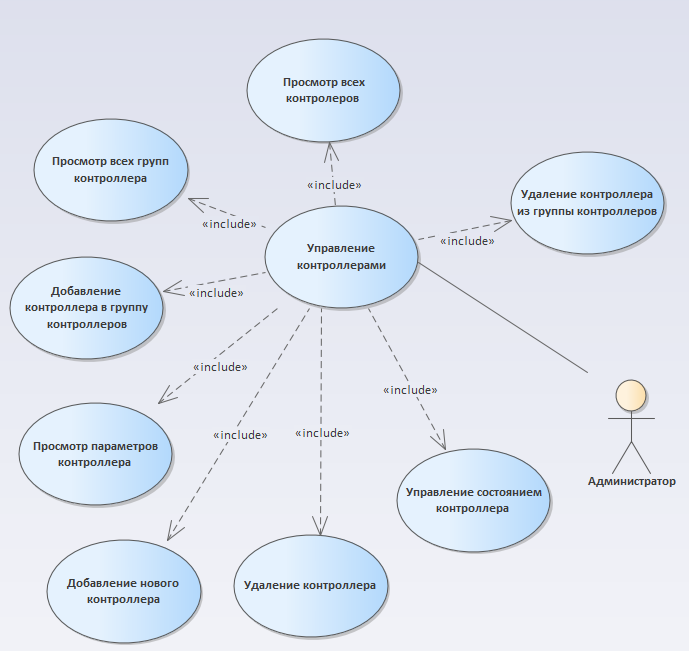


Рисунок 2.4 – Диаграмма вариантов использования для работы с контроллерами

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.

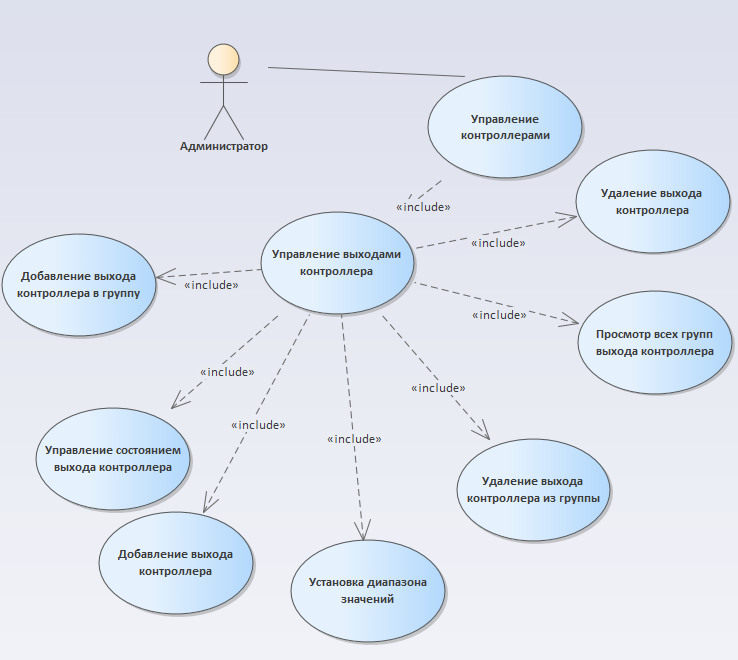


Рисунок 2.5 – Диаграмма вариантов использования для работы с выходами контроллера

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.

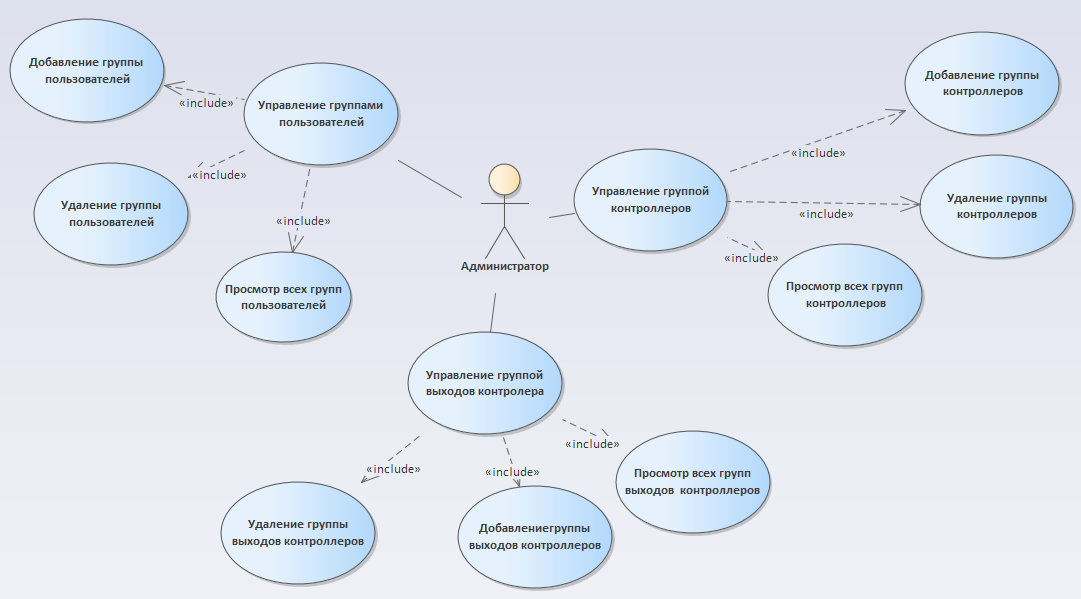


Рисунок 2.6 – Диаграмма вариантов использования для работы с группировкой

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

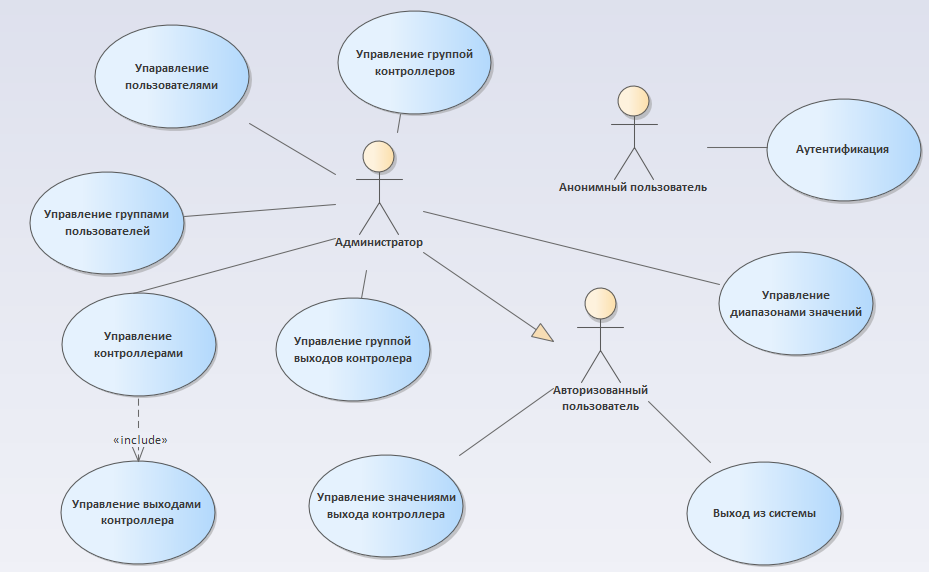


Рисунок 2.7 –Диаграмма вариатов использования системой

## 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 совой 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.

## Разработка информационной модели

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

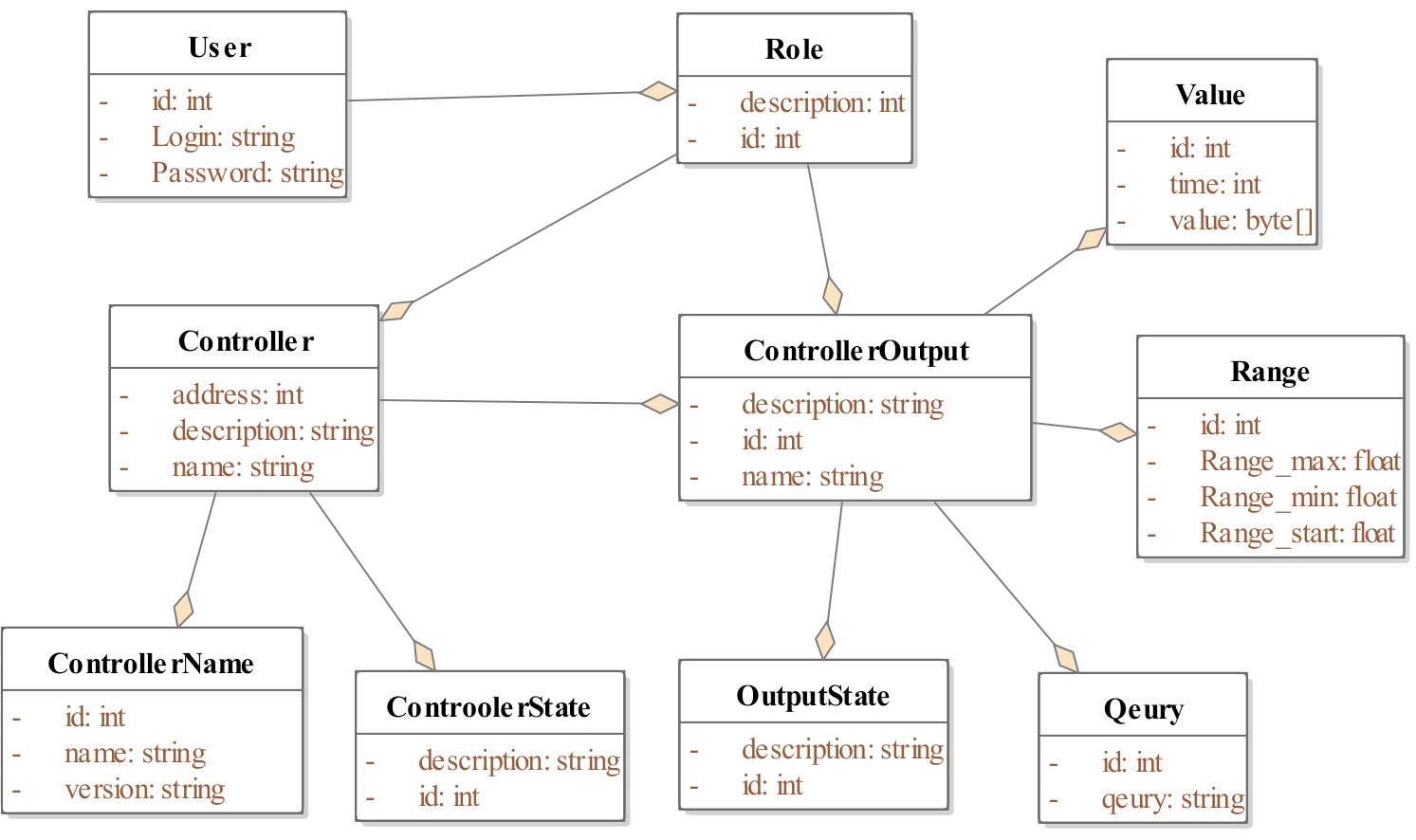


Рисунок 2.2 – Информационная модель программного средства

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

* User – сущность представляющая собой все типы пользователей системы. Атрибут id – идентификатор пользователя, Login – идентифицирующее имя пользователя, Password – пароль пользователя;
* 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;
* Controller – сущность представляющая контроллер. Атрибут address – идентификатор контроллера, хранящий адрес контроллера, description – описание контроллера, name – имя контроллера;
* ControllerName – сущность, которая хранит информацию о производители контроллеров и серии контроллера. Атрибут id – содержит идентификатор сущности, name- содержит информацию о производителе контроллеров, version – содержит серию и версию контроллера.
* ControllerState – сущность, хранящая информацию о состоянии контроллера. Атрибут id – идентификатор состояния контроллера, description – строковое значение состояния.
* ControllerOutput – сущность, предоставляющая информацию выходных значениях контроллера. Атрибут id – идентификатор выхода, description – описание выхода, name- название выхода;
* OutputState – сущность, хранящая информацию о состоянии выхода контроллера. Атрибут id – идентификатор состояния выхода, description – строковое значение состояния.
* Query – сущность, предоставляющая информацию о запросе к контроллеру. Атрибут id – идентификатор запроса, Qeury – строковое представление запроса для контроллера.
* 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.
* 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.

## 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;
* изменять диапазон значений выхода;

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.

# Проектирование программного средства

## 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 программной 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 хранений 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.

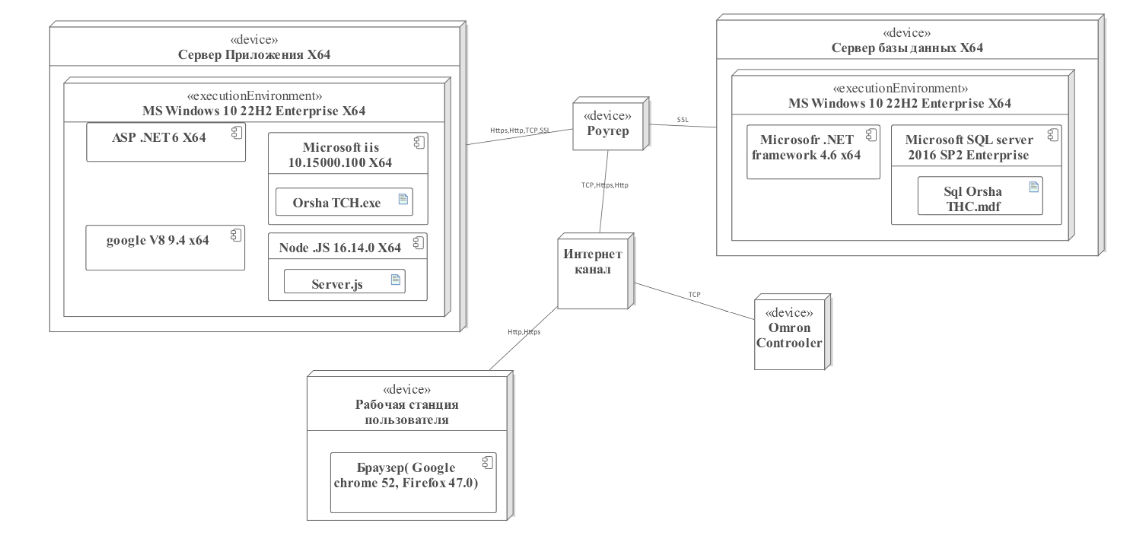


Рисунок 3.1 – Диаграмма развертывания

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.

## 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 архитектуру стоит начинать как можно раньше. Данный этап создания программного средства позволяет избежать большинства проблем во время разработки. К проблемам отнесится:

* низкая производительность. Не продуманная архитектура может привести к неэффективному использованию ресурсов, что может снизить производительность программы;
* сложность сопровождения. Если архитектура не продумана заранее, то код может быть запутанным и сложным для понимания, что затруднит его сопровождение и поддержку в будущем;
* 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;
* 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;
* 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.

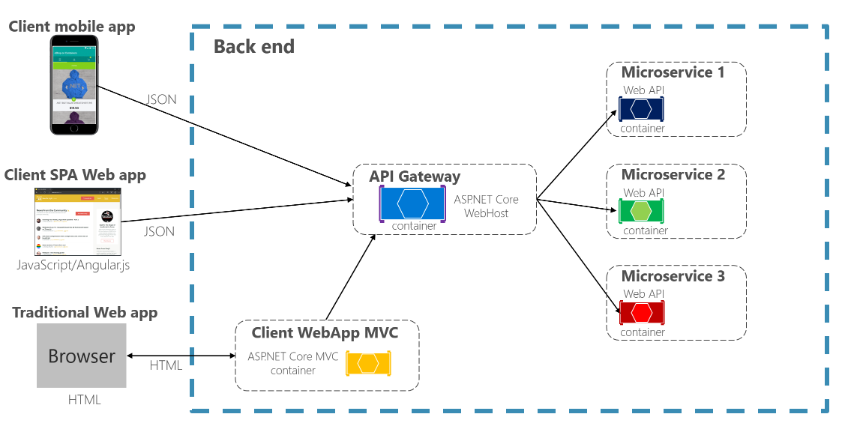


Рисунок 3.2 – API gateway шаблон проектирования.

## Проектирование 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:

* Users – таблица, хранящая информацию о всех типах пользователей, доступных системе. Атрибут u\_id – идентификатор пользователя, u\_password – пароль пользователя, u\_login – идентификатор пользователя в системе;
* users\_role – таблица, хранящая информацию о группе пользователей, имеющая следующие атрибуты: ut\_id- идентификатор группы, description- описание группы;
* m2m\_users\_role – таблица связка, использующаяся для хранения информации о связях между пользователем и его ролью. Атрибут u\_id – хранит идентификатор пользователя, ut\_id – хранит идентификатор роли пользователя.
* microcontroller – таблица, хранящая информацию о контроллерах. Атрибуты

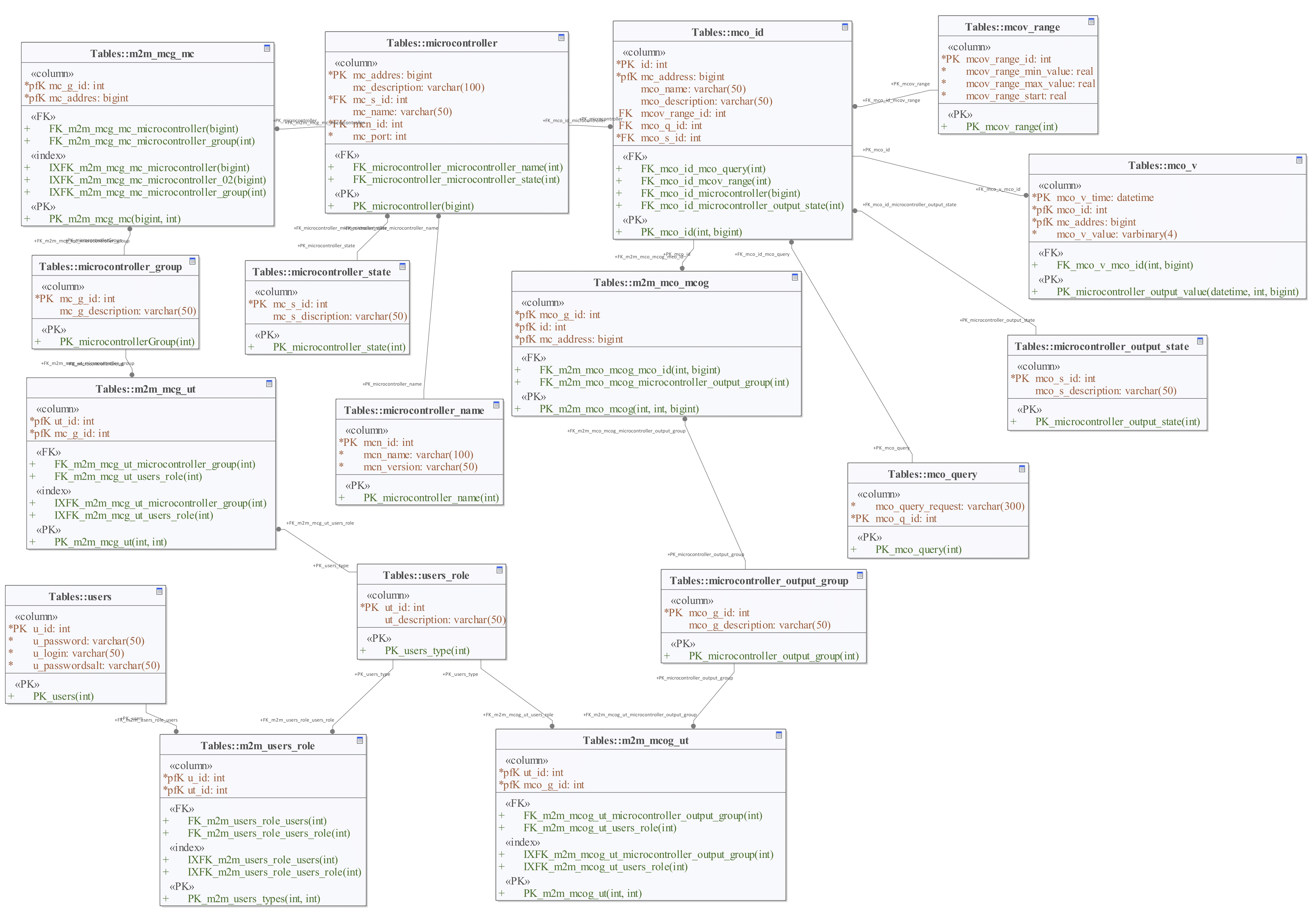


Рисунок 3.2 – Физическая модель базы данных

## 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 пользователе и выходы контроллеров по специальным группам. В ходе проектирование 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

## 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 пользователе 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 сериализации 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 сообщиться об ошибке.

На следующем этапе генерируется 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 к контроллеру

# 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.

## 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 данной программном средстве основным языком разработки был выбран С#. 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.

## 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.

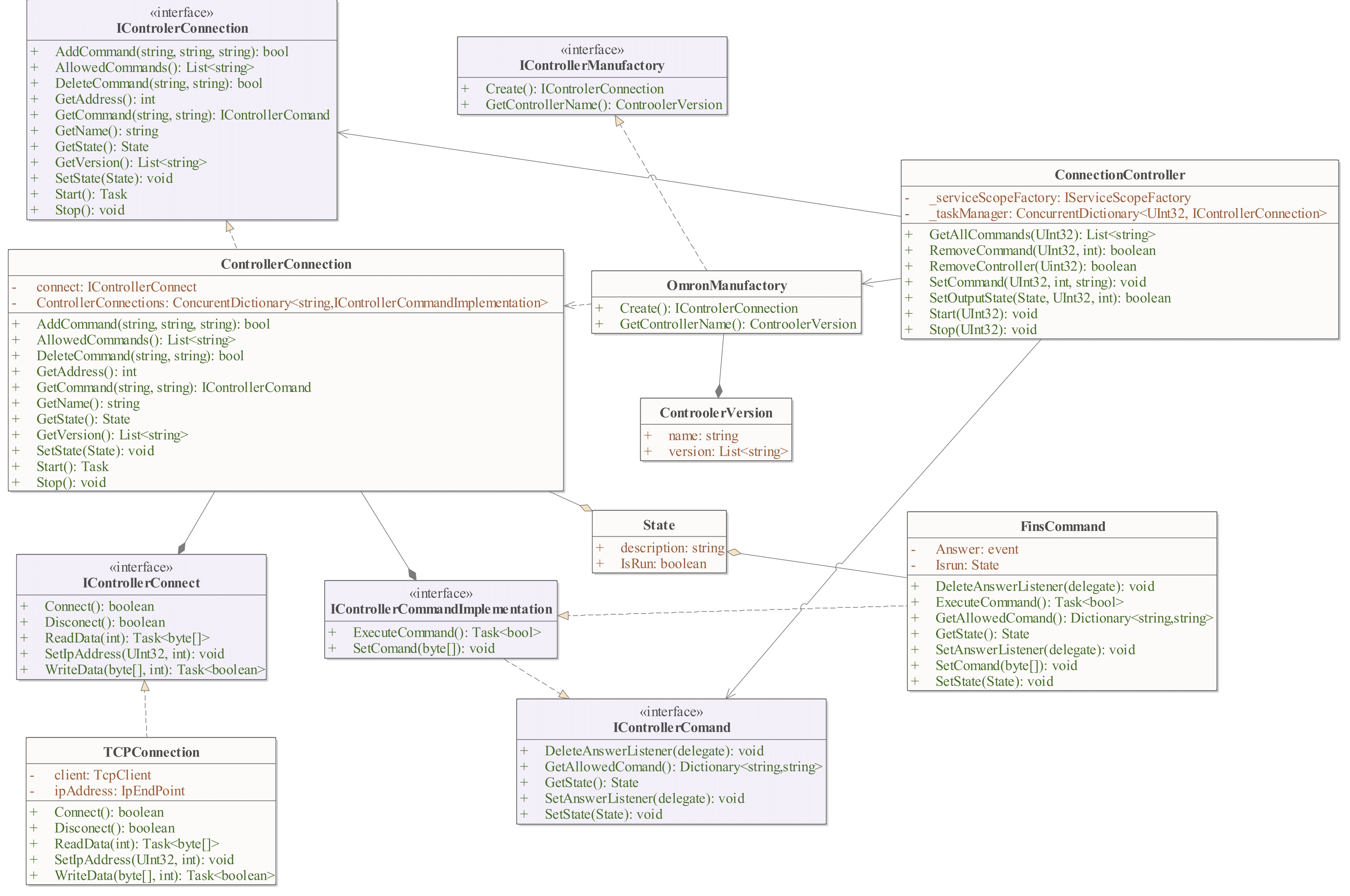


Рисунок 4.1 – Диаграмма классов логики обработки запросов к контроллеру

Для реализации соединения клиенткой и серверной части для передачи данных с контроллера была использована технология веб-сокетов. Данная технология, которая позволяет установить двустороннее соединения между браузером клиента и сервером через одно TCP соединение. Это позволяет передавать большие объемы данных от сервера клиенту без необходимости перегружать страницу на стороне пользователя. Данная технология отлично подходит для создания приложений для отслеживания параметров системы в реальном времени.

Для облегчения разработки приложения была выбрана вспомогательная библиотека от Microsoft – SignalR. SignalR использует технологии веб-сокетов, Server Sent Events и Long pooling. SignalR значительно упрощает разработку приложений, которые поддерживают множественное соединение с пользователем. Библиотека также предоставляет удобный инструментарий для обработки ошибок и управлением соединением.

Важной частью при разработке веб-приложения является создать удобный и понятный API, который обеспечит легкий доступ к данным. Для создания понятных для сервера запросов, клиент должен отправлять корректные данные на сервер. Для описания входных данных используется схема API описанная в таблице 4.1

Таблица 4.1 – Схема API

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название схемы | Параметр | Тип | Обязательность | Описание |
| Controller | Controller.ipAddress | int32 | Да |  |
| Controller.description | string | Нет |  |
| Controller.ipPort | int32 | Да |  |
| Controller.controllerStateId | int32 |  |  |
| Controller.controllerState |  |  |  |
| Controller.name | string |  |  |
| Controller.controlllerNameId | int32 |  |  |
| Controller.controlllerName |  |  |  |
| Controller.controllerGroups |  |  |  |
| Controller.m2m ControllerGroups |  |  |  |
| Controller.outputs |  |  |  |
| ControllerGroup | ControllerGroup.id | int32 | Да |  |
|  | ControllerGroup.groupDescription | string | Да |  |
|  | ControllerGroup.controllers |  |  |  |
|  | ControllerGroup.m2m controllers |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 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 удалилась |
| 8 Удаление пользователя из группы администраторов | Удаление пользователя из группы администраторов, обновление таблицы пользователей | Пользователь удален из группы администраторов, таблица с пользователями удалилась |
| 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.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

# Технико-экономическое обоснование разработки программного средства сбора и визуализации технологических параметров химического цеха на платформе .NET

## 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,
* гибкая настройка отчетов.

## Расчет затрат на разработку программного продукта

Для разработки данного программного продукта нам понадобиться 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

(7.1)

где n – количество работников, занятых выполнением работы;

Кпр – коэффициент, учитывающий процент премий;

Зч.*i*– часовая заработная плата i-го исполнителя, р.;

*ti*– 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.Инженер отдела качества | Ручное и автоматизированное тестирование программного средства | 1800 | 10,7 | 70 | 749 |
| 4.Проектный менеджер | Формирование требований и сроков разработки | 1800 | 10,7 | 50 | 535 |
| Итого | | | | | 5836 |
| Премия (25%) | | | | | 1459 |
| Всего основная заработная плата | | | | | 7295 |

Расчет затрат на дополнительную заработную плату разработчиков осуществляется по формуле:

, (7.2)

где Зо – затраты на основную заработную плату, р.;

Нд – норматив дополнительной заработной платы .

При нормативе дополнительной заработной платы равном 10%, получаем:

.

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

, (7.3)

где Нсоц – норматив отчислений от фонда оплаты труда .

В результате расчетов получаем следующее:

Расчет прочих затрат осуществляется по формуле:

, (7.4)

где Нпз – норматив прочих затрат .

При нормативе прочих затрат равном 110%, получаем:

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 | 7295 |
| 2. Дополнительная заработная плата разработчиков | 729,5 |
| 3. Отчисления на социальные нужды | 2776,477 |
| 4. Прочие затраты | 8024,5 |
| Общая сумма инвестиций в разработку | 18835,477 |

## 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, если организация освобождена от уплаты налога на прибыль, рассчитывается по формуле:

, (7.5)

где Ц – цена реализации ПО заказчику, р.;

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

НДС – сумма налога на добавленную стоимость, р;

Иразр – затраты на разработку программного обеспечения, р.

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

, (7.6)

где Нп  – ставка налога на прибыль, согласно действующему законодательству, (Нп = 18%).

Налог на добавленную стоимость определяется по формуле:

, (7.7)

где Ндс – ставка налога на добавленную стоимость согласно действующему законодательству, (Ндс = 20%).

При цене реализации равной 2000 рублей и ожидаемом количество проданных лицензий – 25 штук, получим:

С учетом налога на прибыль:

Уровень рентабельности рассчитывается по формуле:

, (7.8)

где П (Пч) – чистая прибыль, полученная от реализации ПО на рынке,р;

Иразр – затраты на разработку программного обеспечения, р.

Подставив значения в формулу, получим:

Проект является экономически эффективным, поскольку рентабельность затрат на разработку ПО не меньше средней процентной ставки по банковским депозитным вкладам.

Заключение

Список использованных источников

[1] FINS Commands: reference manual / Omron corporation – Kyoto, June 2001

[2] CJ2 CPU Unit Software: User’s Manual/ Omron corporation – Kyoto, July 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, December 2018.

[4]