

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра «Информационная безопасность систем и технологий»

Политехнический институт
Факультет информационных
технологий и электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность — 10.05.03 —

Информационная безопасность
автоматизированных систем

Специализация —

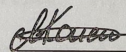
Защищённые автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

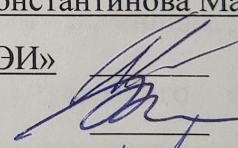
на тему

СИСТЕМА ЦЕНТРАЛИЗОВАННОГО УПРАВЛЕНИЯ УЗЛАМИ СЕТИ С
РЕАЛИЗАЦИЕЙ ФУНКЦИЙ ЗАЩИТЫ ОТ НЕСАНКЦИОНИРОВАННОГО
ДОСТУПА

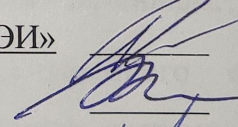
Студент

 Константинова Мария Денисовна

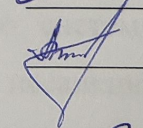
Руководитель нач. НИЛ-1 ф. «Аргус» АО «ПНИЭИ»

 Кваша Г.М.

Нормоконтролёр

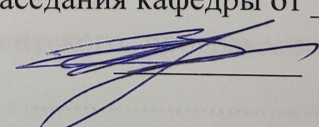
 Иванов А.П.

Рецензент нач. сектора АО «ПНИЭИ»

 Зорин А.Н.

Работа допущена к защите (протокол заседания кафедры от 28.06.21 № 16)

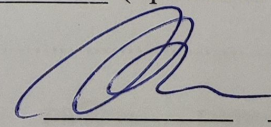
Заведующий кафедрой

 Зефиоров С.Л.

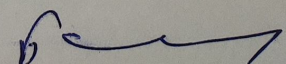
Работа защищена с оценкой хорошо (протокол заседания ГЭК от

28.06.21 № 11)

Секретарь ГЭК

 А.П. Иванов

Пенза 2021

Проверка 11 Самов


Оглавление

1 Анализ угроз для централизованной системы управления узлами сети из банка данных угроз ФСТЭК.....	6
2 Разработка сервера авторизации, работающего в режиме терминала.....	11
2.1 Разработка архитектуры сервера авторизации.	11
2.2 Выбор функции хэширования пароля.....	12
2.3 Разработка протокола взаимодействия сервера авторизации и клиентского программного обеспечения.....	14
2.4 Разработка алгоритма работы сервера авторизации.	16
2.4.1 Разработка алгоритма для программного модуля сервера, который отвечает за установку TCP/IP соединения.....	16
2.4.2 Разработка алгоритма для программного модуля сервера, который должен отвечать за обработку запросов клиента.	18
2.4.3 Разработка программного модуля, который должен отвечать за работу с базой данных авторизации.	27
2.4.4 Разработка алгоритма программного модуля сервера, который должен отвечать за работу сервера в режиме терминала.....	30
3 Разработка клиентского графического программного обеспечения «Настройка сети», реализующего централизованную систему управления узлами сети с реализацией функций защиты от несанкционированного доступа.....	33
3.1 Разработка архитектуры клиентского программного обеспечения «Настройка сети».	33
3.2 Разработка алгоритма работы программного обеспечения «Настройка сети».	34
3.2.1 Разработка алгоритма работы главной функции программного обеспечения «Настройка сети».	34

3.2.2 Разработка алгоритма работы функции чтения файлов, необходимых для работы программы.....	36
3.2.3 Разработка алгоритма работы функций графического интерфейса.....	37
3.3 Разработка пользовательского графического интерфейса для программного обеспечения «Настройка сети».....	53
3.3.1 Разработка шаблонов пользовательского графического интерфейса для программного обеспечения «Настройка сети».....	53
3.3.2 Разработка программной реализации для шаблонов пользовательского графического интерфейса.....	59
3.3.3 Построение диаграммы классов для разработанного графического программного обеспечения «Настройка сети».....	65
4 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.....	70
4.1 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации.....	70
4.2 Методы и средства физической культуры, снижающие воздействие психофизических факторов, влияющих на психоэмоциональное состояние специалиста по защите информации.....	74
Заключение	79
Список используемых источников.....	80
Приложение А. Исходные файлы сервера авторизации, отвечающие за работу в терминале	81
Приложение Б. Исходные файлы сервера авторизации, отвечающие за обработку запросов	86

Приложение В. Исходные файлы сервера авторизации, отвечающие за работу с базой данных авторизации	97
Приложение Г. Исходный файл графического программного обеспечения «Настройка сети», отвечающие за установку соединения с сервером авторизации.....	100
Приложение Д. Исходные файлы графического программного обеспечения «Настройка сети», отвечающие за работу главного окна	103
Приложение Е. Исходные файлы графического программного обеспечения «Настройка сети», отвечающие за работу диалоговых окон.....	122
Приложение Ж. Исходные файлы графического программного обеспечения «Настройка сети», отвечающие за криптографические преобразования....	138
Приложение И. Исходные файлы графического программного обеспечения «Настройка сети», отвечающие за работу функций настройки сети.....	139

Введение

В современном развитии мире невозможно представить работу организаций без использования локальных сетей. Локальная сеть осуществляет возможность совместного доступа к информационным ресурсам, таким как: компьютерные программы и различные виды данных. У коллектива людей, работающих над одним проектом, появляется возможность работать с одними и теми же объектами и средствами информатизации. Локальная сеть предоставляет возможность работать с одним и тем же оборудованием, что значительно упрощает, например, задачу печати.

Так же у локальной сети есть и административная функция, которая значительно облегчает задачу контроля действий с различных устройств в сети по сравнению с администрированием каждого автономного компьютера в отдельности.

Наиболее распространённым и удобным способом администрирования локальной сети является система централизованного управления сетью. Но такая система требует особого подхода к обеспечению информационной безопасности, ведь уязвимость в её реализации вносит уязвимость во всю локальную сеть в целом. [1]

На основании данных выводов была определена актуальность темы выпускной квалификационной работы – «Система централизованного управления узлами сети с реализацией функций защиты от несанкционированного доступа».

1 Анализ угроз для централизованной системы управления узлами сети из банка данных угроз ФСТЭК

Из банка данных угроз ФСТЭК были выделены основные угрозы информационной безопасности для системы централизованного управления сетью:

УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными. Угроза заключается в возможности осуществления нарушителем несанкционированного доступа к передаваемой системе защищаемой информации за счёт деструктивного воздействия на протоколы сетевого/локального обмена.

Источники угрозы: внутренний нарушитель с низким потенциалом, внешний нарушитель с низким потенциалом.

Последствия реализации угрозы: нарушение конфиденциальности, нарушение целостности, нарушение доступности. [2]

УБИ.104: Угроза определения топологии вычислительной сети. Угроза заключается в возможности определения нарушителем состояния сетевых узлов дискредитируемой системы для получения сведений о топологии дискредитируемой вычислительной сети, которые могут быть использованы в дальнейшем при попытках реализации других угроз.

Источник угрозы: внешний нарушитель с низким потенциалом. Последствия реализации угрозы: нарушение конфиденциальности. [2]

УБИ.131: Угроза подмены субъекта сетевого доступа. Угроза заключается в возможности осуществления нарушителем несанкционированного доступа к защищаемым данным пользователей сети или проведения различных мошеннических действий путём скрытой подмены в отправляемых дискредитируемым пользователем сетевых запросах сведений об отправителе сообщения.

Источники угрозы: внешний нарушитель со средним потенциалом.
Последствия реализации угрозы: нарушение конфиденциальности, нарушение целостности. [2]

УБИ.140: Угроза приведения системы в состояние «отказ в обслуживании». Угроза заключается в возможности отказа дискреционной системой в доступе легальным пользователям при лавинообразном увеличении числа сетевых соединений с данной системой или при использовании недостатков реализации сетевых протоколов.

Источники угрозы: внутренний нарушитель с низким потенциалом, внешний нарушитель с низким потенциалом.

Последствия реализации угрозы: нарушение доступности. [2]

УБИ.178: Угроза несанкционированного использования системных сетевых утилит. Угроза заключается в возможности осуществления нарушителем деструктивного программного воздействия на систему за счёт использования имеющихся или предварительно внедрённых стандартных сетевых утилит, предназначенных для использования администратором для диагностики и обслуживания системы.

Источники угрозы: внутренний нарушитель с низким потенциалом, внешний нарушитель с низким потенциалом.

Последствия реализации угрозы: нарушение конфиденциальности, нарушение целостности, нарушение доступности. [2]

По результатам проведённого анализа угроз были разработаны меры для защиты от несанкционированного доступа:

а) должен быть разработан сервер авторизации. На стороне сервера авторизации, должна находиться база данных авторизации в текстовом виде. Такое расположение базы данных повышает безопасность хранимой в ней информации, дополнительно ограничивая к ней доступ посторонних лиц. При такой архитектуре системы сервер авторизации служит посредником между пользователем и базой данных;

б) организация двухфакторной авторизации на основании пароля и на основании IP-адреса;

в) в целях обеспечения информационной безопасности локальной сети, были разработаны роли безопасности авторизованного пользователя программного обеспечения. Описание ролей безопасности программного обеспечения «Настройка сети» представлено в таблице 1;

Таблица 1 – Описание ролей безопасности программного обеспечения «Настройка сети»

Название роли безопасности	Описание роли безопасности
Системный администратор	Имеет право подключаться к другим узлам сети, сбрасывать и менять их IP-адреса, проверять возможность подключения данного узла к выбранному списку адресов. Системный администратор может входить в систему только со своего устройства, что уменьшает возможность подключения к системе другого пользователя от имени администратора. Системный администратор имеет право добавлять пользователей в систему, удалять из неё пользователей и менять их права.
Локальный администратор	Имеет право сбрасывать и менять свой IP-адрес, проверять возможность подключения данного узла к выбранному списку адресов. Локальный администратор может входить в систему только со своего устройства, что уменьшает возможность подключения к системе другого пользователя от имени администратора. Локальный администратор

Продолжение таблицы 1

Название роли безопасности	Описание роли безопасности
	имеет право добавлять пользователей в систему, удалять из неё пользователей и менять их права. Локальный администратор не имеет права присваивать другим пользователям уровень прав выше, чем уровень «Локальный администратор»
Локальный пользователь	Имеет право проверять возможность подключения данного устройства к выбранному списку адресов. Локальный пользователь может входить в систему только со своего устройства, что уменьшает вероятность реализации угрозы определения топологии сети.

г) в целях обеспечения безопасности хранения пароля в базе данных авторизации исключено его хранение в открытом виде;

д) для обеспечения конфиденциальности визуальной информации, вводимый пользователем пароль должен быть скрыт для просмотра.

Была составлена таблица соответствий угроз безопасности информации и нейтрализующих их защитных мер. Эта информация представлена в таблице 2.

Таблица 2 – Таблица соответствий угроз безопасности информации в разрабатываемой системе и нейтрализующих их защитных мер

Угроза ИБ	Защитная мера
УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	б

Продолжение таблицы 2

Угроза ИБ	Защитная мера
УБИ.104: Угроза определения топологии вычислительной сети.	в
УБИ.131: Угроза подмены субъекта сетевого доступа.	б
УБИ.140: Угроза приведения системы в состояние «отказ в обслуживании»	а, б, в
УБИ.178: Угроза несанкционированного использования системных сетевых утилит.	а, б, г, д

2 Разработка сервера авторизации, работающего в режиме терминала

2.1 Разработка архитектуры сервера авторизации

На стороне сервера авторизации, который необходимо разработать, находится база данных авторизации в текстовом виде. В этой базе данных должна храниться информация об именах пользователей, их правах доступа, паролях в виде хэша и об их IP, с которых пользователю разрешён вход. Сервер авторизации должен уметь выполнять обработку базы данных, изменять информацию в ней и добавлять новую.

Сервер авторизации должен получать запросы от клиентского программного обеспечения «Настройка сети», обрабатывать эти запросы и отправлять клиенту ответ. В целях сохранения конфиденциальности передаваемой информации передача пароля в открытом виде между клиентом сервером запрещена. Для осуществления данного запрета одной из функций сервера должна являться генерация соли, для создания хэш-суммы пароля на стороне клиента и сравнения полученного значения с хэш-суммой пароля, который хранится на сервере.

Программа сервера авторизации должна состоять из нескольких логических автономных модулей.

Первый логический модуль программы должен отвечать за TCP/IP соединение сервера и клиента. В этом блоке выполняются подключение сервера к сети, обработка подключений клиентов к серверу, приём клиентских запросов.

Второй программный модуль должен отвечать за обработку запросов клиента. В этом блоке должны быть реализованы все функции сервера, предусмотренные протоколом взаимодействия сервера и клиента.

Третий программный модуль должен отвечать за работу с базой данных авторизации. В этом блоке должны быть реализованы все основные функции работы с базой данных, такие как: чтение базы данных, добавление

информации в базу данных (БД), удаление информации из БД и изменение информации в БД.

Четвёртый логический модуль программы должен отвечать за работу сервера в режиме терминала. Этот блок должен включать обработку параметров работы сервера и выдачу справки.

2.2 Выбор функции хэширования пароля

В целях обеспечения безопасности хранения пароля в базе данных авторизации исключено его хранение в открытом виде. Для этого необходимо использовать функцию хэширования. На данный момент существует несколько функций хэширования, которые применяются для хранения пароля в защищённом виде.

Криптоанализ хэш-функций нацелен на исследование уязвимостей к различным видам атак, таких как:

- нахождение коллизий (ситуация, когда двум различным исходным сообщениям соответствует одно и то же хэш-значение);
- нахождение прообраза (исходного сообщения по его хэш-сумме).

От устойчивости хэш-функции к нахождению коллизий зависит безопасность электронной цифровой подписи с использованием данного хэш-алгоритма. От устойчивости к нахождению прообраза зависит безопасность хранения хэш-сумм паролей для целей аутентификации.

SHA-1 – алгоритм криптографического хэширования, который генерирует для входного значения произвольной длины выходное значение, размер которого составляет 20 байт. Данный алгоритм реализует функцию, основанную на идее сжатия.

При решении методом «грубой силы»:

Задача нахождения коллизий требует порядка 2^{80} операций, при использовании атаки Дней рождения.

Задача нахождения прообраза требует 2^{160} операций. [3]

SHA-256 – криптографический алгоритм, основанный на методике Меркла-Дамгарда. Массив информации делится на блоки, состоящие из 16-ти частей. После выполнения 64 перемешиваний, система выводит сумму, на которой будет основана обработка следующего блока. Процесс непрерывен и взаимосвязан. Абсолютно невозможно получить код отдельного криптоблока, не имея данных о его предшественнике. Данный алгоритм вычисления хэш-суммы обладает очень высокой криптографической стойкостью, но при этом он является очень медленным и занимает много места в базе данных. Размер хэш-подписи блока – 32 байт. [3]

CRC – циклический алгоритм нахождения контрольной суммы. Порождающий полином является одним из основных параметров данного алгоритма. С ним связан и другой параметр — степень полинома, которая определяет количество битов, которые будут использоваться для вычисления значения контрольной суммы. Наиболее распространены слова длиной в 8, 16 и 32 бита, что связано с особенностями современной архитектуры вычислительной техники. Самый популярный и рекомендуемый IEEE полином для CRC-32 используется в Ethernet, FDDI; также этот многочлен является генератором кода Хемминга. Математическая сложность алгоритма CRC очень небольшая, что является его большим преимуществом, так как его использование в программе является оптимальным как по времени, так и по памяти. Однако сам алгоритм не обладает высокой криптографической стойкостью. Для увеличения сложности криптоанализа перед вычислением хэш-суммы к открытому тексту добавляется соль.[3]

Для создания хэша пароля была выбрана функция CRC32, так как эта функция обладает достаточно высокой криптографической стойкостью от подбора по радужным таблицам и при этом не требует высокой вычислительной мощности. Для повышения криптографической стойкости было принято решение добавлять к паролю соль перед вычислением хэша. Соль должна генерироваться на стороне сервера и передаваться клиенту в случае необходимости вычисления хэш-суммы.

2.3 Разработка протокола взаимодействия сервера авторизации и клиентского программного обеспечения

Согласно базовой эталонной модели взаимодействия открытых систем (ГОСТ Р ИСО/МЭК 7498-1-99) [4], сетевая модель OSI состоит из 7 уровней:

- прикладной уровень (application layer);
- представительский уровень (presentation layer);
- сеансовый уровень (session layer);
- транспортный уровень (transport layer);
- сетевой уровень (network layer);
- канальный уровень (data link layer);
- физический уровень (physical layer).

Для осуществления диалога между клиентом и сервером, был разработан протокол прикладного уровня OSI. Протокол представлен в таблице 3.

Таблица 3 – Протокол взаимодействия сервера авторизации и клиентского программного обеспечения

Клиент (запрос)	Сервер (ответ)
0x00 – авторизация, N длина имени пользователя (4 байт), пароль в виде хэша (8 байт), имя пользователя (N байт).	0x00 – успех + код доступа: 0x00 – пользователь, 0x01 – локальный администратор, 0x02 – системный администратор.
	0x01 – ошибка работы сервера
	0x02 – ошибка авторизации по паролю
	0x03 – ошибка авторизации по IP

Продолжение таблицы 3

Клиент (запрос)	Сервер (ответ)
0x01 – смена пароля, N длина имени пользователя (4 байт), старый пароль (8 байт), новый пароль (8 байт), имя пользователя (N байт).	0x00 – успех
	0x01 – ошибка работы сервера
	0x02 – неверный старый пароль
0x02 – добавление нового пользователя, N длина имени пользователя (4 байт), пароль в виде хэша (8 байт), код доступа (1 байт), IP-адрес (2 байта), имя пользователя (N байт).	0x00 – успех
	0x01 – ошибка работы сервера
	0x02 – пользователь с таким именем уже существует
0x03 – смена прав пользователя, N длина имени пользователя (4 байт), код доступа (1 байт), имя пользователя (N байт).	0x00 – успех
	0x01 – ошибка работы сервера
0x04 – удаление пользователя, N длина имени пользователя (4 байт), пароль администратора в виде хэша (8 байт), имя пользователя (N байт).	0x00 – успех
	0x01 – ошибка работы сервера
0x05 – генерация соли	Соль (4 байта)
0x06 – вывод списка всех пользователей системы	Количество пользователей (4 байта). Для каждого пользователя: длина имени пользователя N (4 байта), имя пользователя (N байт).

Продолжение таблицы 3

Клиент (запрос)	Сервер (ответ)
0x07 – инициализация. Данная операция необходима для проверки корректности соединения	Слово «Ok»
0x08 – сообщение о закрытии клиентского приложения	

2.4 Разработка алгоритма работы сервера авторизации

2.4.1 Разработка алгоритма для программного модуля сервера, который отвечает за установку TCP/IP соединения

Был разработан алгоритм для программного модуля сервера, который отвечает за установку TCP/IP соединения. Для реализации такого рода соединения был использован сокет – механизм коммуникации, который позволяет клиент-серверным системам работать как локально, на одной машине, так и по сети.

Сокеты, в отличие от пайпов, создают чёткое различие между клиентом и сервером. Этот механизм способен поддерживать соединение с несколькими клиентами одновременно на одном сервере. [5]

Сначала программа сервера создаёт сокет, который как файловый дескриптор представляет собой ресурс, присеваемый единственному серверному процессу. [5]

Далее сервер присваивает сокету имя, у сетевых сокетов имя файла будет идентификатором сервиса, относящегося к определённой сети, к которой могут подключаться клиенты. Именуется сокет с помощью системного вызова bind. [5]

Далее серверный процесс ждёт подключения клиента к именованному сокету. Системный вызов listen формирует очередь входящих подключений.

Сервер может принять их с помощью системного вызова `accept`. Когда сервер вызывает `accept`, создаётся новый сокет, отличающийся от именованного сокета. Этот новый сокет применяется только для взаимодействия с данным клиентом. Именованный сокет сохраняется для дальнейших подключений других клиентов. [5]

Для работы сервера был использован потоковый тип сокета. Потоковые сокеты (в чём-то подобные стандартным потокам ввода/вывода) обеспечивают соединение, представляющее собой последовательный и надёжный двунаправленный поток байтов. Следовательно, гарантируется, что без указания возникшей ошибки данные не будут потеряны, продублированы или переупорядочены. Сообщения большого объема фрагментируются, передаются и снова собираются воедино. Это напоминает файловый поток, который принимает большие объёмы данных и делит их на меньшие блоки для записи на физический диск. У потоковых сокетов предсказуемое поведение. Сокеты такого типа реализованы на базе протоколов TCP/IP. [5]

В качестве домена создаваемого сокета был выбран домен `AF_INET` – интернет-протоколы ARPA (Advanced Research Projects Agency, управление перспективных исследований и разработок). [5]

Сокеты домена `AF_INET` могут применяться программами, взаимодействующими в сетях на базе протоколов TCP/IP, включая Интернет. Сокеты домена `AF_INET` связаны с номером IP-порта. Системный вызов `bind` присваивает адрес неименованному сокету. [5]

После установления соединения с клиентом сервер помещает работу с клиентом в новый поток, который сервер сам создаёт. Полученные от клиента запросы серверная программа передаёт в обработчик запросов.

Весь вышеизложенный алгоритм представлен в виде блок-схемы на рисунке 1 и на плакате «Алгоритм работы сервера авторизации».

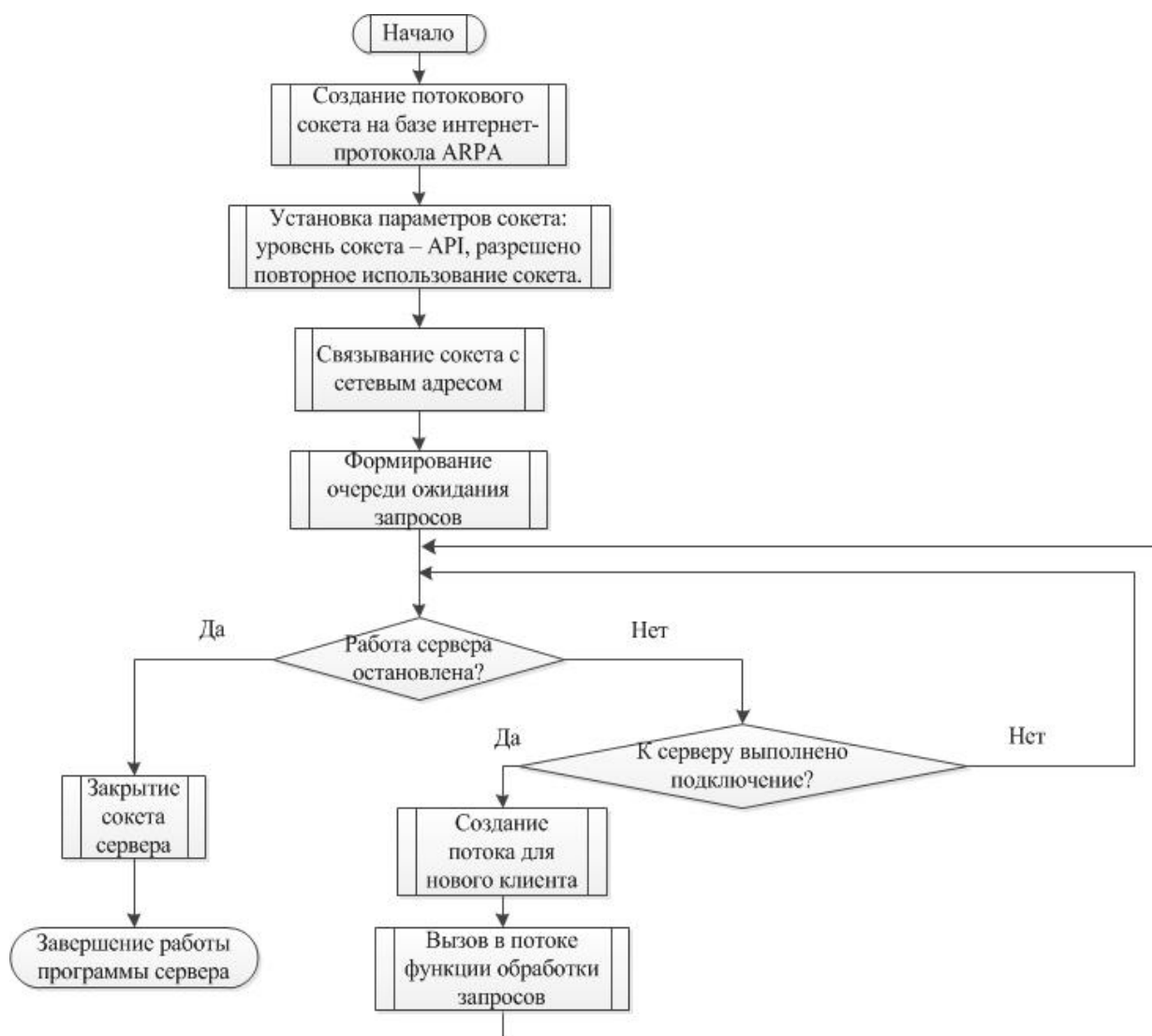


Рисунок 1 – Алгоритм установки TCP/IP соединения сервера с клиентами

2.4.2 Разработка алгоритма для программного модуля сервера, который должен отвечать за обработку запросов клиента

Был разработан алгоритм для функции обработки запросов клиента.

В ходе выполнения алгоритма обработки запросов функция принимает идентификатор операции. Затем в соответствии с номером идентификатора обработчик вызывает необходимую функцию обработки.

Алгоритм работы функции обработки запросов клиента в виде блок-схемы представлен на рисунке 2 и на плакате «Алгоритм работы сервера авторизации».

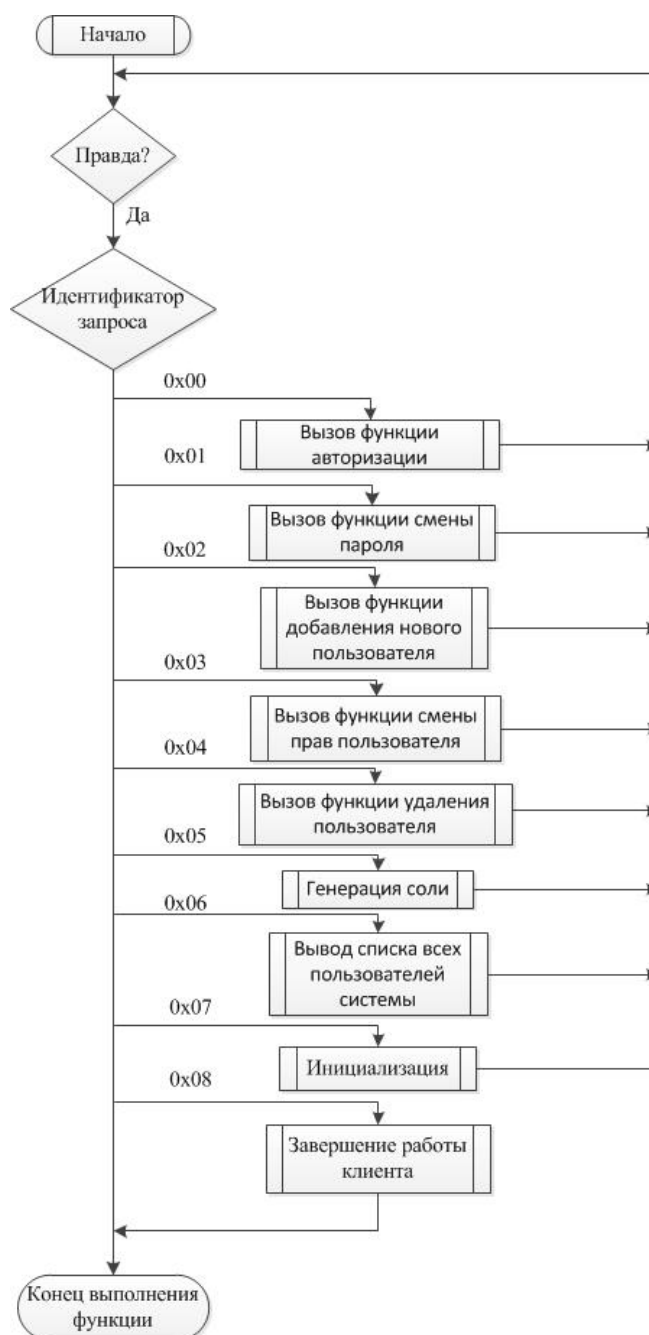


Рисунок 2 - Алгоритм работы функции обработки запросов клиента

Был разработан алгоритм для функции сервера, которая должна отвечать за обработку запросов на авторизацию.

Функция авторизации является основной защитной функцией от несанкционированного доступа в разрабатываемом программном обеспечении. Наличие данной функции в разрабатываемой системе управления ущемлами сети обуславливает наличие в данной системе сервера авторизации.

Данная функция включает в себя два основных пункта обеспечения безопасности разрабатываемого графического обеспечения. В ней реализуются оба фактора авторизации: по паролю и по IP-адресу. Пароль в закрытом виде сервер получает из заголовка, который клиент отправляет на сервер авторизации сам.

Сервер получает IP-адрес из информации о клиенте, с которым установлено соединение. Данная информация располагается в заголовке пакетов клиента.

В начале выполнения алгоритма сервер получает от клиента заголовок. Затем серверная программа производит чтение информации из базы данных. Если во время чтения БД произошла ошибка, то сервер отправляет клиенту код ошибки работы сервера, и функция завершается с ошибкой.

Если в базе данных не было обнаружено пользователя с таким именем, то сервер отправляет клиенту код ошибки авторизации по паролю и функция авторизации завершается.

Если хэш-сумма пароля пользователя не совпадает с хэш-суммой пароля этого пользователя из базы данных авторизации, то сервер отправляет клиенту код ошибки авторизации по паролю и функция авторизации завершается.

Если IP клиента не совпадает с IP для данного пользователя из БД авторизации, то сервер отправляет клиенту код ошибки авторизации по IP и функция авторизации завершается.

После успешного прохождения всех проверок сервер отправляет клиенту код успешного завершения авторизации и код доступа для данного авторизованного пользователя. Функция авторизации завершается, возвращая код успешной работы.

Алгоритм работы функции авторизации в виде блок-схемы представлен на рисунке 3 и на плакате «Алгоритм работы сервера авторизации».

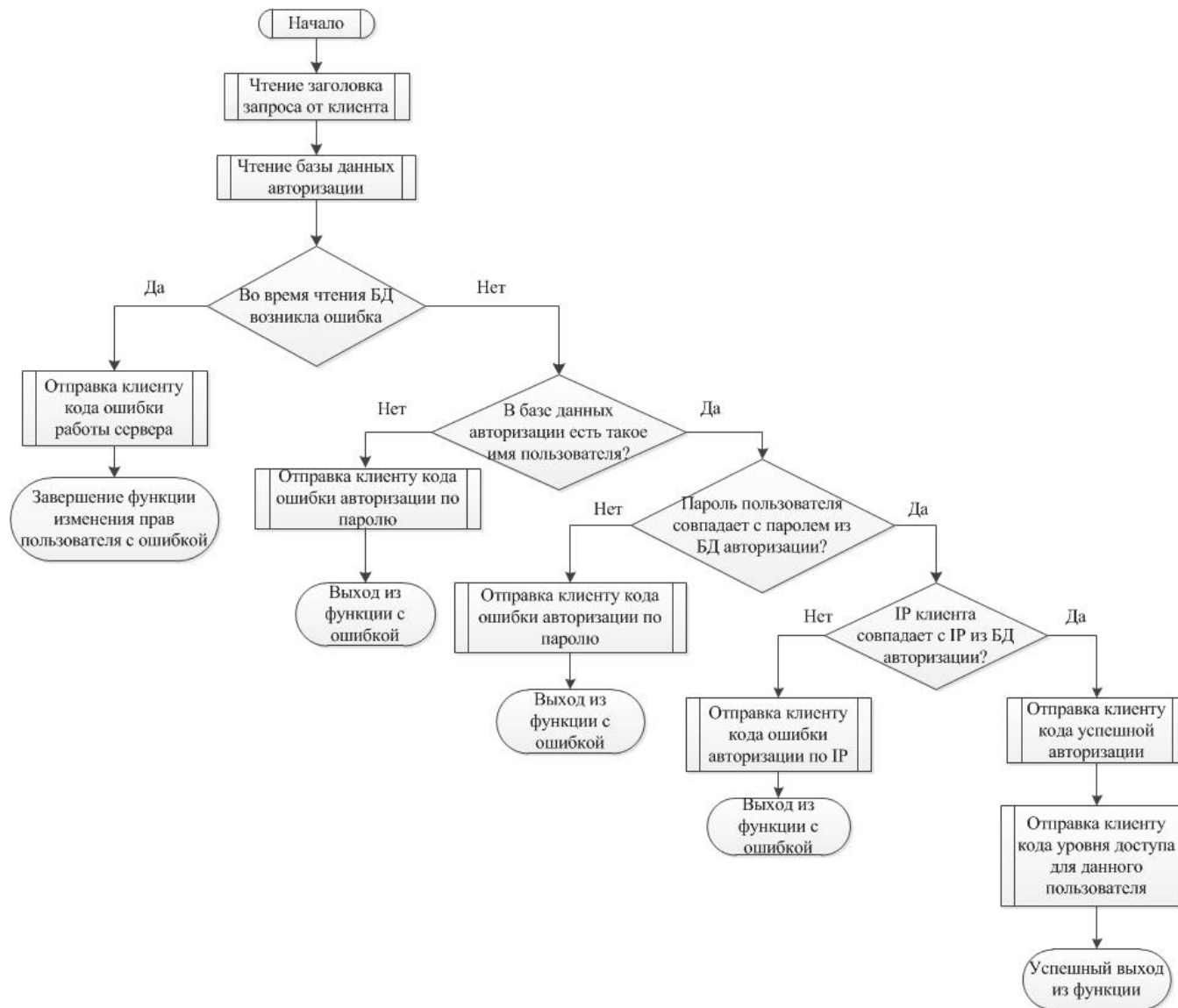


Рисунок 3 – Алгоритм авторизации на стороне сервера

Был разработан алгоритм для функции сервера, которая должна отвечать за обработку запросов на добавление нового пользователя.

В начале работы серверная программа читает заголовок клиента, в котором получает информацию о новом пользователе. Затем происходит чтение базы данных авторизации. Если во время чтения БД произошла ошибка, то сервер отправляет клиенту код ошибки работы сервера, и функция завершается с ошибкой.

Если в базе данных авторизации уже есть пользователь с таким именем, то сервер отправляет клиенту код ошибки повторного использования имени пользователя. Функция завершается с ошибкой.

Сервер осуществляет запись в БД авторизации, которая включает в себя всю необходимую информацию о новом пользователе. Функция добавления нового пользователя завершается успешно.

Алгоритм работы функции добавления нового пользователя в виде блок-схемы представлен на рисунке 4.

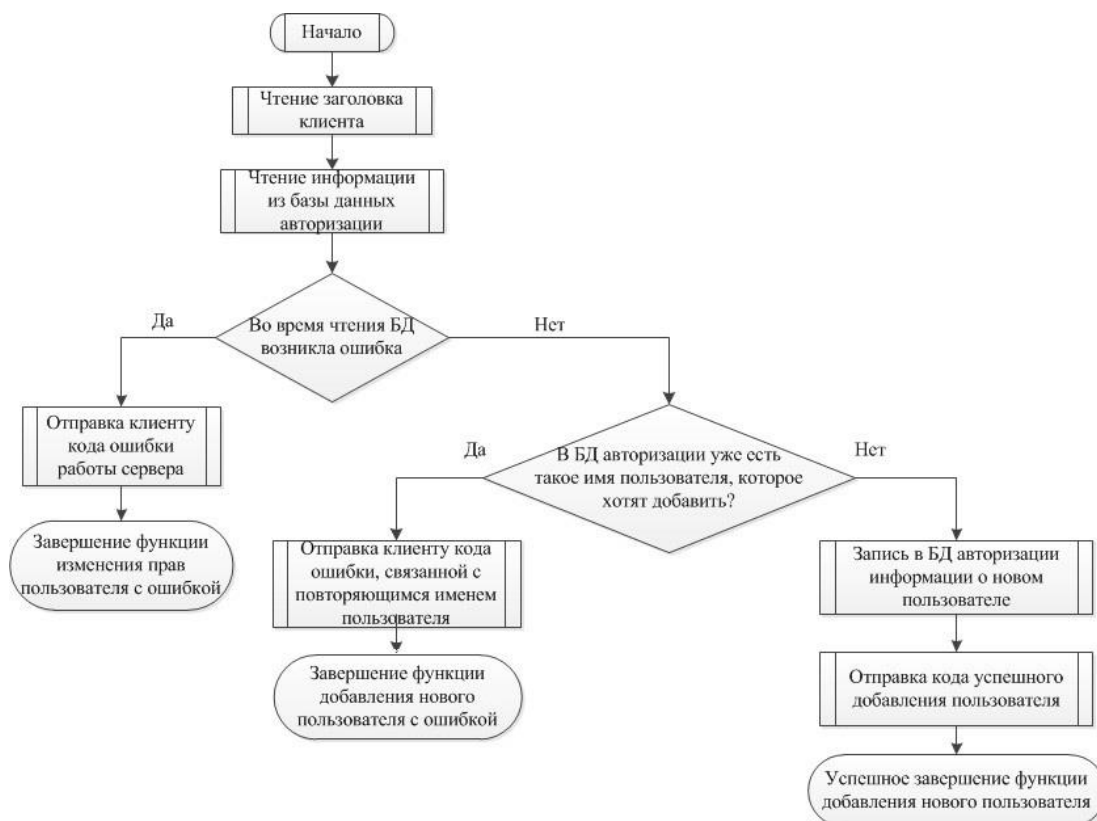


Рисунок 4 – Алгоритм добавления нового пользователя на сервере

Был разработан алгоритм для функции сервера, которая должна отвечать за обработку запросов на изменение прав пользователя.

В начале работы функции сервер осуществляет чтение заголовка клиента, в котором находится информация об имени пользователя и его новом коде доступа к системе. Затем происходит чтение базы данных авторизации. Если во время чтения БД произошла ошибка, то сервер отправляет клиенту код ошибки работы сервера, и функция завершается с ошибкой.

Сервер осуществляет полную перезапись БД авторизации с учётом новых прав пользователя. Сервер отправляет клиенту код успешного выполнения операции, и функция изменения прав пользователя завершается успешно.

Алгоритм работы функции изменения прав пользователя в виде блок-схемы представлен на рисунке 5.

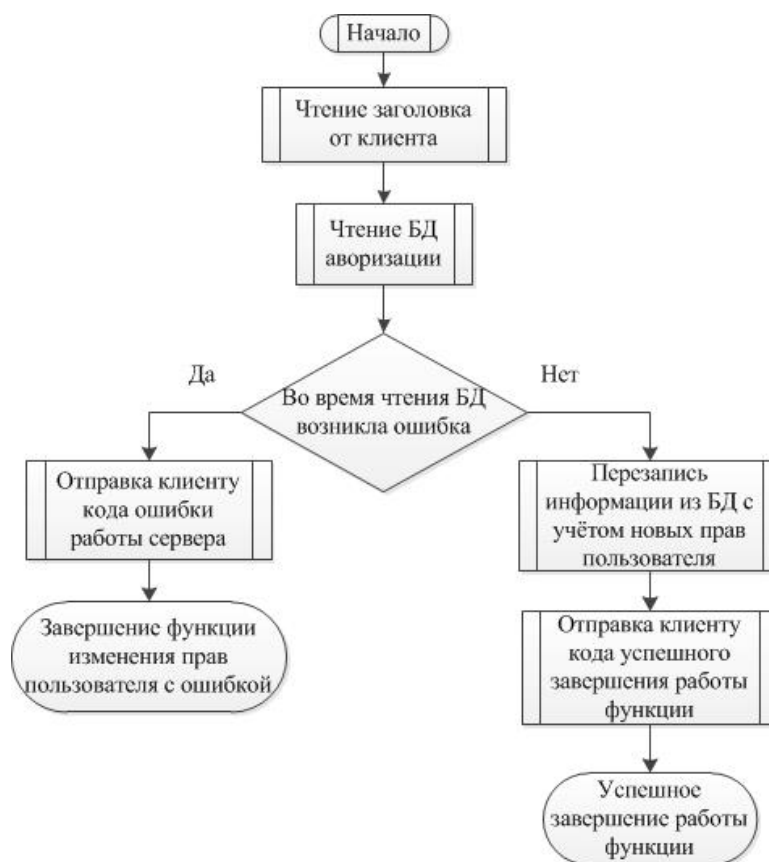


Рисунок 5 – Алгоритм работы функции изменения прав пользователя на сервере

Был разработан алгоритм для функции сервера, которая должна отвечать за обработку запросов на смену пароля пользователя. В начале работы функции сервер осуществляет чтение заголовка клиента, в котором находится информация об имени пользователя, о его старом пароле и его новом пароле в системе. Затем происходит чтение базы данных авторизации. Если во время чтения БД произошла ошибка, то сервер отправляет клиенту код ошибки работы сервера, и функция завершается с ошибкой. Сервер осуществляет полную перезапись БД авторизации с учётом нового пароля пользователя. Сервер отправляет клиенту код успешного выполнения операции, и функция смены пароля пользователя завершается успешно.

Алгоритм работы функции смены пароля пользователя в виде блок-схемы представлен на рисунке 6.

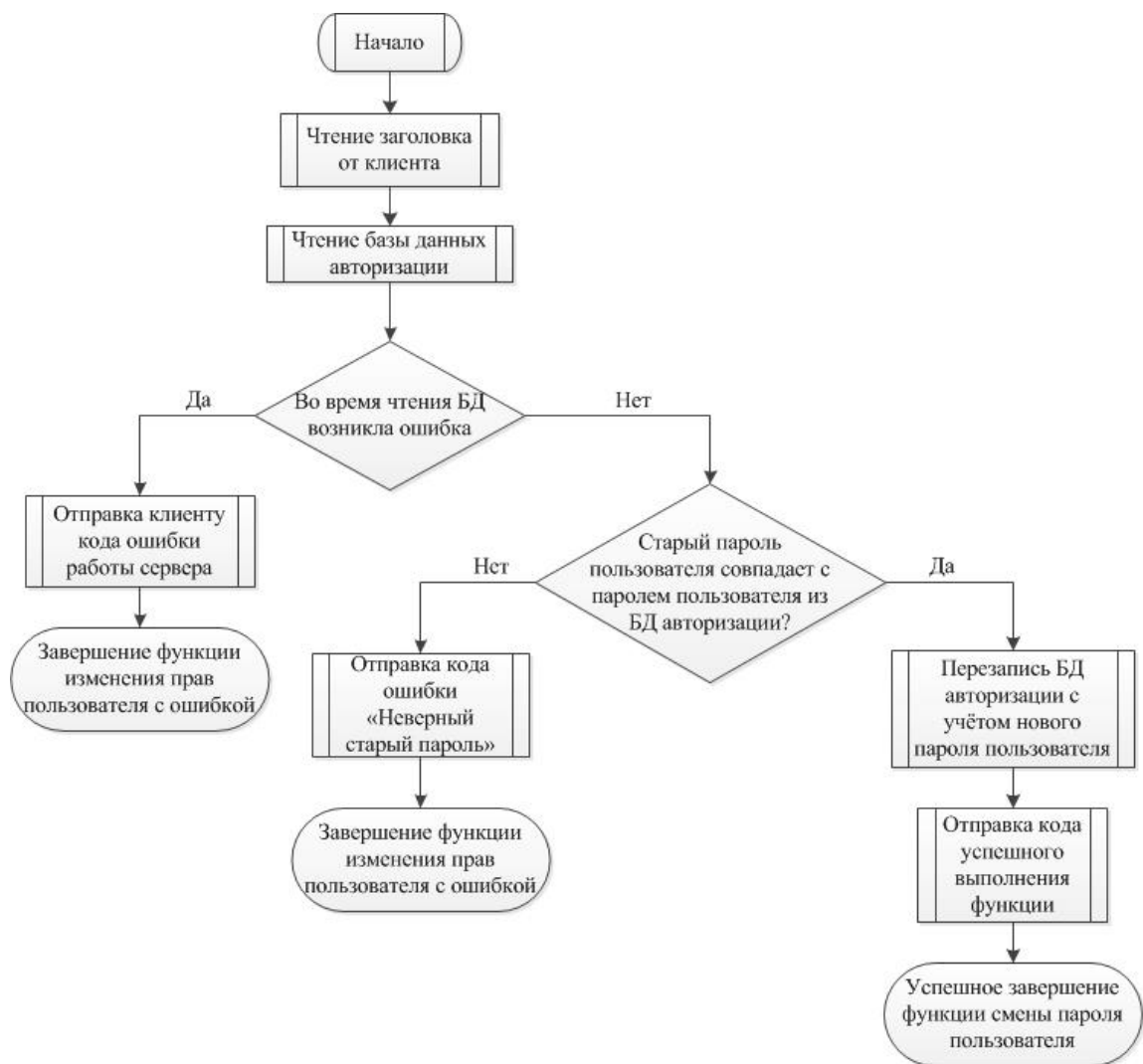


Рисунок 6 – Алгоритм смены пароля пользователя на сервере

Был разработан алгоритм для функции сервера, которая должна отвечать за удаление пользователя из системы управления локальной сетью.

В начале работы функции сервер осуществляет чтение заголовка клиента, в котором находится информация об имени пользователя. Затем происходит чтение базы данных авторизации. Если во время чтения БД произошла ошибка, то сервер отправляет клиенту код ошибки работы сервера, и функция завершается с ошибкой.

Сервер осуществляет полную перезапись БД авторизации с учётом исключения удаляемого пользователя. Сервер отправляет клиенту код успешного выполнения операции, и функция смены пароля пользователя завершается успешно.

Алгоритм работы функции удаление пользователя из системы управления локальной сетью в виде блок-схемы представлен на рисунке 7.

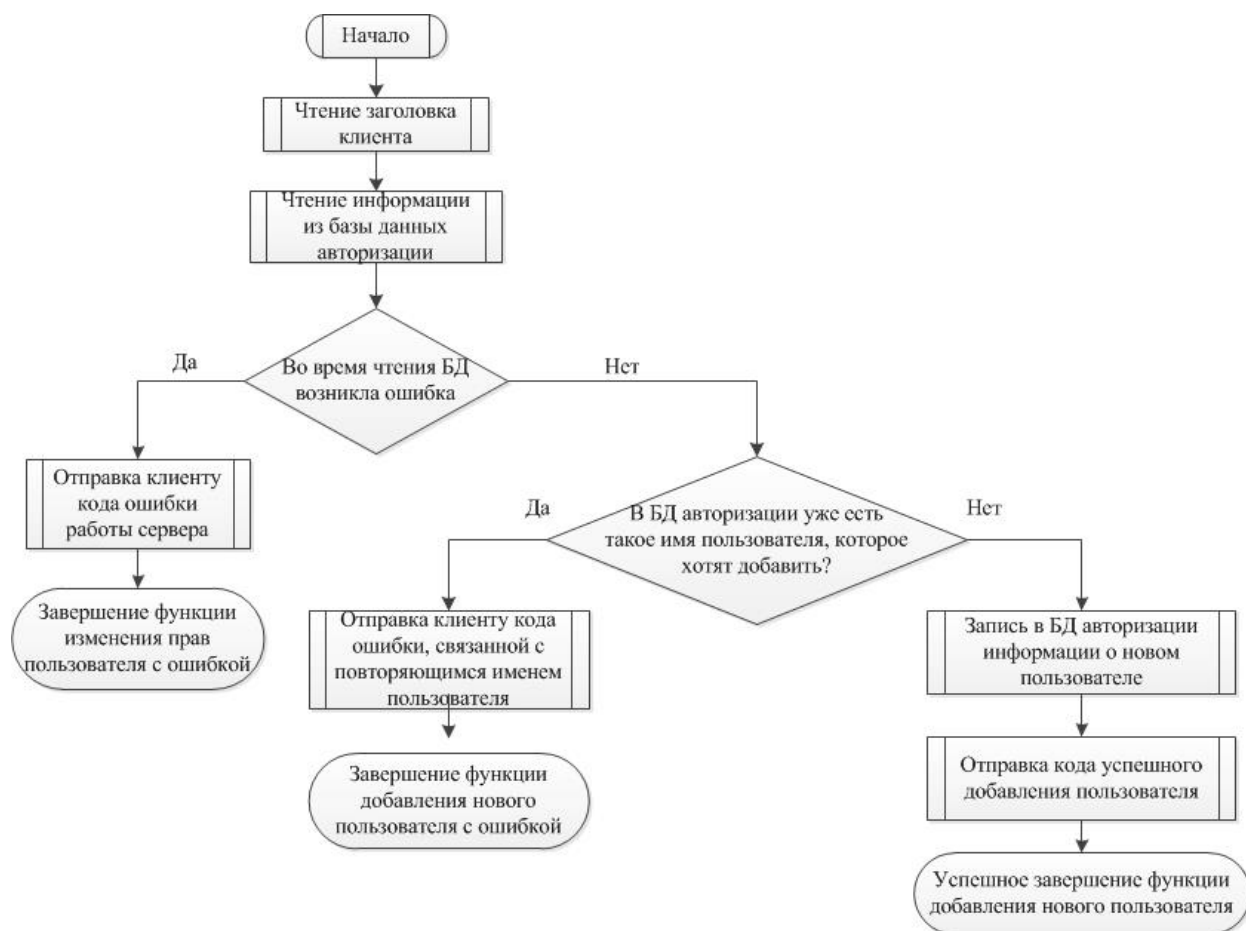


Рисунок 7 – Алгоритм работы функции удаления пользователя на стороне клиента

Был разработан алгоритм для функции сервера, которая должна отвечать за генерацию соли.

В начале работы функции выполняется задание константного символического значения размером 4 байта. Затем это значение передаётся клиенту. Функция генерации соли завершается успешно.

Алгоритм работы функции генерации соли в виде блок-схемы представлен на рисунке 8 и на плакате «Алгоритм работы сервера авторизации».



Рисунок 8 – Алгоритм работы функции генерации соли

Был разработан алгоритм для функции сервера, которая должна отвечать за получение списка всех пользователей системы. Данная функция необходима для возможности получения всегда актуальной информации.

В начале работы функции происходит чтение базы данных авторизации. Если во время чтения БД произошла ошибка, то сервер отправляет клиенту код ошибки работы сервера, и функция завершается с ошибкой.

Затем сервер сообщает клиенту о количестве пользователей системы. Пока не закончатся все пользователи системы, сервер отправляет клиенту длину имени пользователя, а затем имя этого пользователя. Когда список пользователей заканчивается, функция получения имён всех пользователей успешно завершается.

Алгоритм работы функции получения полного списка пользователей системы в виде блок-схемы представлен на рисунке 9.

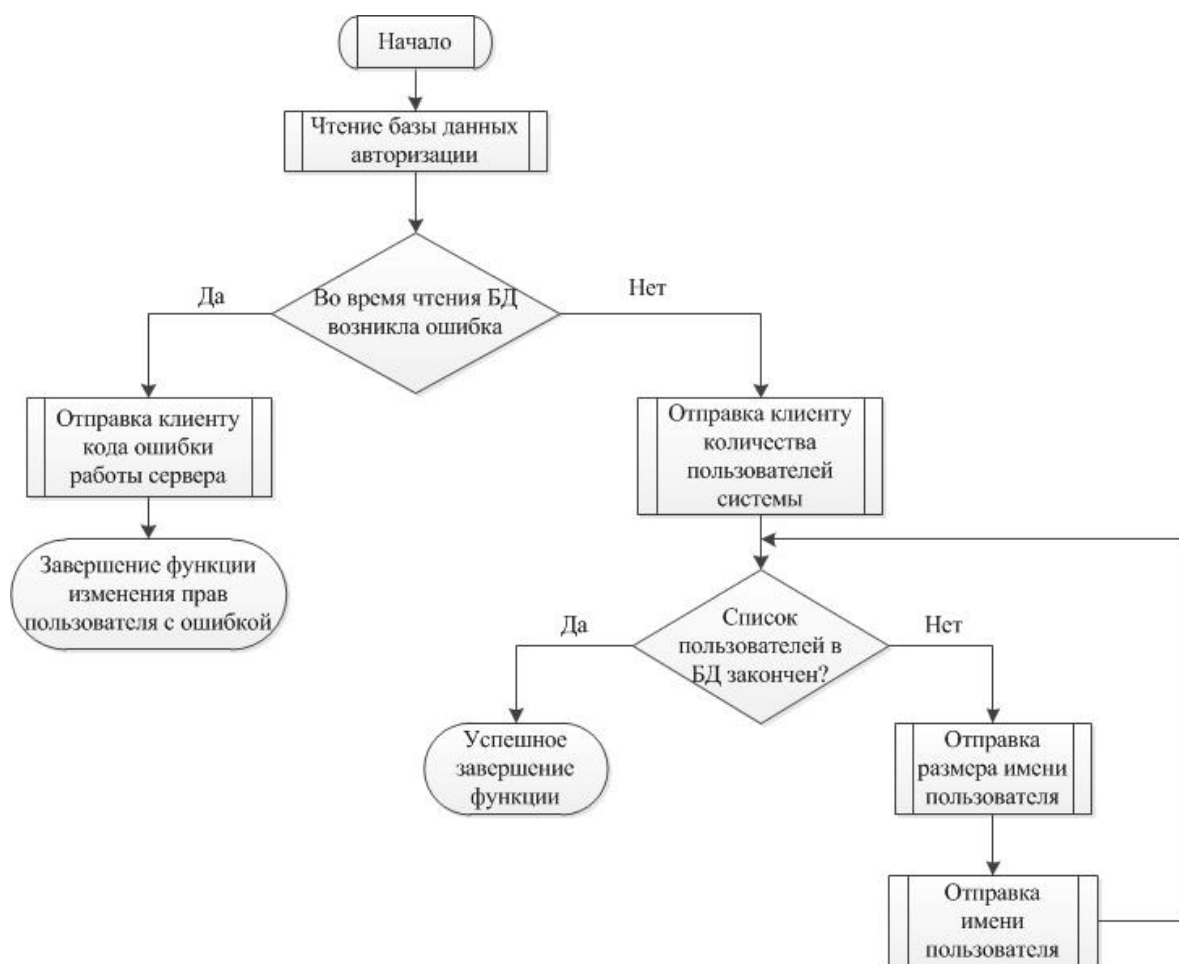


Рисунок 9 – Алгоритм работы функции получения списка пользователей системы

2.4.3 Разработка программного модуля, который должен отвечать за работу с базой данных авторизации

Для корректной работы с базой данных авторизации была разработана её структура. БД авторизации представляет собой текстовый файл, информация в котором размещается по определённому принципу: в одной строке содержится информация только об одном пользователе. Каждая строка БД имеет вид, представленный на рисунке 10.



Рисунок 10 – Структура строки в БД авторизации

Каждый элемент строки базы данных разделён пробелом. К элементам строки БД предъявляются следующие требования:

- имя пользователя должно быть написано латинскими буквами. Допускаются как заглавные, так и строчные буквы;
- к допустимым символам имени относятся цифры от 0 до 9 и символ «_»;
- хэш-сумма должна представлять собой строку, состоящую из 32 символов;
- допустимыми символами хэш-суммы являются цифры от 0 до 9 и строчные латинские буквы;
- для доступа пользователя в системе представляет собой один символ, представляющий собой цифру в диапазоне от 0 до 2;
- IP адрес представляет собой строку, состоящую из цифр, разделённых точками. Максимальная длина строки адреса – 15 символов.

Был разработан алгоритм работы функции чтения базы данных.

В начале работы функции чтения БД выполняется очистка всех сохранённых ранее данных о БД. Затем осуществляется открытие файла БД в потоке на чтение. Если файл открыт некорректно, функция завершается с ошибкой.

Выполняется проверка, является ли текущая строка файла последней строкой. Пока не закончится файл, производится чтение одной строки из него.

Выполняется парсинг полученной строки в заранее подготовленную структуру данных для данного файла. Если парсинг прошёл с ошибкой, файловый поток закрывается и выполнение функции завершается с ошибкой. Когда чтение доходит до конца файла, файловый поток закрывается, и функция успешно завершается.

Алгоритм работы функции чтения БД в виде блок-схемы представлен на рисунке 11 и на плакате «Алгоритм работы сервера авторизации».

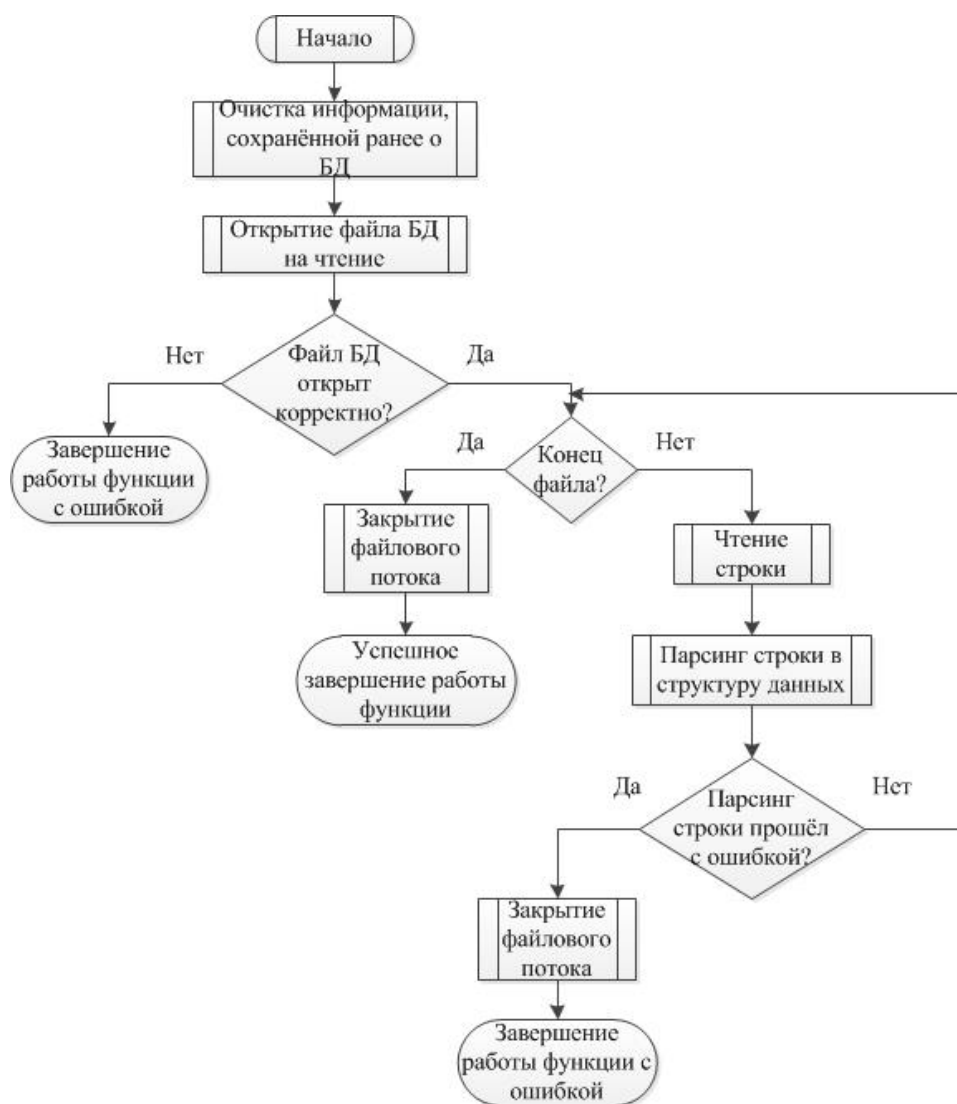


Рисунок 11 – Алгоритм работы функции чтения БД авторизации

Был разработан алгоритм работы функции записи в базу данных. В начале работы алгоритма файл БД авторизации открывается в файловом потоке на запись. Если во время открытия файла произошла ошибка, функция записи завершается с ошибкой.

Производится последовательная запись через пробел имени пользователя, хэш-суммы пароля пользователя, кода доступа и допустимого IP адреса для данного пользователя. После успешной записи выполняется закрытие файлового потока. Функция записи завершается, возвращая код успешной работы.

Алгоритм работы функции записи в БД в виде блок-схемы представлен на рисунке 12.

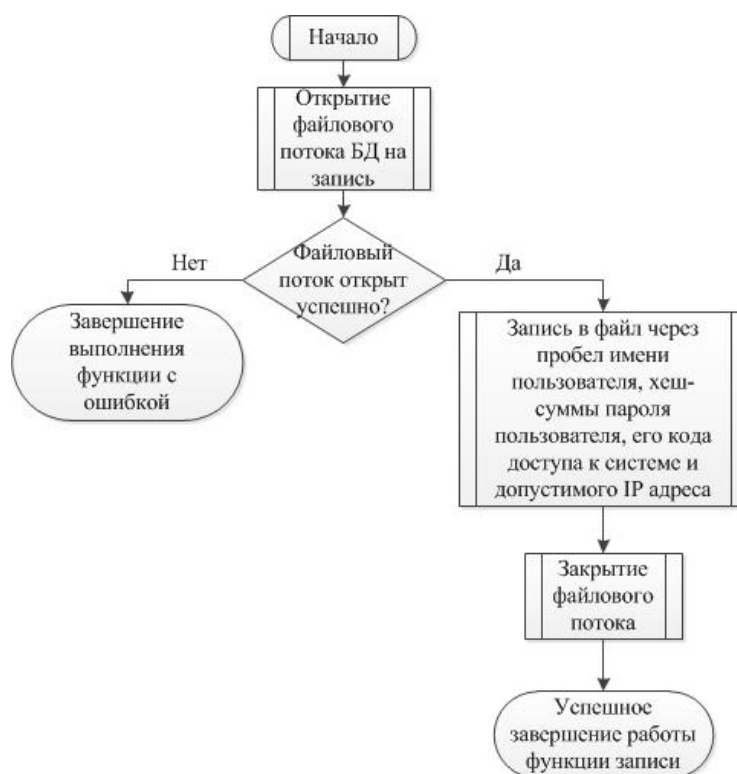


Рисунок 12 – Алгоритм записи в БД авторизации

2.4.4 Разработка алгоритма программного модуля сервера, который должен отвечать за работу сервера в режиме терминала

Программа сервера должна осуществлять работу в режиме терминала. Для корректной работы программы, в терминальном режиме должны быть реализованы возможности запуска программы с различными параметрами.

Были разработаны параметры для терминального запуска, они представлены в таблице 4.

Таблица 4 – Параметры работы сервера в режиме терминала.

Параметр	Значение параметра
--port или -p	Порт для установки соединения между сервером и клиентом. По умолчанию значение порта равно 5000.
--authorization_file или -a	Путь к файлу базы данных авторизации с указанием имени

Параметр	Значение параметра
	файла. По умолчанию значение этого параметра: «./log.ini».

Был разработан алгоритм работы функции разбора параметров программы. В начале выполнения функции параметрам работы программы присваиваются их значения по умолчанию. Затем осуществляется получение параметров, введенных пользователем. Пока не будут прочитаны все введенные параметры, каждый параметр сравнивается со списком допустимых имён. Если такой параметр предусмотрен программой, его значение присваивается параметру вместо значения по умолчанию.

Алгоритм работы функции разбора параметров программы из терминала в виде блок-схемы представлен на рисунке 13.

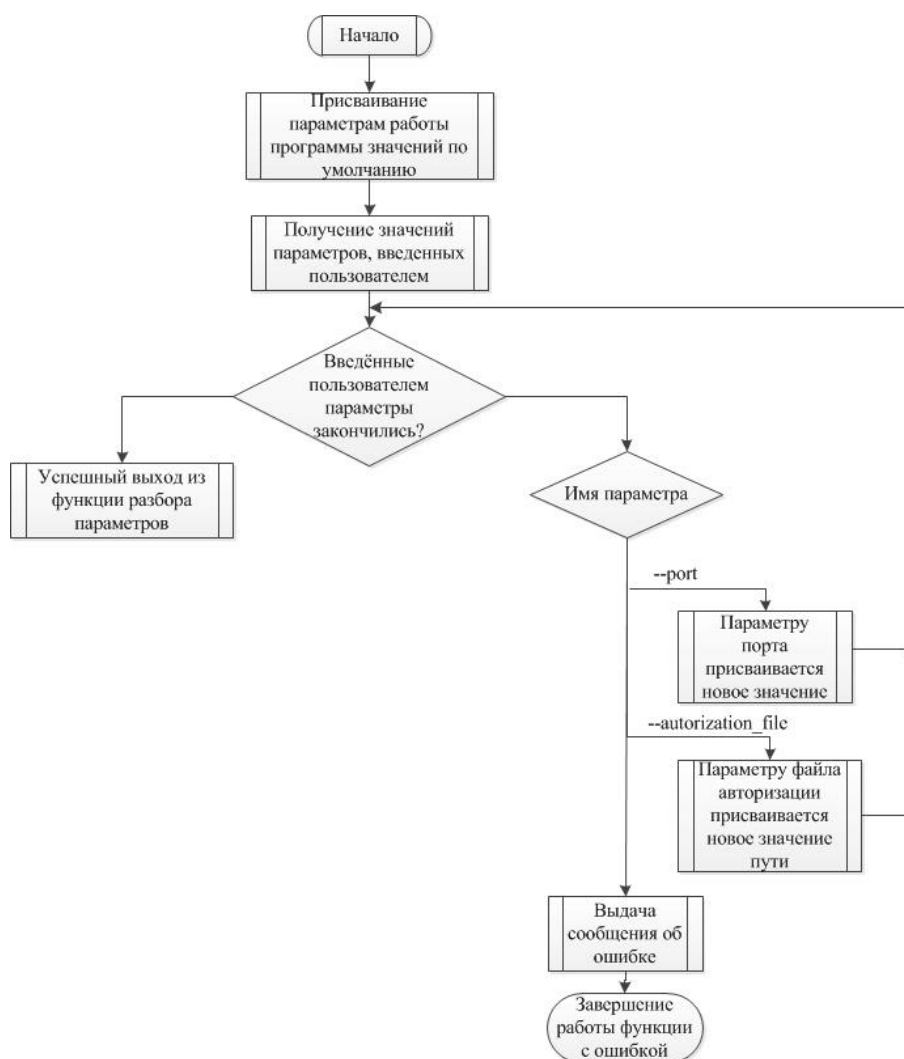


Рисунок 13 – Алгоритм разбора параметров программы из терминала

2.5 Разработка программного обеспечения сервера авторизации

На основании разработанных алгоритмов и протокола взаимодействия сервера авторизации с клиентом было написано программное обеспечение для сервера авторизации. В качестве языка разработки был выбран высокоуровневый язык C++. Исходный код программного обеспечения сервера авторизации представлен в приложениях А-В.

Работа программы сервера представлена на рисунке 14.

```
mary@debian:/media/work/server$ ./server
IP client: 10.0.11.113
initalize
go to gen salt
go to autorize
```

Рисунок 14 – Работа программы сервера

Для демонстрации взаимосвязей представленных в приложениях файлов были построены диаграммы связей исходных файлов и классов. Диаграммы связей были построены с использованием программного обеспечения Doxygen. Полученные диаграммы связности файлов представлены на рисунках 15 и 16.

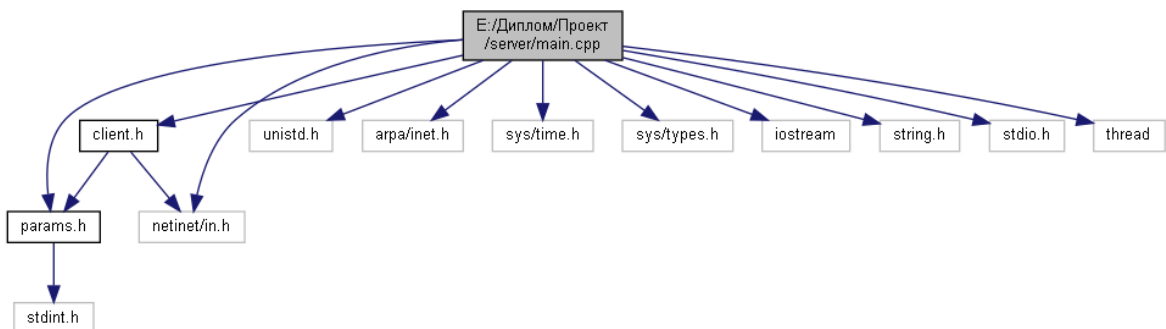


Рисунок 15 – Диаграмма связности файлов с main.cpp

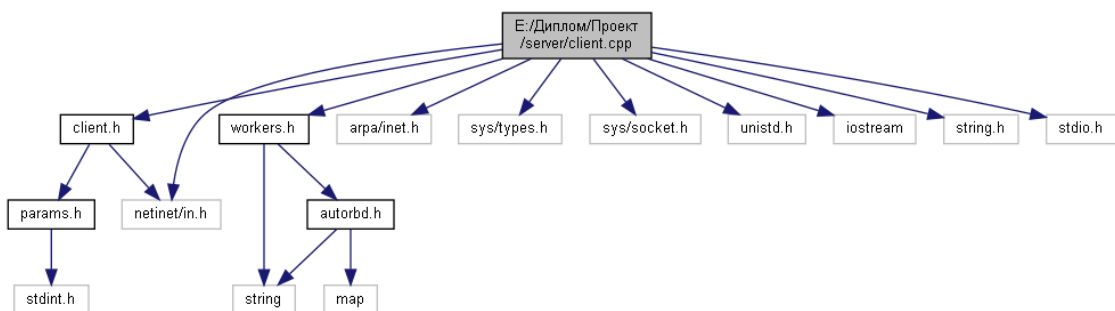


Рисунок 16 - Диаграмма связности файлов с client.cpp

3 Разработка клиентского графического программного обеспечения «Настройка сети», реализующего централизованную систему управления узлами сети с реализацией функций защиты от несанкционированного доступа

3.1 Разработка архитектуры клиентского программного обеспечения «Настройка сети»

Разрабатываемое клиентское программное обеспечение предназначено для настройки локальной сети организации. «Настройка сети» является инструментом, использовать который могут как системные администраторы, так и простые пользователи сети.

Для реализации функций, связанных с использованием различных конфигураций узла на стороне клиента был размещён текстовый файл инициализации со всеми разработанными для данной сети конфигурациями. В начале выполнения программы осуществляется чтение файла инициализации для последующего использования этих данных в функциях настройки сети.

Так же на стороне клиента располагается текстовый файл, который хранит в себе информацию об IP адресах всех устройств сети. Эта информация считывается программой для реализации возможности подключения системного администратора к другим устройствам сети.

Для внедрения ролей безопасности в программное обеспечение, было принято решение реализовать в нём многопользовательский режим. Для функционирования этого режима, в программном обеспечении должны быть реализованы следующие инструменты.

Для осуществления всех функций многопользовательского режима клиенту необходимо выполнять запросы к серверу, так как все эти функции требуют обращения к базе данных авторизации, которая хранится на

серверной стороне. Протокол взаимодействия между пользователем и клиентом представлен в разделе 3.3.

3.2 Разработка алгоритма работы программного обеспечения «Настройка сети»

3.2.1 Разработка алгоритма работы главной функции программного обеспечения «Настройка сети»

Работа программы начинается с запуска главной функции `main.cpp`. В начале работы этой функции выполняется вызов функции чтения файла инициализации, в котором хранятся все конфигурации узлов сети. Если чтение этого файла прошло с ошибкой, программа выдаёт пользователю сообщение об этом и завершается с кодом неуспеха.

Далее выполняется вызов функции чтения файла, в котором хранятся соответствия названий устройств и адресов в сети. Если чтение этого файла прошло с ошибкой, программа выдаёт пользователю сообщение об этом и завершается с кодом неуспеха.

Выполняется создание потокового сокета для работы в сети ARPA. Подробнее технология сокетов описана в пункте 2.4.1. Если процесс создания сокета произошла неуспешно, программа завершается с ошибкой.

Далее клиентской программе задаётся IP адрес для работы в сети, соответствующий IP адресу устройства, на котором работает программа-клиент. Клиенту задаётся номер порта и устанавливается соединение с сервером.

Для проверки корректности соединения, клиентская программа отправляет серверу код инициализации. Если ответное сообщение сервера соответствует значению, заданному протоколом, описанным в пункте 2.3, то запускается графическая часть программы. После завершения работы графического интерфейса, сокет клиента закрывается, а программа успешно завершается.

Если ответное сообщение сервера не соответствует заданному значению, сокет клиента завершается, и программа завершается с ошибкой.

Алгоритм работы главной функции программы «Настройка сети» в виде блок-схемы представлена на рисунке 17.

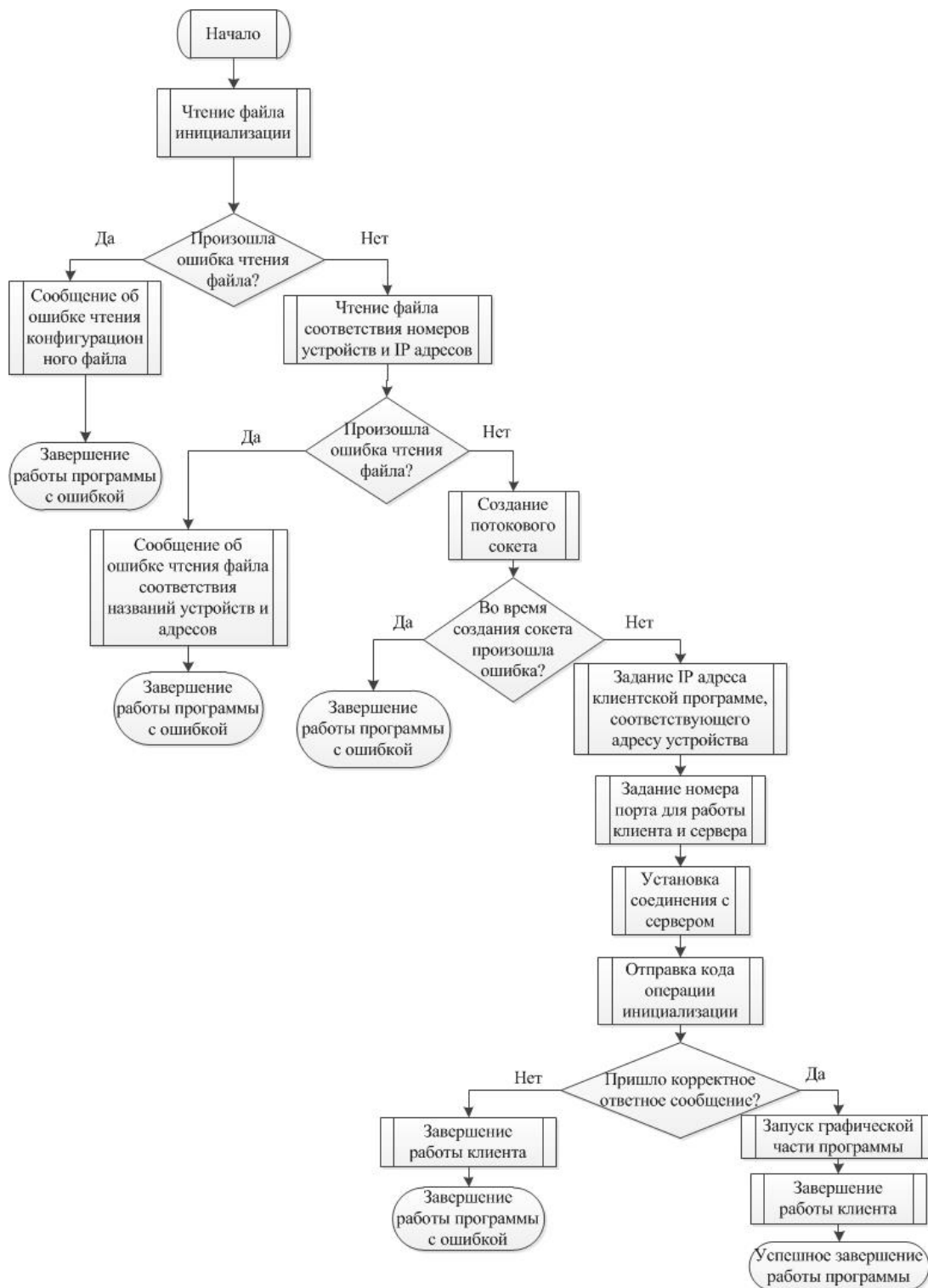


Рисунок 17 – Алгоритм работы главной функции программы «Настройка сети»

3.2.2 Разработка алгоритма работы функции чтения файлов, необходимых для работы программы

Данная функция предназначена для чтения двух типов файлов, которые могут понадобиться для работы графического программного обеспечения «Настройка сети».

Первый тип файла – это файл конфигурации. В этом файле располагаются все возможные конфигурации локальной сети, предусмотренные системным администратором. Данные в файле располагаются следующим образом: название конфигурации в отдельной строке в квадратных скобках. В последующих строках названия интерфейсов узла и их адреса. Второй тип файла – файл соответствия адресов. В этом файле располагаются имена узлов данной локальной сети и соответствующие им адреса.

В начале работы функции чтения происходит открытие заданного файла в файловом потоке для чтения. Если открытие файла прошло некорректно, то функция чтения завершается с ошибкой.

Далее выполняется проверка информативного содержания файла. Если функция была вызвана с параметром файла инициализации, то пока не закончится файл, выполняется его построчное чтение. Затем выполняется парсинг прочитанной строки в структуру инициализации. Если во время парсинга произошла ошибка, то происходит закрытие файлового потока, и функция завершается с ошибкой.

Если функция была вызвана с параметром файла соответствия адресов, то пока не закончится файл, выполняется его построчное чтение. Затем выполняется парсинг прочитанной строки в структуру соответствия адресов. Если во время парсинга произошла ошибка, то происходит закрытие файлового потока, и функция завершается с ошибкой.

После успешного чтения файла, файловый поток закрывается, и функция завершается успешно.

Алгоритм работы функции чтения файла, требуемого для работы программы, в виде блок-схемы представлен на рисунке 18.

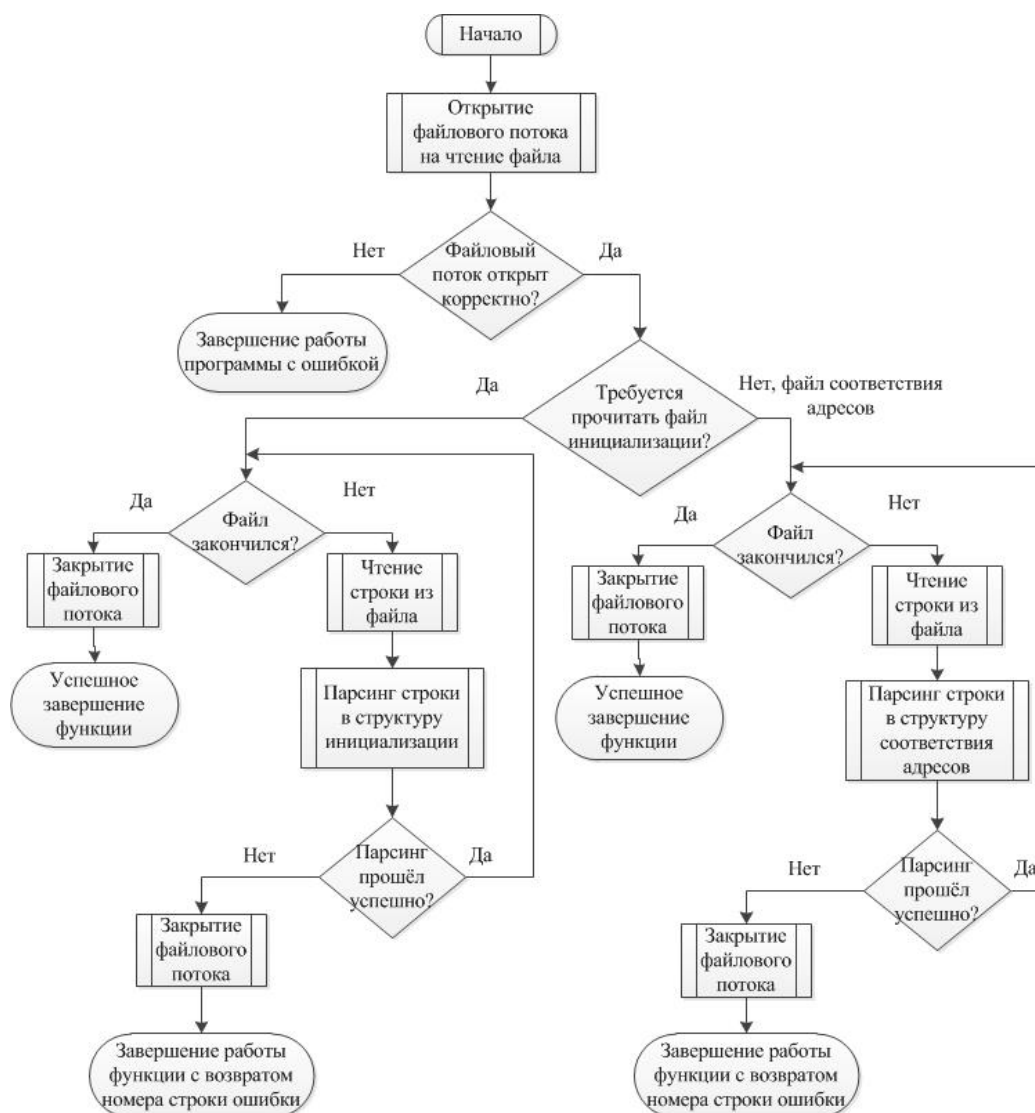


Рисунок 18 – Алгоритм чтения файлов, требуемых для работы программы «Настройка сети»

3.2.3 Разработка алгоритма работы функций графического интерфейса

Был разработан алгоритм главного окна графического ПО «Настройка сети». В начале работы главного окна выполняется вызов окна авторизации. После успешной работы окна авторизации включается видимость главного окна. Выполняется настройка доступности логических блоков главного окна в соответствии с кодом доступа авторизованного пользователя. Осуществляется настройка параметров таблиц и параметра обхода элементов

по часовой стрелке. Выполняется заполнение всех выпадающих списков и таблиц. Создаются акселераторы на все кнопки окна.

Затем сигналы о нажатии на элементы окна («Сменить пользователя», «Сменить пароль», «Добавить пользователя», «Изменить права», «Удалить пользователя», «Подключиться», «Применить», «Сбросить», «Проверить») связываются со слотами, которые их обрабатывают.

Алгоритм работы главного окна, требуемого для работы программы, в виде блок-схемы представлен на рисунке 19 и на плакате «Алгоритмы работы функций многопользовательского режима в разработанном графическом ПО «Настройка сети».

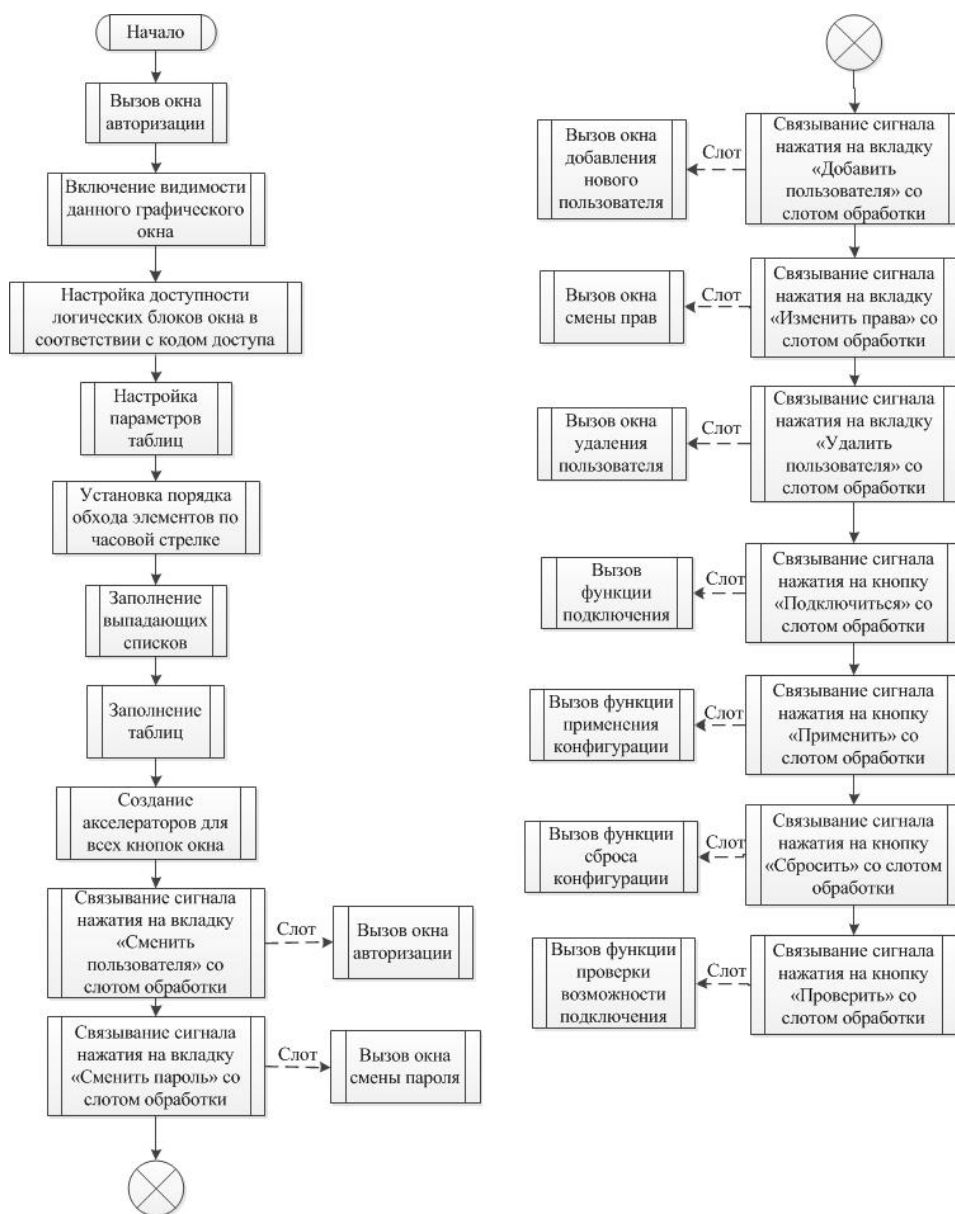


Рисунок 19 – Алгоритм работы главного окна

Был разработан алгоритм работы окна авторизации. В начале работы окна включается его видимость. Затем, в целях обеспечения безопасности визуальной информации, выключается видимость вводимого текста в поле пароля. Выполняются соединения сигналов нажатия кнопок «ОК» и «Отмена» со слотами, которые их обрабатывают.

В случае нажатия на кнопку «ОК» на сервер отправляется запрос на генерацию соли. После чтения соли от сервера вычисляются хэш-сумма пароля и длина имени пользователя. Полученная информация записывается в заголовок. На сервер отправляется запрос на авторизацию и заголовок. Затем на сервер отправляется имя пользователя той длины, которая была указана ранее в заголовке. Если сервер вернул успешный код выполнения операции, то выполняется чтение кода доступа данного пользователя, окно авторизации закрывается. Если сервер вернул код ошибки работы сервера, то пользователю выводится сообщение об ошибке работы сервера, и выполняется очистка полей ввода данного окна.

Если сервер вернул код ошибки авторизации по паролю, то пользователю выводится сообщение о неверно введённом логине или пароле, и выполняется очистка полей ввода данного окна. Если сервер вернул код ошибки авторизации по IP, то пользователю выводится сообщение с просьбой выполнить авторизацию на своём рабочем компьютере, и выполняется очистка полей ввода данного окна.

В случае нажатия на кнопку «Отмена», программа «Настройка сети» завершает свою работу. Это необходимо для того, чтобы пользователь не получил доступа к программе после неуспешной авторизации. Сигнал о завершении работы клиента подаётся на сервер. Это необходимо для того, чтобы сервер мог завершить сеанс пользователя и не продолжал ожидать запросов.

Алгоритм работы окна авторизации в виде блок-схемы представлен на рисунке 20 и на плакате «Алгоритм работы аутентификации на основе пароля и IP-адреса узла сети разработанного ПО».

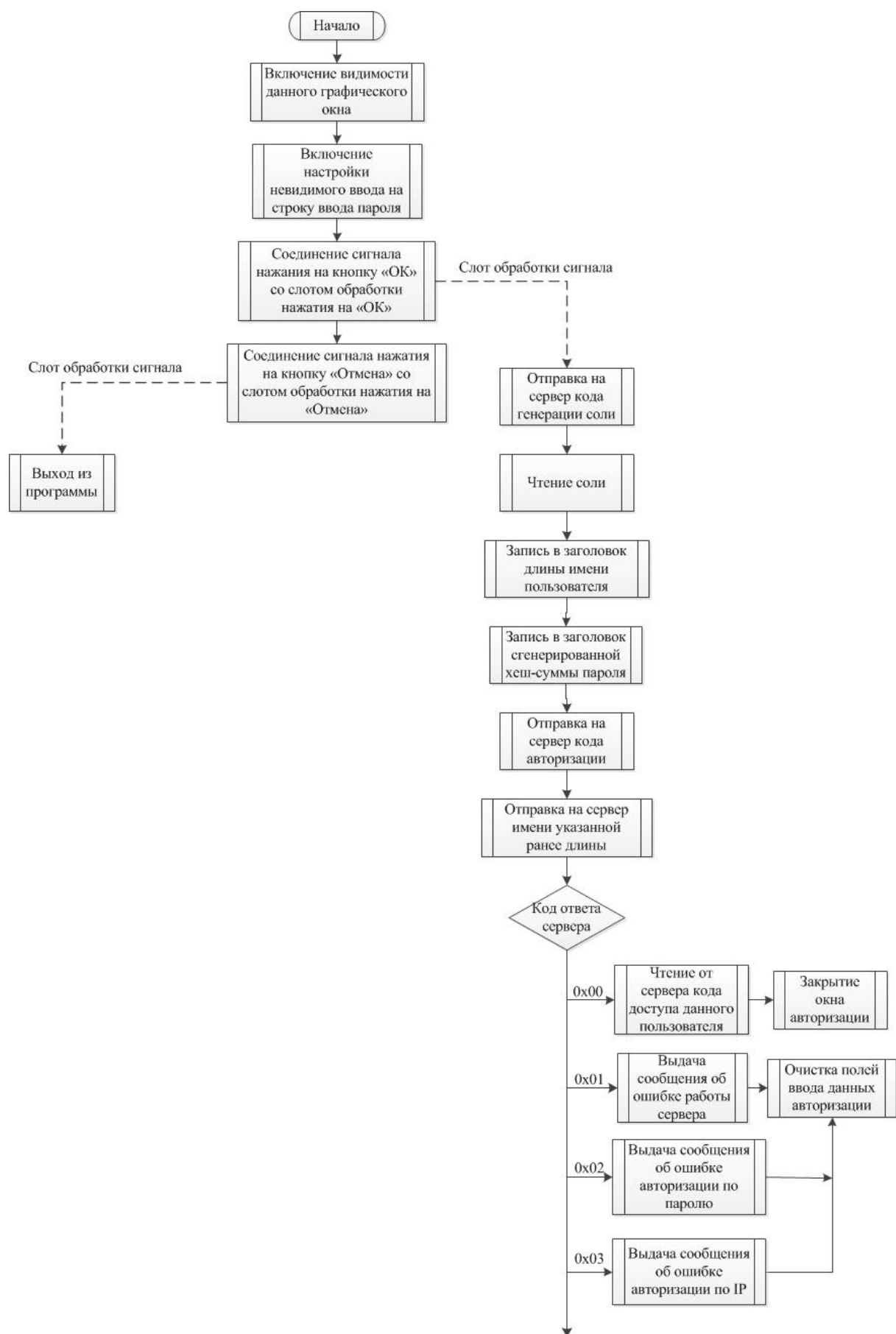


Рисунок 20 – Алгоритм работы окна авторизации

Был разработан алгоритм работы окна смены пароля.

Для успешной смены пароля необходимо подтвердить свою личность, введя правильный старый пароль.

В начале работы окна включается его видимость. Текстовый блок заполняется именем пользователя, который авторизован в текущем сеансе. Далее выполняется соединение сигнала о нажатии на клавишу «ОК» со слотом, который его обрабатывает.

Если был вызван слот обработки сигнала о нажатии клавиши «ОК», то выполняется проверка о правильности введения нового пароля. Если новый пароль не был одинаково написан дважды, то пользователю выводится сообщение об ошибке с указанием на несовпадение паролей. Все поля для ввода в текущем окне очищаются.

Иначе на сервер отправляется запрос на генерацию соли. После получения соли от сервера, выполняется заполнение заголовка длиной имени пользователя и сгенерированными хэш-суммами нового и старого паролей.

На сервер отправляется запрос на смену пароля пользователя и заголовок. Затем выполняется отправка имени пользователя длины, указанной ранее в заголовке.

Если сервер ответит кодом успешного выполнения смены пароля, то окно смены пароля закрывается.

Если сервер ответит кодом ошибки работы сервера, то пользователь получит сообщение об ошибке, связанной с некорректной работы сервера авторизации. Все поля окна будут очищены.

Если сервер ответит кодом ошибки смены пароля, то пользователь получит сообщение об ошибке смены пароля, связанной с неверно введенным старым паролем. Все поля окна будут очищены.

Очистка полей окна производится для возможности заполнения их новыми значениями при следующем использовании.

Алгоритм работы окна смены пароля в виде блок-схемы представлен на рисунке 21.

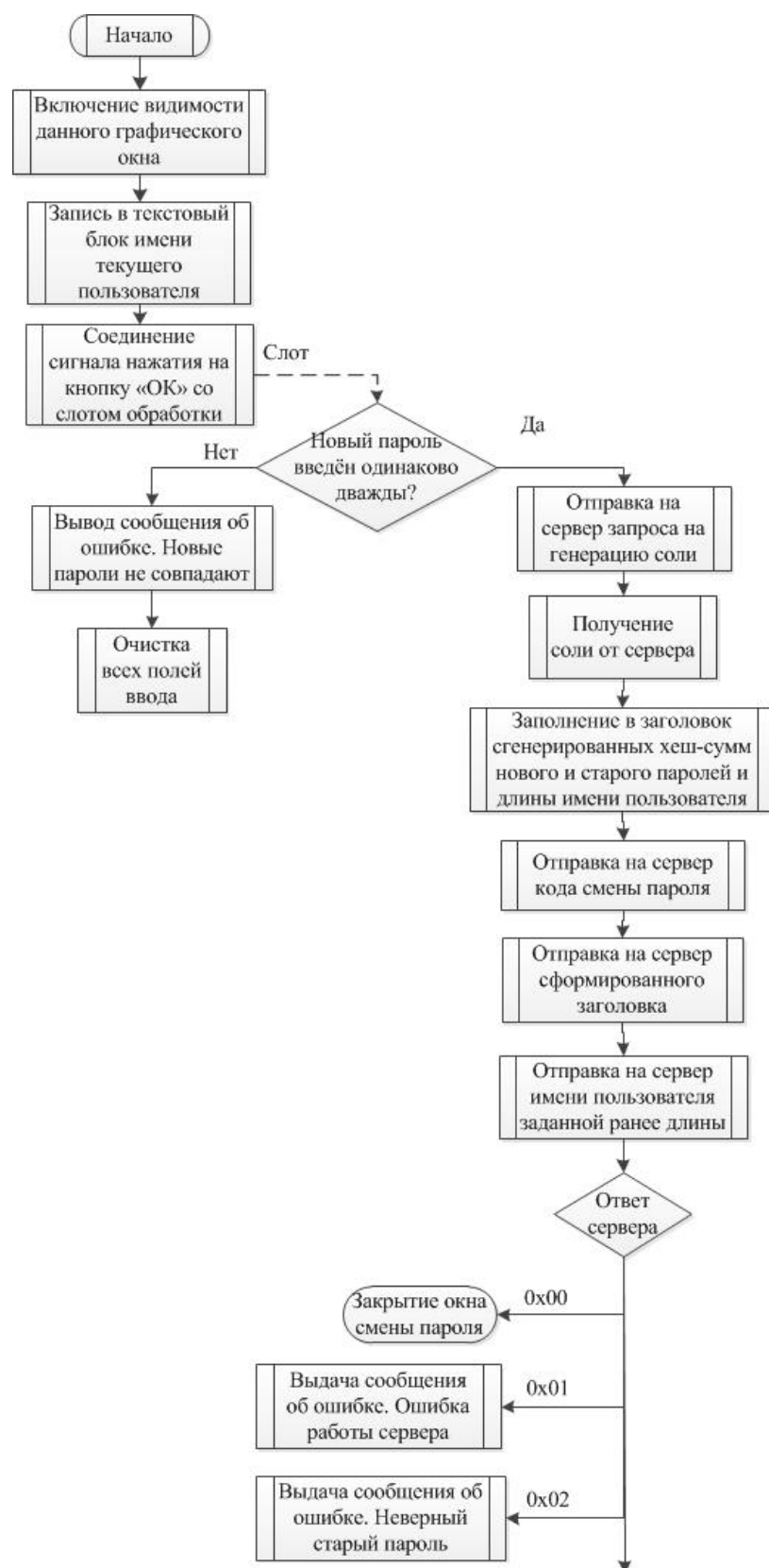


Рисунок 21 – Алгоритм работы окна смены пароля

Был разработан алгоритм работы окна добавления нового пользователя. В начале работы окна включается его видимость. Далее

выполняется соединение сигнала о нажатии на клавишу «ОК» со слотом, который его обрабатывает.

Если кнопка «ОК» была нажата, выполняется проверка на пустые поля. Если в окне остались незаполненные поля, то пользователь получает сообщение об ошибке с просьбой заполнить все поля формы. Все поля ввода очищаются.

Далее выполняется проверка корректного ввода нового пароля. Если введенные два новых пароля не совпадают, пользователь получает сообщение об ошибке с указанием, что новые пароли не совпадают. Все поля ввода очищаются.

Если все проверки пройдены успешно, то выполняется запрос серверу на генерацию соли. После получения соли от сервера, заполняется заголовок. В заголовок входят вычисленные на основе соли хэш-суммы старого и нового паролей, длина имени пользователя и IP-адрес, с которого пользователь может осуществлять доступ к разрабатываемой системе.

Затем на сервер выполняется запрос на добавление нового пользователя. Осуществляется отправка сформированного заголовка и имени пользователя, указанной в заголовке длины. Если сервер вернул код успешного выполнения операции, осуществляется вывод информационного сообщения об успешном добавлении нового пользователя. Все поля для ввода очищаются.

Если сервер вернул код ошибки работы сервера, пользователь получает сообщение об ошибке, связанной с некорректной работой сервера авторизации. Все поля для ввода очищаются.

Если сервер вернул код ошибки добавления нового пользователя, осуществляется вывод сообщения об ошибке, связанной с тем, что пользователь с таким именем уже зарегистрирован в системе. Все поля для ввода очищаются.

Алгоритм работы окна добавления пользователя в виде блок-схемы представлен на рисунке 22.

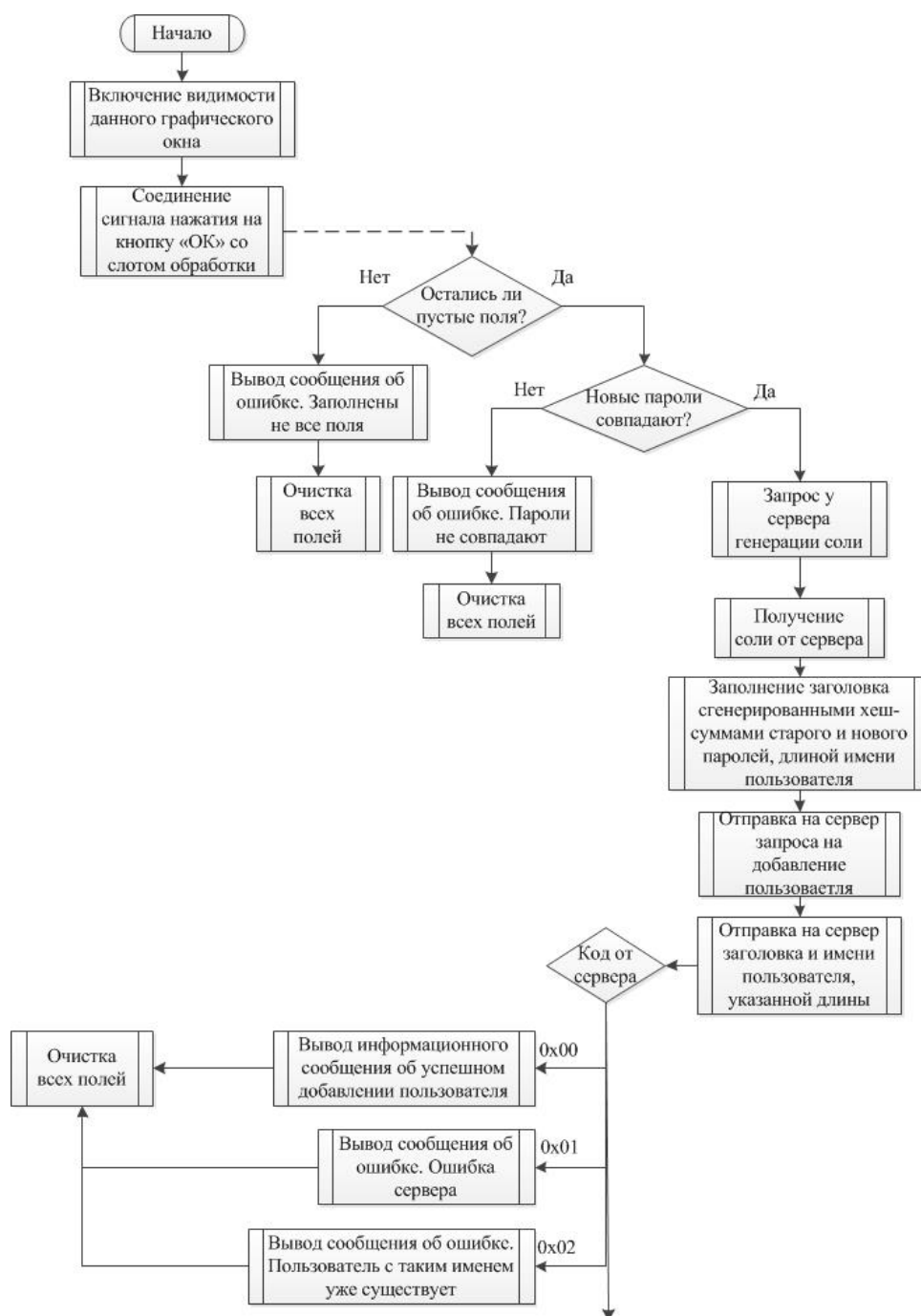


Рисунок 22 – Алгоритм работы окна добавления нового пользователя

Был разработан алгоритм работы окна изменения прав пользователя. В начале работы окна включается его видимость. Затем выполняется заполнение выпадающего списка с типами доступа, предусмотренными в системе. Далее соединяется сигнал о нажатии кнопки «ОК» со слотом, который его обрабатывает.

Если была нажата кнопка «ОК», на сервер отправляется запрос на генерацию соли. После получения соли от сервера, выполняется заполнение

заголовка, в который входят длины имени пользователя и код доступа, выбранный из выпадающего списка.

На сервер выполняется запрос на смену прав пользователя. Отправляется имя пользователя длины, указанной ранее в заголовке.

Алгоритм работы окна изменения прав пользователя в виде блок-схемы представлен на рисунке 23 и на плакате «Алгоритмы работы функций многопользовательского режима в разработанном графическом ПО «Настройка сети».

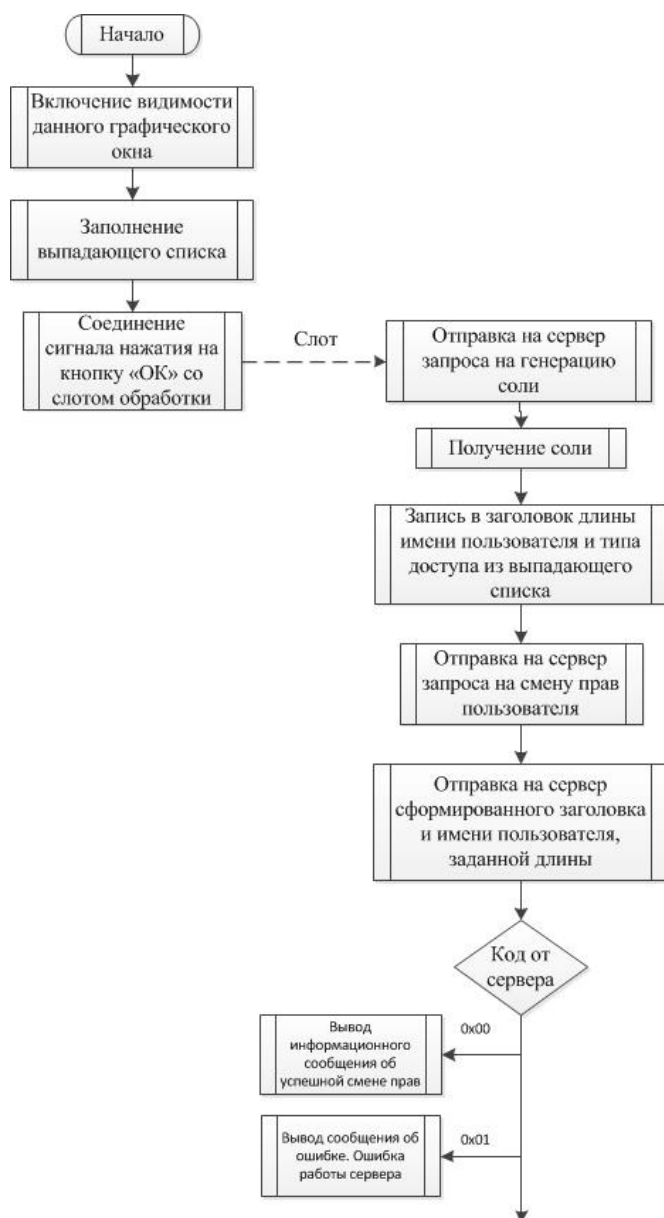


Рисунок 23 – Алгоритм работы окна изменения прав пользователя

Был разработан алгоритм работы окна удаления пользователя. Операция удаления пользователя является критической операцией, поэтому её реализация требует повышенного уровня безопасности. Было принято решение, что для выполнения операции удаления пользователя, администратор должен подтвердить свою личность повторным вводом пароля.

В начале работы окна включается его видимость. Затем серверу отправляется запрос на получение полного списка имён пользователей, зарегистрированных в системе. После получения этого списка, на основании его данных выполняется заполнение выпадающего списка с именами всех пользователей, зарегистрированных в системе. Далее соединяется сигнал о нажатии кнопки «ОК» со слотом, который его обрабатывает.

Если была нажата кнопка «ОК», на сервер отправляется запрос на генерацию соли. После получения соли от сервера, выполняется заполнение заголовка, в который входят длина имени администратора и сгенерированная с помощью соли хэш-сумма его пароля. На сервер выполняется отправка запроса на авторизацию и сформированного заголовка. Затем на сервер отправляется имя администратора длины, указанной ранее в заголовке. Если авторизация администратора прошла неуспешно, пользователю выводится сообщение об ошибке, связанной с неверно введённым паролем администратора. Все поля ввода очищаются.

После успешной авторизации в новый заголовок записывается длина имени пользователя, которого требуется удалить. Затем на сервер отправляется запрос на удаление и сформированный заголовок. На сервер отправляется имя пользователя длины, указанной ранее в заголовке.

Если сервер вернул код успешного удаления, то пользователь получает информационное сообщение об успешном удалении пользователя, все поля для ввода очищаются. Иначе, пользователь получает сообщение об ошибке, связанной с некорректной работой сервера авторизации. Все поля очищаются. Очистка полей окна производится для возможности заполнения

их новыми значениями при следующем использовании. Алгоритм работы окна удаления пользователя в виде блок-схемы представлен на рисунке 24 и на плакате «Алгоритмы работы функций многопользовательского режима в разработанном графическом ПО «Настройка сети».

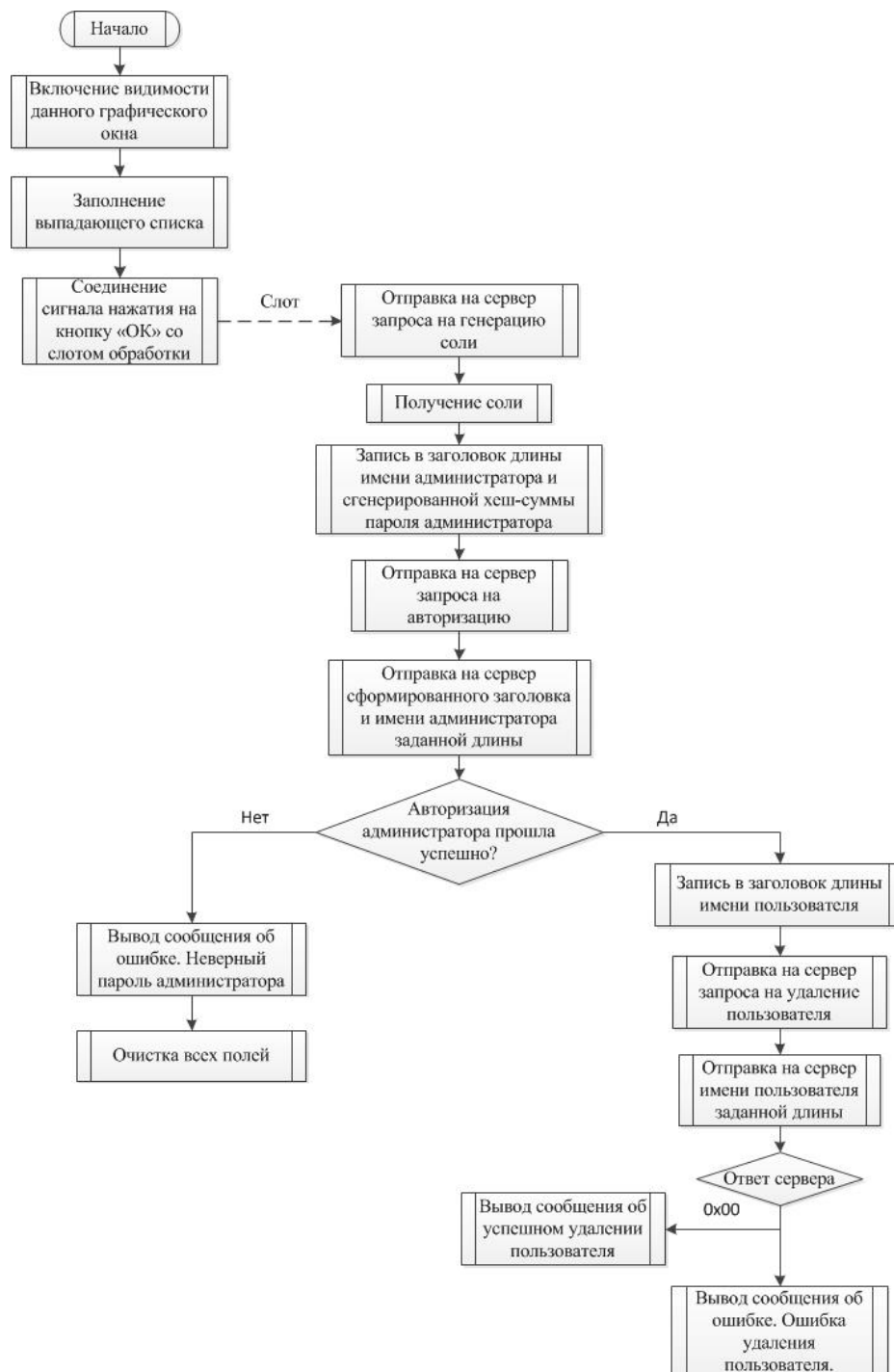


Рисунок 24 – Алгоритм работы окна удаления пользователя

Был разработан алгоритм применения конфигурации узла. Данная функция использует таблицу конфигурации и выпадающий список всех

возможных конфигураций. Данную информацию эти элементы графики получают из файла конфигурации. Пользователь выбирает в выпадающем списке название конфигурации. В соответствии с этим названием в таблице конфигураций меняется информация.

Если пользователь хочет выбрать именно ту конфигурацию, которая представлена в таблице, он нажимает на кнопку «Применить». В начале выполнения алгоритма обнуляется счётчик ошибок. Пока не закончится таблица конфигурации, читается строка таблицы. Если в строке есть нулевые элементы, выводится сообщение об ошибке, сигнализирующей о том, что заполнены не все поля строки. Счётчик ошибок увеличивается на один. Иначе выполняется парсинг элемента строки из колонки «Адрес» в структуру, состоящую из IP-адреса и маски. Указанный в данной строке интерфейс выключается, и выполняется смена адреса интерфейса, указанного в строке, на адрес, указанный в этой же строке. Если во время смены адреса произошла ошибка, выводится сообщение об ошибке смены адреса. Счётчик ошибок увеличивается на один. Иначе выполняется смена маски интерфейса, указанного в строке, на маску, указанную в этой же строке.

Если во время смены маски произошла ошибка, выводится сообщение об ошибке смены маски. Счётчик ошибок увеличивается на один. Иначе интерфейс включается обратно. Осуществляется переход к следующей строке таблицы, если текущая строка не является последней. Если обработана последняя строка таблицы, выполняется проверка счётчика ошибок. Таблица текущего состояния обновляется. Если счётчик ошибок нулевой, в выпадающий список и таблицу блока проверки подключения возвращается ранее применённая конфигурация. Текущая конфигурация из этого логического блока удаляется. Операция удаления конфигурации из блока пинга необходима для предотвращения возможности проверки подключения узла к самому себе.

Алгоритм применения конфигурации в виде блок-схемы представлен на рисунке 25.

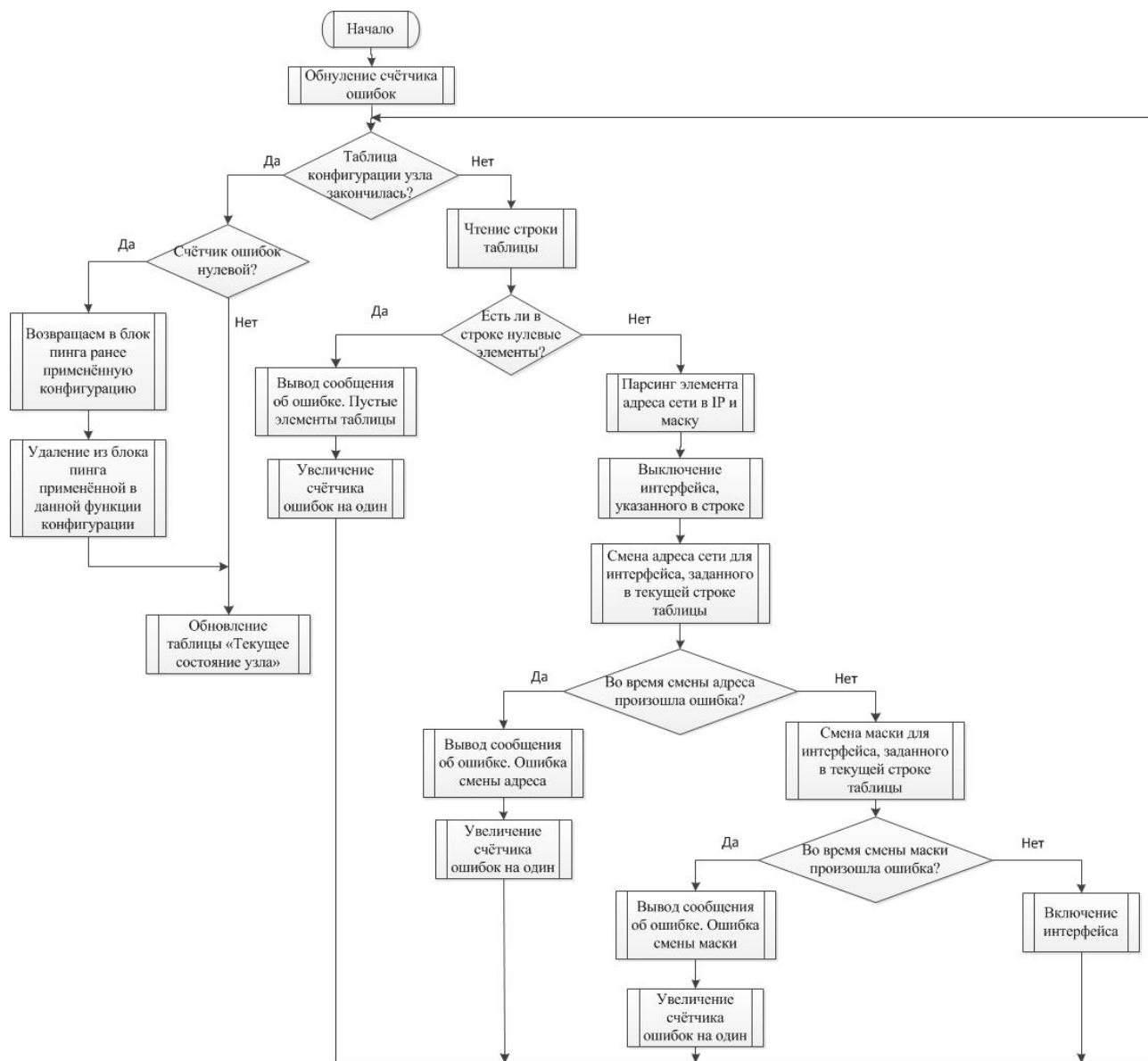


Рисунок 25 – Алгоритм применения конфигурации узла

В начале выполнения алгоритма обнуляется счётчик правильно сброшенных адресов. Пока не закончится таблица текущей конфигурации узла, выполняется чтение строки. Интерфейс, указанный в данной строке, выключается. Если выключение интерфейса прошло с ошибкой, выводится сообщение об ошибке выключения интерфейса. Иначе IP-адрес этого интерфейса меняется на «0.0.0.0». Выполняется переход на следующую строку, если текущая строка не является последней. Иначе выполняется проверка счётчика правильно сброшенных адресов. В завершении работы алгоритма обновляется таблица текущей конфигурации.

Алгоритм сброса конфигурации в виде блок-схемы представлен на рисунке 26.

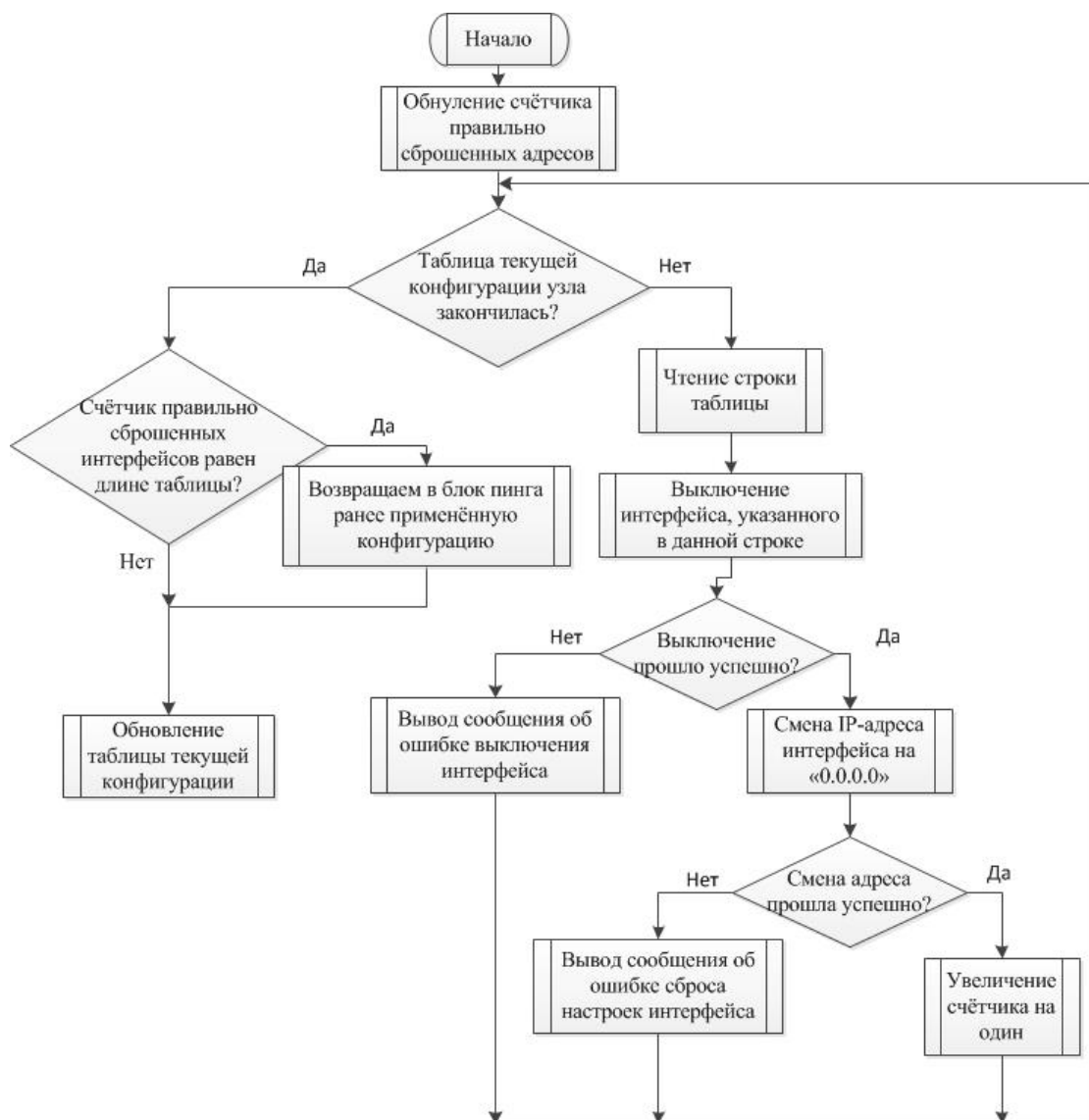


Рисунок 26 – Алгоритм сброса конфигурации узла сети

Был разработан алгоритм проверки возможности подключения к выбранным узлам сети. В начале работы алгоритма для пользователя выключается доступ к изменению состояния выпадающего списка и кнопки «Проверить» в логическом блоке проверки возможности подключения к выбранным узлам сети. Задаются цвета для всех возможных результатов работы функции («Доступен», «Не доступен», «Ошибка», «Проверка»). В колонке «Состояние» устанавливается значение «Проверка».

Пока не закончится таблица узлов, выполняется чтение строки. Значение хоста из этой строки заносится в предварительно созданную хост-структуру, которая содержит имя хоста и статус его доступности. Формируется проверочный пакет для рассылки по списку хостов. Создаётся сокет для реализации взаимодействия со списком хостов из таблицы.

Пока в хост-структуре не закончатся элементы, из каждого очередного элемента получают IP-адрес по номеру хоста. Формируется запрос для отправки на полученный IP-адрес. Сформированный запрос отправляется адресату. Когда алгоритм достиг конца структуры, осуществляется переход в бесконечный цикл.

Выполняется функция ожидания ответов на сокете проверки подключения. Если во время работы этой функции произошла ошибка, в хост-структуру заносится значение «Ошибка», и выполняется выход из цикла.

Если за время работы функции не было получено ни одного ответа, выполняется выход из цикла.

Если на сокете был обнаружен ответ, выполняется получение адресной структуры ответившего узла, и читается его ответ. Если длина ответа узла меньше 76, цикл переходит на новую итерацию. Иначе осуществляется чтение ICMP-заголовка. Если в этом заголовке указано, что хост доступен, то в хост-структуру в соответствии с адресом узла заносится значение «Доступен».

После выхода из цикла проверочный сокет закрывается. Выполняется заполнение таблицы проверки возможности подключения в соответствии с хост-структурой.

В конце работы алгоритма выполняется включение возможности пользователем изменять состояние элементов логического блока проверки подключения.

Алгоритм проверки возможности подключения к выбранным узлам сети в виде блок-схемы представлен на рисунке 27.

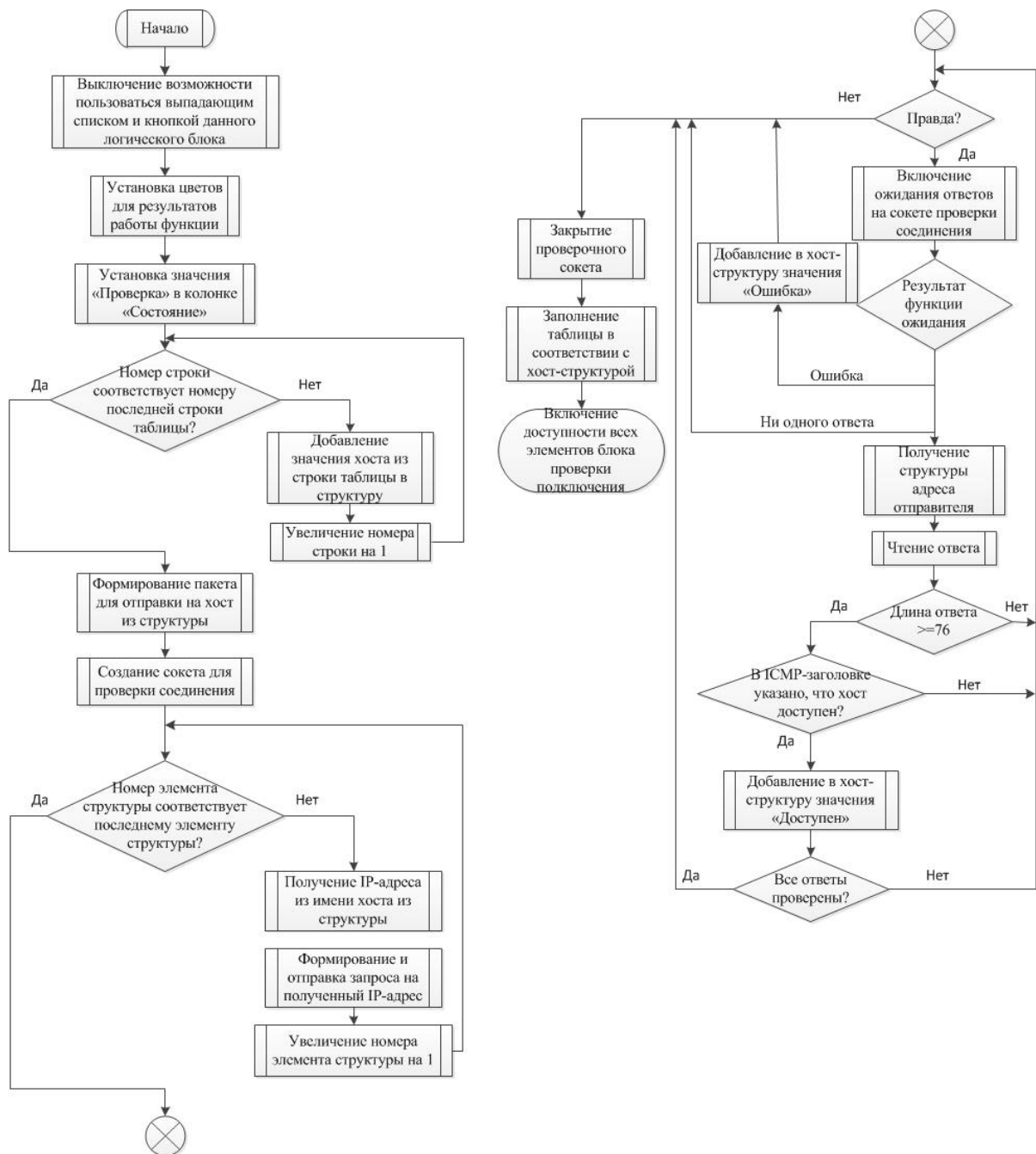


Рисунок 27 – Алгоритм проверки возможности подключения к узлам

3.3 Разработка пользовательского графического интерфейса для программного обеспечения «Настройка сети»

3.3.1 Разработка шаблонов пользовательского графического интерфейса для программного обеспечения «Настройка сети»

С помощью ПО «QT 4 Designer» были разработаны шаблоны пользовательского графического интерфейса для программы «Настройка сети». В результате были сформированы файлы с расширением «.ui».

Был разработан шаблон диалогового окна авторизации. Данное окно включает в себя два параметра: «Имя пользователя» и «Пароль». Для обеспечения конфиденциальности визуальной информации, вводимый пользователем пароль скрыт для просмотра. Внизу окна содержится стандартный набор кнопок «Отмена» и «ОК».

Шаблон диалогового окна авторизации представлен на рисунке 28.

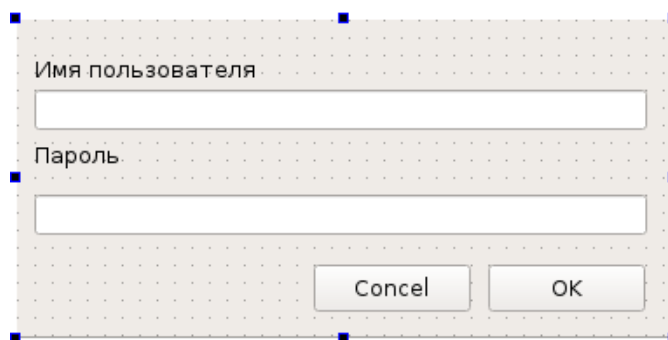


Рисунок 29 – Шаблон диалогового окна авторизации

Был разработан шаблон главного окна. Главное окно состоит из четырёх логических блоков: «Подключиться к ...», «Профиль узла», «Текущее состояние», «Проверить взаимодействие с ...».

В разработанную можель были внесены данне таблиц и выпадающих списков для наглядности будущей раюоты программы. Внесенные данные будут сохранены в пользовательском инерфейсе, а программный код в начале своего выполнения сотрёт все исходые данные из интерфейса.

Разработанная модель окна состоит из взаимонезависимых логических блоков, каждый из которых имеет возможность быть заблокированным для использования. Это выражается в отсутствии возможности нажать на кнопку, просмотреть выпадающий список или содержимое таблицы. Такое свойство макета позволяет реализовать разграничение в использовании программы различными категориями пользователей.

В логическом блоке «Подключиться к ...» использован выпадающий список, который заполняется названиями узлов, доступных для выполнения подключения. В данный блок входит кнопка «Подключиться», которая срабатывает по нажатию левой кнопки мыши или по нажатию клавиши Enter.

В логическом блоке «Профиль узла» использован выпадающий список, элементы которого заполняются названиями существующих конфигураций данного узла. Так же в данный блок входит таблица, состоящая из параметров «Интерфейс» и «Адрес». Таблица расширяется динамически в соответствии с объёмом информации. В «Профиль узла» находится кнопка «Подключиться», которая срабатывает по нажатию левой кнопки мыши или по нажатию клавиши Enter.

В логическом блоке «Текущее состояние» находится таблица, состоящая из параметров «Интерфейс» и «Адрес». Таблица расширяется динамически в соответствии с объёмом информации. В данный блок входит кнопка «Сбросить», которая срабатывает по нажатию левой кнопки мыши или по нажатию клавиши Enter.

В логическом блоке «Проверить взаимодействие с...» использован выпадающий список, элементы которого заполняются названиями существующих конфигураций узла. В данный блок входит таблица, состоящая из параметров «Узел», «Интерфейс», «Адрес» и «Состояние». В «Проверить взаимодействие с ...» находится кнопка «Проверить», которая срабатывает по нажатию левой кнопки мыши или по нажатию клавиши Enter.

Результат создания шаблона главного окна представлен на рисунке 30.

Мой аккаунт
Администрирование
Пишите здесь

Подключиться к
Подключиться

Профиль узла
СКПИ1_1
Применить

Интерфейс	Адрес
1 Сеть1	192.168.3.41
2 Сеть2	192.168.2.12

Текущее состояние
Сбросить

Интерфейс	Адрес
1 Сеть1	192.168.3.6
2 Сеть2	192.168.4.88
3 Сеть3	не установ...
4 Сеть4	не установ...

Проверить взаимодействие с
СОИ1
Проверить

Узел	Интерфейс	Адрес	Состояние
1 СОИ1	Сеть1	192.168.2.44	доступен

Рисунок 30 – Шаблон главного окна

В шаблоне главного окна использовано контекстное меню в верхней части экрана. Данное меню состоит из категорий «Мой аккаунт» и «Администрирование». При наведении мыши на одну из категорий, она раскрывается выпадающим списком.

Категория «Мой аккаунт» состоит из элементов «Сменить пользователя» и «Сменить пароль». Категория «Администрирование» состоит из элементов «Добавить пользователя», «Изменить права доступа» и «Удалить пользователя». Выпадающее меню этих двух категорий представлено на рисунках 31 и 32.

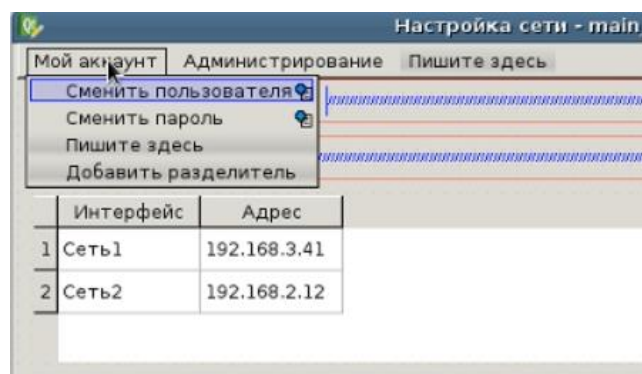


Рисунок 31 – Шаблон первой вкладки главного меню приложения

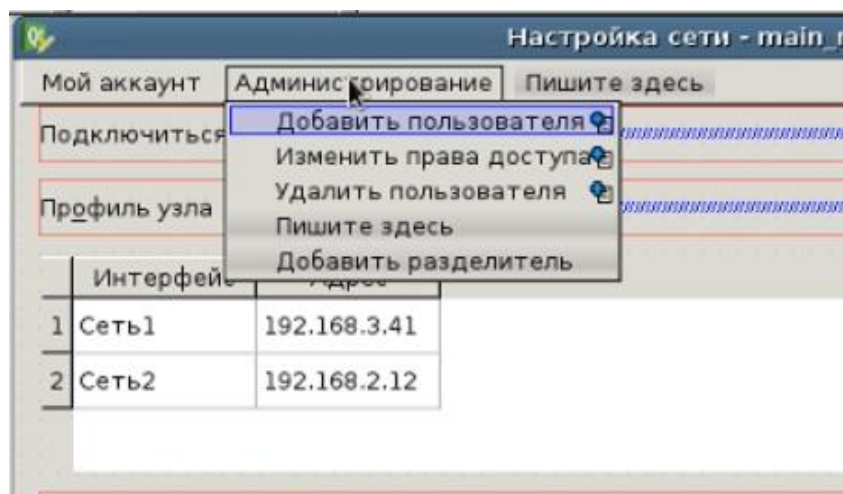


Рисунок 32 – Шаблон второй вкладки главного меню приложения

Был разработан шаблон диалогового окна смены пароля. Данное окно состоит из четырёх горизонтально отформатированных строк. Первая строка «Пользователь» содержит динамически изменяемую текстовую вставку, в которой будет содержаться имя текущего пользователя. Строки «Старый пароль», «Новый пароль» и «Новый пароль ещё раз» содержат поля для ввода. В целях обеспечения безопасности визуальной информации, при вводе пароля содержимое этих полей скрыто.

Внизу окна содержится стандартный набор кнопок «Отмена» и «ОК».

Разработанный шаблон диалогового окна смены пароля представлен на рисунке 33.

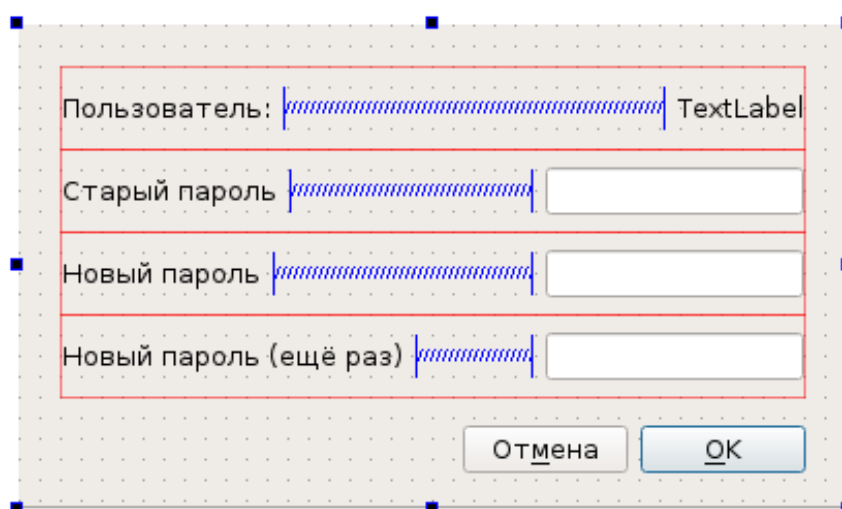


Рисунок 33 – Шаблон диалогового окна смены пароля

Был разработан шаблон диалогового окна добавления нового пользователя. Данное окно состоит из пяти горизонтально отформатированных строк. В первой строке находится поле для ввода имени пользователя.

Вторая и третья строки содержат поля для ввода пароля пользователя. В целях обеспечения безопасности визуальной информации, текст, вводимый в этих полях, скрыт.

Строка «Тип доступа» содержит выпадающий список, элементы которого включают названия ролей безопасности, разработанных для данного программного обеспечения.

Внизу окна содержится стандартный набор кнопок «Отмена» и «ОК».

Разработанный шаблон диалогового окна добавления нового пользователя представлен на рисунке 34.

The image shows a template for a dialog box to add a new user. It consists of five horizontal rows, each with a label and an input field. The labels are: 'Имя пользователя' (User name), 'Пароль' (Password), 'Пароль (ещё раз)' (Password (again)), 'Тип доступа' (Access type), and 'IP пользователя' (User IP). The input fields are: a text box for the name, two password boxes with masked characters, a dropdown menu for the access type (currently showing 'Пользователь'), and a text box for the IP address. At the bottom right of the dialog are two buttons: 'Отмена' (Cancel) and 'ОК' (OK).

Рисунок 34 – Шаблон диалогового окна добавления нового пользователя

Был разработан шаблон диалогового окна смены прав пользователя. Данное окно состоит из двух горизонтально отформатированных строк. Первая строка «Имя пользователя» включает в себя выпадающий список, элементы которого состоят из имён всех пользователей, зарегистрированных в системе.

Во второй строке «Тип доступа» так же содержится выпадающий список, элементы которого состоят из названий ролей безопасности, разработанных для данной системы.

Внизу окна содержится стандартный набор кнопок «Отмена» и «ОК».

Разработанный шаблон диалогового окна смены прав пользователя представлен на рисунке 35.

The diagram shows a rectangular dialog box template with a light gray background and a dotted grid. It contains two rows of controls, each enclosed in a red rectangular border. The first row has a text label 'Имя пользователя' followed by a text input field with blue wavy lines and a dropdown arrow. The second row has a text label 'Тип доступа' followed by a dropdown menu showing 'Пользователь' and a dropdown arrow. At the bottom right, there are two buttons: 'Отмена' and 'ОК'.

Рисунок 35 – Шаблон диалогового окна смены прав пользователя

Был разработан шаблон диалогового окна удаления пользователя. Это окно состоит из двух отформатированных по горизонтали строк.

В первой строке «Пользователь» содержится выпадающий список, состоящий из имён всех пользователей, зарегистрированных в системе.

Во второй строке «Пароль администратора» содержится поле для ввода пароля. В целях обеспечения безопасности визуальной информации, текст, вводимый в это поле, скрыт.

Внизу окна содержится стандартный набор кнопок «Отмена» и «ОК».

Разработанный шаблон диалогового окна удаления пользователя представлен на рисунке 36.

The diagram shows a rectangular dialog box template with a light gray background and a dotted grid. It contains two rows of controls, each enclosed in a red rectangular border. The first row has a text label 'Пользователь' followed by a dropdown menu with a dropdown arrow. The second row has a text label 'Пароль администратора' followed by a text input field with blue wavy lines. At the bottom right, there are two buttons: 'Отмена' and 'ОК'.

Рисунок 36 – Шаблон диалогового окна удаления пользователя

3.3.2 Разработка программной реализации для шаблонов пользовательского графического интерфейса

На языке C++ с использованием пакета Qt4 была написана программа, реализующая работу разработанных в пункте 4.3.2 шаблонов пользовательского графического интерфейса. Текст программы представлен в Приложениях Г - 3.

После компиляции проекта был осуществлён запуск программы «Настройка сети». Для того, чтобы все функции программы работали корректно, её необходимо запускать в режиме суперпользователя в операционной системе Linux. Разработанное программное обеспечение предназначено для работы только в ОС Linux.

При запуске программы открывается окно авторизации, представленное на рисунке 37.

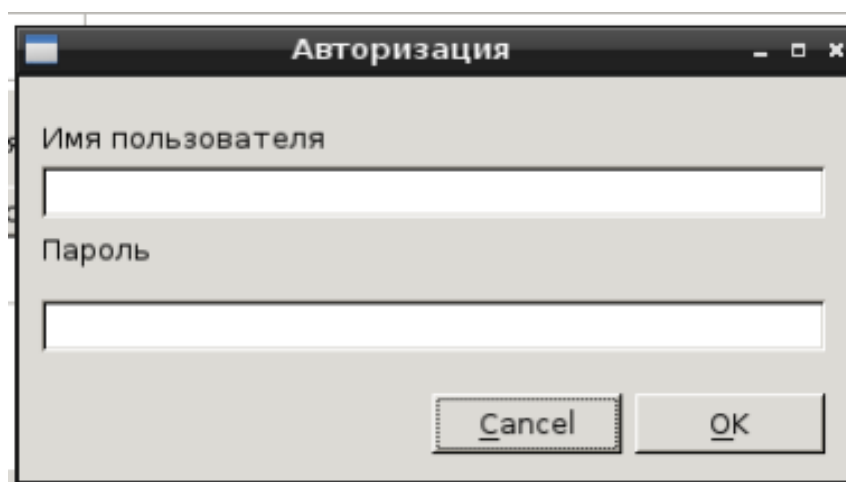


Рисунок 37 – Окно авторизации

При нажатии на кнопку «Cancel» или кнопку закрытия окна, закрывается вся программа «Настройка сети». В случае неуспешной авторизации пользователю выдаётся сообщение об ошибке авторизации. Содержимое всех текстовых полей стирается, пользователю предоставляется новая попытка авторизации. Варианты сообщений об ошибке авторизации представлены на рисунках 38 и 39.

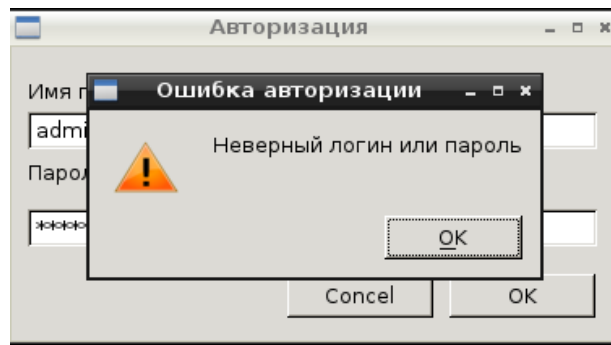


Рисунок 38 – Сообщение об ошибке авторизации по паролю

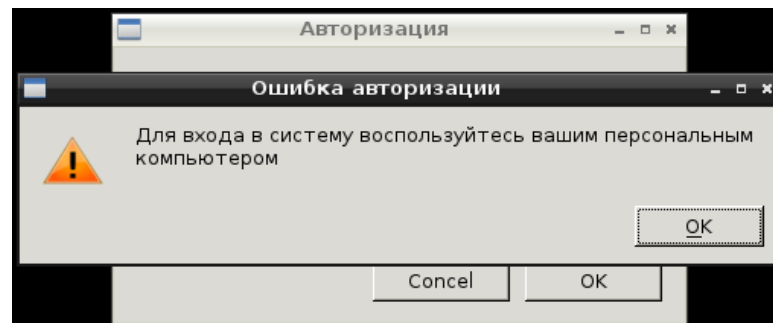


Рисунок 39 – Сообщение об ошибке авторизации по IP-адресу

После успешной авторизации открывается главное окно программы. Вид главного окна отличается в зависимости от типа доступа авторизованного пользователя. Для системного администратора, главное окно будет иметь вид, представленный на рисунке 40.

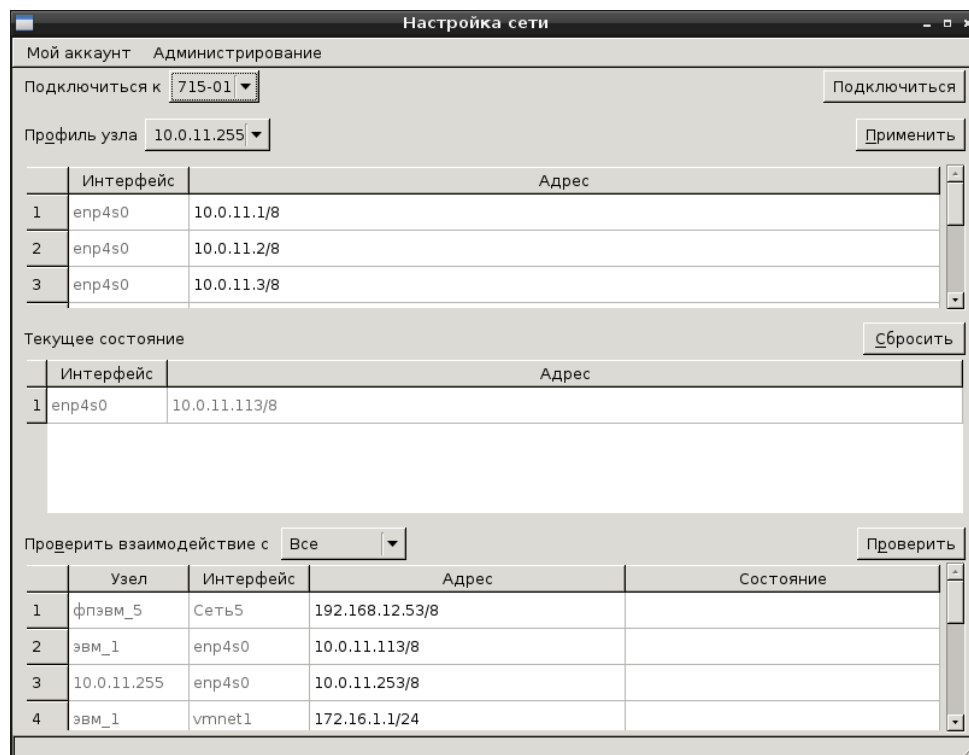


Рисунок 40 – Главное окно для системного администратора

В случае входа в систему пользователя с типом доступа «Локальный администратор», поле «Подключиться к ...» будет заблокировано для всех видов действий. Главное окно программы будет иметь вид, представленный на рисунке 41.

	Интерфейс	Адрес
1	enp4s0	10.0.11.1/8
2	enp4s0	10.0.11.2/8
3	enp4s0	10.0.11.3/8

	Интерфейс	Адрес
1	enp4s0	10.0.11.113/8

	Узел	Интерфейс	Адрес	Состояние
1	фпэвм_5	Сеть5	192.168.12.53/8	
2	эвм_1	enp4s0	10.0.11.113/8	
3	10.0.11.255	enp4s0	10.0.11.253/8	
4	эвм_1	vmnet1	172.16.1.1/24	

Рисунок 41 – Главное окно для локального администратора

В случае входа в систему пользователя с типом доступа «Пользователь», ему будут доступны только поля «Проверить взаимодействие с». Все остальные поля для него будут заблокированы. Главное окно для локального пользователя представлено на рисунке 42.

Главное окно содержит контекстное меню. Содержание этого меню представлено на рисунках 43 и 44.

При нажатии на вкладку «Сменить пользователя» открывается окно авторизации.

При нажатии на вкладку «Сменить пароль» открывается окно смены пароля, представленное на рисунке 45.

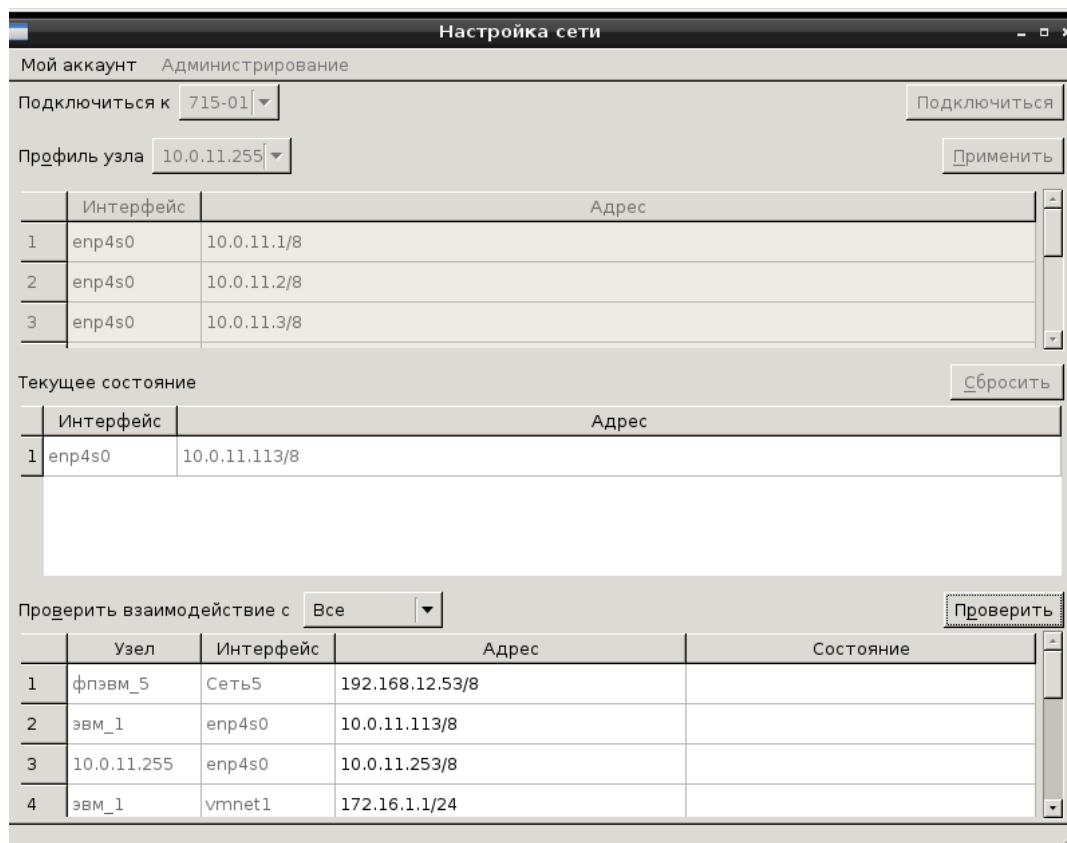


Рисунок 42 – Главное окно для локального пользователя

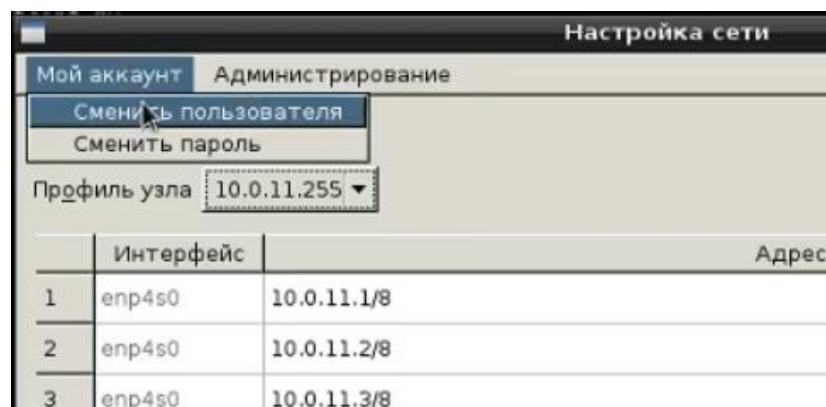


Рисунок 43 – Первая вкладка контекстного меню главного окна

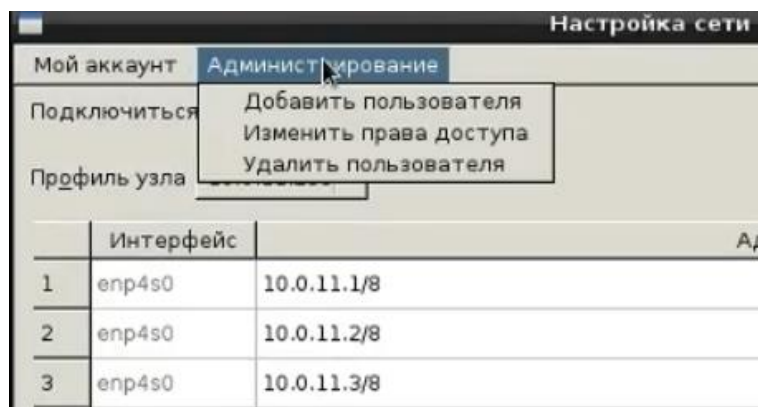


Рисунок 44 – Вторая вкладка контекстного меню главного окна

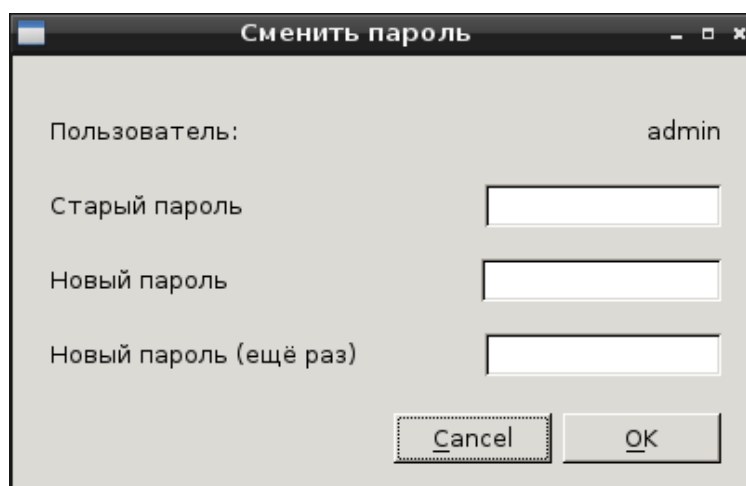


Рисунок 45 – Окно смены пароля

Смена пароля пользователя осуществляется только в том случае, если старый пароль соответствует старому паролю пользователя, а новый пароль введён одинаково дважды. В противном случае пользователь получает сообщение об ошибке смены пароля.

При нажатии на вкладку «Добавить пользователя» открывается окно создания нового пользователя, представленное на рисунке 46.

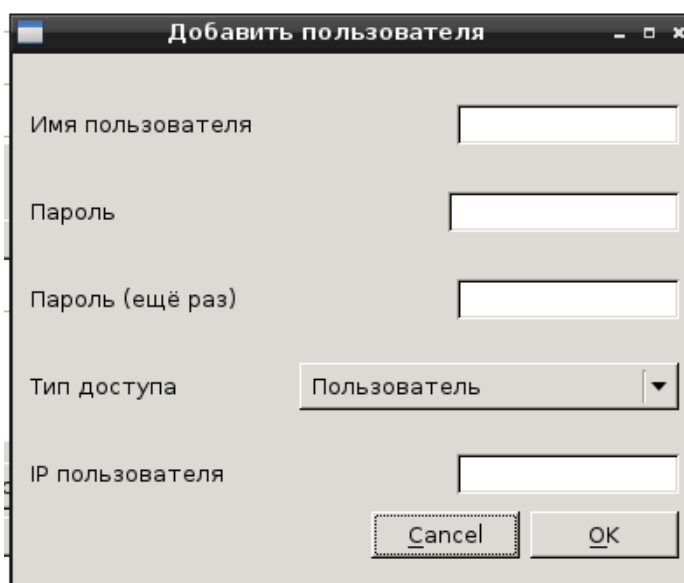


Рисунок 46 – Окно добавления пользователя

Новый пользователь будет создан при условии, что пользователь с таким именем ещё не зарегистрирован в системе, и пароль нового пользователя был одинаково введён дважды. В противном случае программа

выдаст сообщение об ошибке создания нового пользователя, представленное на рисунке 47.

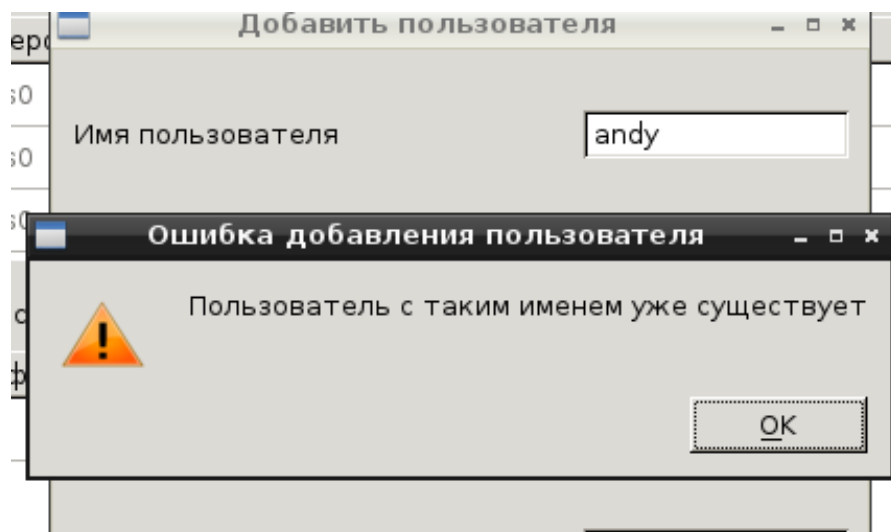


Рисунок 47 – Сообщение об ошибке добавления пользователя

При нажатии на вкладку «Изменить права доступа» открывается окно смены прав доступа, представленное на рисунке 48.

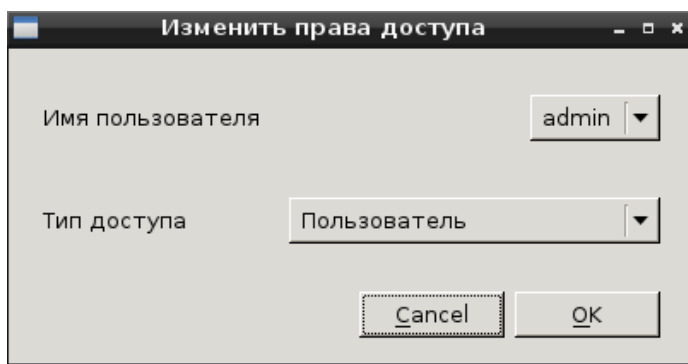


Рисунок 48 – Окно смены прав доступа

Значение по умолчанию в выпадающем списке «Имя пользователя» задано именем авторизованного в текущем сеансе работы. Значение по умолчанию в выпадающем списке «Тип доступа» задано «Пользователь». В случае неудачной попытки изменить права доступа, пользователь получит сообщение об ошибке изменения прав. В целях обеспечения информационной безопасности локальной сети, если пользователь текущего сеанса авторизован с правами доступа локального администратора, в выпадающем списке «Тип доступа» будет отсутствовать вариант «Системный администратор».

При нажатии на кнопку «Удалить пользователя» будет открыто окно удаления пользователя, представленное на рисунке 49.

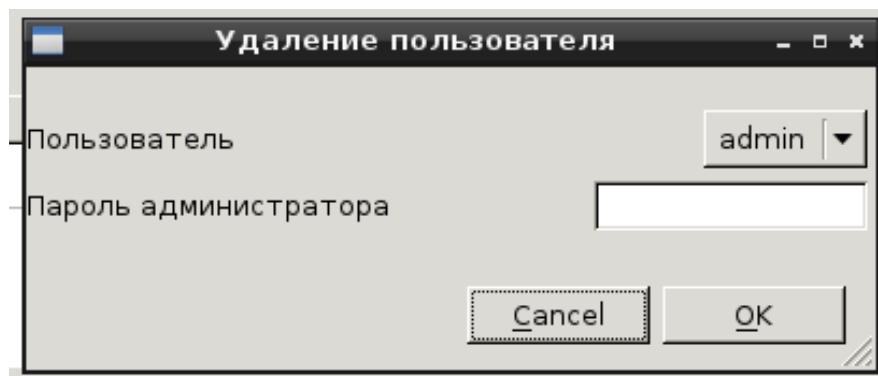


Рисунок 49 – Окно удаления пользователя

Значение по умолчанию в выпадающем списке «Пользователь» задано именем пользователя, авторизовавшемся в системе в текущем сеансе работы. В случае неудачной попытки удалить пользователя, программа выдаст сообщение об ошибке удаления пользователя.

3.3.3 Построение диаграммы классов для разработанного графического программного обеспечения «Настройка сети»

Диаграмма классов предназначена для демонстрации общей структуры иерархии классов, их атрибутов, методов и взаимосвязей. Для построения диаграммы классов для графического программного обеспечения «Настройка сети» был составлен полный список классов, которые входят в данное программное обеспечение. [6]

Полный список классов, которые входят в ПО «Настройка сети»:

- класс главного окна `CMainDlg`;
- класс диалогового окна авторизации `CAutorDlg`;
- класс диалогового окна смены пароля `CChPasswdDlg`;
- класс диалогового окна добавления пользователя `CAddUserDlg`;
- класс диалогового окна смены прав пользователя `CChmodDlg`;
- класс диалогового окна удаления пользователя `CCillUserDlg`;
- класс записи из конфигурационного файла `CRec`.

Как и любой другой язык, UML имеет собственные правила оформления моделей и синтаксис. С помощью графической нотации UML можно визуализировать систему, объединить все компоненты в единую структуру, уточнять и улучшать модель в процессе работы.

С помощью программного обеспечения Doxygen был исследован исходный код графического программного обеспечения «Настройка сети» и построены UML-диаграммы классов.

На рисунке 50 представлена UML-диаграмма класса CRec.

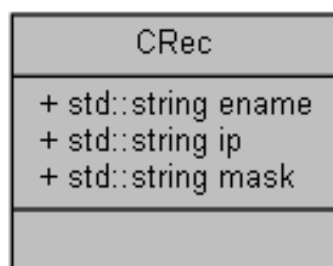


Рисунок 50 – UML-диаграмма класса CRec

На рисунке 51 представлена UML-диаграмма класса CAutorDlg.

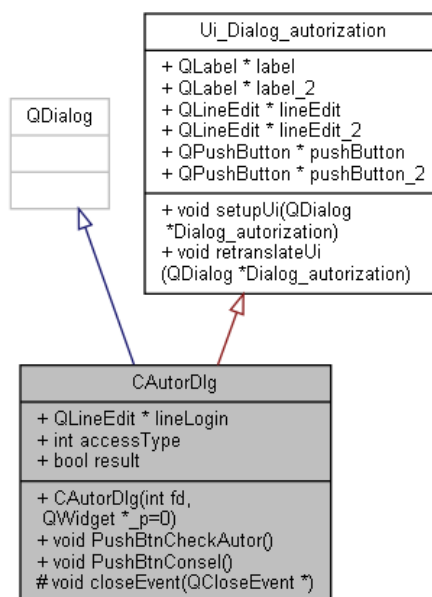


Рисунок 51 – UML-диаграмма класса CAutorDlg

На рисунке 52 представлена UML-диаграмма класса CMainDlg.

На рисунке 53 представлена UML-диаграмма класса ChPasswdDlg.

На рисунке 54 представлена UML-диаграмма класса CAddUserDlg.

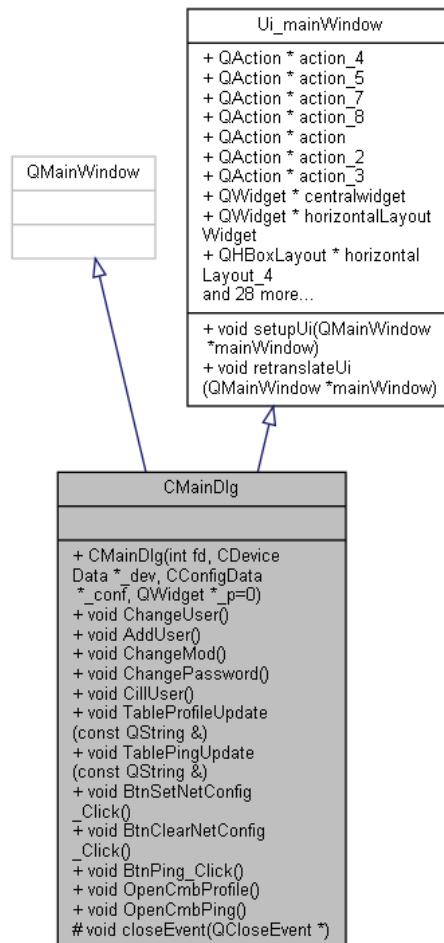


Рисунок 52 – UML-диаграмма класса CMainDlg

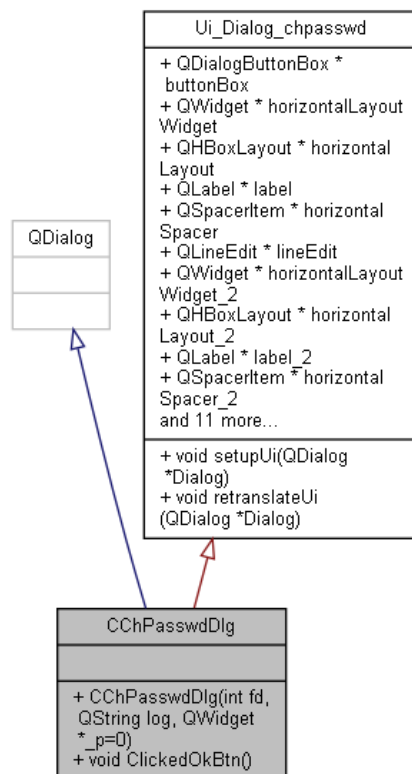


Рисунок 53 – UML-диаграмма класса ChPasswdDlg

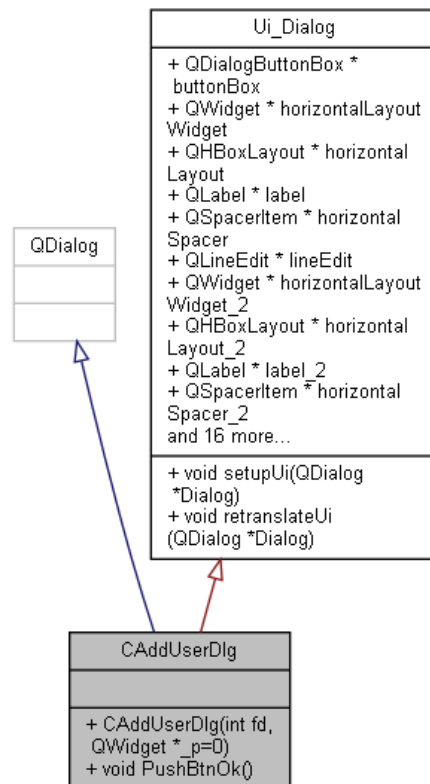


Рисунок 54 – UML-диаграмма класса CAddUserDlg

На рисунке 55 представлена UML-диаграмма класса CChmodDlg.

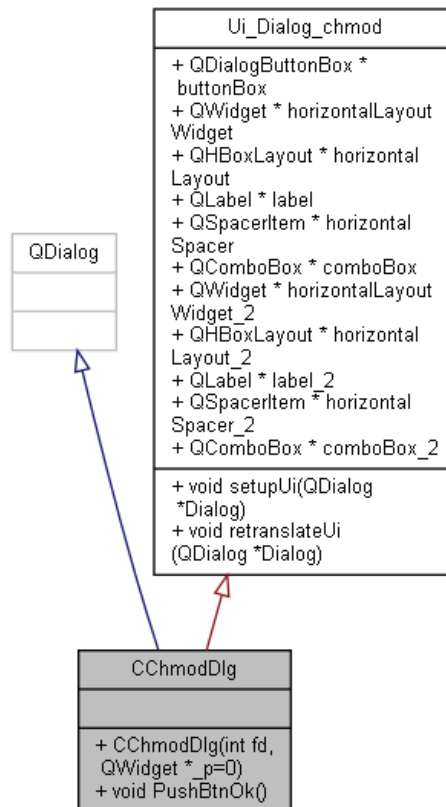


Рисунок 55 – UML-диаграмма класса CChmodDlg

На рисунке 56 представлена UML-диаграмма класса CCillUserDlg.

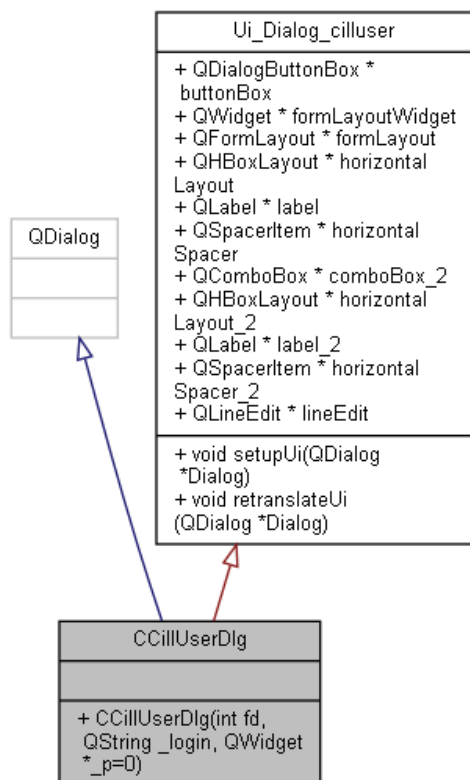


Рисунок 56 – UML-диаграмма класса CCillUserDlg

4 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

4.1 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации

Основным фактором, влияющим на производительность труда людей, работающих в сфере защиты информации, являются комфортные и безопасные условия труда. Так как работа специалиста по информационной безопасности выполняется в основном на персональном компьютере, большинство вредных психофизических факторов этой работы сводятся именно к вредным факторам работы с персональным компьютером.

Условия труда человека, работающего с персональным компьютером, определяются:

- качеством организации рабочего места;
- условиями производственной среды (освещение, микроклимат, шум, электромагнитные и электростатические поля, визуальные эргономические параметры дисплея и т. д.);
- характеристиками информационного взаимодействия пользователя и персональных компьютеров.

К психофизиологическим вредным и опасным факторам в работе специалиста по информационной безопасности относятся:

- напряжение зрения и внимания;
- интеллектуальные, эмоциональные и длительные статические нагрузки;
- монотонность труда;
- большой объем информации, обрабатываемый в единицу времени;
- нерациональная организация рабочего места.

Все эти факторы могут приводить к стрессу у работника. Стресс – это эмоциональное напряжение, внутренние переживания, вызванные различными событиями. Во время работы с персональным компьютером он возникает, в первую очередь, при повреждении или потере информации. Причинами этих событий могут быть: отсутствие резервных копий, рабочие ошибки, компьютерные вирусы, поломки жестких дисков. Иногда стрессы являются причиной инфарктов. Стрессы бывают не только эмоционально отрицательными, но и эмоционально положительными. Они могут быть кратковременными или долгосрочными, острыми или хроническими, физиологическими или психологическими (информационными и эмоциональными). Реакция организма на стресс запускает биохимические процессы, направленные на подавление экстремальной ситуации. Стрессовые ситуации и переживания, связанные с ними, вызывают в организме множественные негативные изменения. [7]

Исследования влияния условий труда на операторов автоматизированных производств, работающих с ВТ, выполненные в 80-х годах кафедрой охраны труда и эргономики Ленинградского политехнического института под руководством профессора В.И. Барабаша, подтверждают негативное влияние на психику человека. Наряду с жалобами на зрение, на боли в различных частях тела у 57,7 % обследованных операторов были отмечены жалобы общевротического характера: повышенная общая утомляемость, снижение бодрости, работоспособности, тяжесть в голове, плохой сон, головные боли и др. У большого количества работающих с компьютерами (40,3 %) выявились стойкие нервно-психические нарушения в виде повышенной раздражительности, ощущений беспокойства и депрессивных состояний.

У людей, которые заняты работой на вычислительной технике, выявлено значительно более выраженное нервно-сенсорное напряжение по показателям критической частоты слияния мельканий и электрической чувствительности глаза по сравнению с другими профессиональными

группами. Характер изменения этих показателей свидетельствует о повышенной возбудимости и нестабильности их нервной системы.

Результаты исследования показали наличие волнения в процессе работы у 85 % испытуемых, утомляемость из-за монотонной обстановки - у 60 %, снижение работоспособности к концу рабочего дня - у 63 %. После работы у испытуемых отмечались раздражительность - у 70 %, тяжелое утомление - у 68 %, поверхностный сон - у 35 %. К приёму успокаивающих средств прибегли 30 % испытуемых.

Приведенные субъективные характеристики труда операторов ЭВМ подтверждаются данными объективного исследования "арифметические действия", по результатам которого умственная утомляемость отмечалась у 33 % обследуемых.

Но стресс в работе специалиста по информационной безопасности вызывает не только работа за компьютером. Должность специалиста по ИБ является очень ответственной. От принимаемых им решений зависит конфиденциальность, целостность и доступность защищаемой информации. Ошибки, совершённые специалистом по ИБ, могут оказаться очень дорогими для организации, в которой он работает. Они могут нанести непоправимый материальный или моральный (репутационный) ущерб организации.

Профессиональный стресс — это такое состояние работника, которое вызвано напряжением, возникшим под воздействием эмоциональных и экстремальных факторов, вызванных профессиональной деятельностью. В отличие от физиологического стресса, стресс профессиональный представляет собой результат осознания ответственности субъектом деятельности за результат своих действий и последующих событий, но невозможность управления ими и их полного контроля. Профессиональный стресс влияет на здоровье работника и работоспособность. Вследствие этого страдают и производительность, и качество его труда. Профессиональный стресс, как и любой другой вид стресса, имеет и психологические, и физиологические особенности проявления и развития.

Выделяют следующие факторы, связанные с трудовой деятельностью:

- интенсивность работы;
- выделенное время на выполнение работы;
- внештатные ситуации, которые меняют привычный уклад работы;
- отсутствие чётко обозначенных трудовых задач;
- многозадачность, сложность задач;
- недостаток ресурсов или информации;
- отсутствие безопасности работы.

Организационные факторы профессионального стресса — это факторы, которые связаны с правилами и традициями производственной организации.

К ним относятся:

- противоречия в регламентах и инструкциях;
- несправедливые формы оплаты труда;
- несправедливые придижки;
- несправедливые оценки процессов и результатов труда со стороны

администрации.

Личностные факторы :

- повышенная личностная тревожность;
- пониженная самооценка, склонность к чувству вины;
- выраженная эмоциональная лабильность;
- наружный фокус контроля (в жизни опираются на случайность, удачу, достижения и мнения других людей);
- пассивность, избегающие стратегии выхода из сложных ситуаций;
- проблемы в профессиональной подготовке работников.

4.2 Методы и средства физической культуры, снижающие воздействие психофизических факторов, влияющих на психоэмоциональное состояние специалиста по защите информации.

Некоторые исследования подтверждают, что регулярные физические упражнения средней интенсивности способствуют появлению своего рода иммунитета к стрессу. Доказано, что физически активные люди более устойчивы к нервным срывам, чем люди, ведущие малоподвижный образ жизни. Таким образом, постоянные занятия физическими упражнениями тренируют не только тело человека, но и поднимают настроение, повышают самооценку, закаляют нервную систему.

Одно из таких исследований было проведено Ю. Л. Ханиным, Г. В. Булановой. Цель этого исследования заключалась в изучении влияния на эмоциональное состояние студентов физической культуры. В эксперименте приняло участие 229 человек в возрасте от 18 до 22 лет. Результаты показали снижение уровня тревоги после занятий спортом вне зависимости от вида физической активности.

Таким образом, представленные результаты позволяют с достаточной уверенностью говорить о том, что занятия физической культурой способствуют снижению реактивной тревоги.

В результате физических упражнений происходит улучшение кровообращения. Это улучшает работу сердца, так как кислород, который активно поступает в лёгкие, улучшает работу дыхательной системы. В результате улучшения обмена веществ выделяется эндорфин - гормон счастья. Этот процесс значительно сужает уровень стресса. [8]

Все упражнения, основная нагрузка при которых приходится на ноги, такие как ходьба, езда на велосипеде или танцы, заставляют сокращаться мышцы бедер и икр. Сокращение этих мышц оказывает сильное воздействие на вены и лимфатические сосуды, что способствует усиленному перекачиванию к сердцу лимфы и венозной крови. Активное сгибание,

разгибание и вытягивание стоп улучшает приток лимфы к сердцу, стимулируя подошвенный рефлекс. Такие движения характерны для танцев, и особенно для бальных танцев.

Упражнения, стимулирующие работу брюшного пресса, повышают тонус и укрепляют мышцы живота. Укрепление брюшного пресса благотворно влияет на осанку, это вызвано тем, что таз возвращается на свое анатомически правильное место, и открывается пережатый узкий паховый канал. В результате такого процесса улучшается ток венозной крови и лимфы и уменьшается давление на бедренную вену.

Одним из видов физических активностей, которые способны помочь справиться со стрессом, несомненно, является плавание. Оно оказывает положительный эффект как на психическое, так и на соматическое здоровье человека. Его воздействие особенно благоприятно для сердечно-сосудистой и дыхательной системы, для мышечного аппарата (мышцы рук, бедер, ягодиц, живота, плечей), суставов (бедро, руки, шея) и опорно-двигательного аппарата. Плавание, так же, является одним из видов закаливания, которое оказывает положительное влияние на иммунную систему. Это является очень важным фактором при борьбе со стрессом, так как в первую очередь при сильных нагрузках на организм ослабевает именно иммунная система. Занятия плаванием оказывают положительное влияние на центральную нервную систему, что способствует формированию сильного и уравновешенного типа нервной деятельности. Плавание уравнивает процессы возбуждения и торможения, улучшает кровоснабжение мозга.

Приятные ассоциации, возникающие при мысли о плавании, благотворно влияют на состояние психики, способствуют формированию позитивного эмоционального фона. Плавание помогает побороть страхи, снимает усталость, помогает при тревоге и депрессии, делает сон более качественным, поднимает настроение, улучшает внимание и память. К одним из больших преимуществ плавания можно отнести и то, что им могут заниматься люди с практически любым состоянием здоровья, уровнем

физической подготовки и в любом возрасте. Нельзя не отметить благотворное влияние воды на организм. Вода расслабляет организм, снимает напряжение в мышцах и повышает работоспособность.

Ещё одним видом физической активности, доступным практически каждому, является йога. Йога — это понятие в индийской культуре, которое в широком смысле означает совокупность разнообразных психических, духовных и физических практик, которые были разработаны в разных направлениях буддизма и индуизма, и которые нацелены на управление психическими и физиологическими функциями организма для достижения человеком определённого духовного и психического состояния.

Йога отличается широким разнообразием направлений и техник, одни из которых представляют собой комплекс статических упражнений, вторые — динамических, третьи посвящены дыханию, а четвертые сосредоточены на медитации. Такой большой выбор может позволить каждому найти наиболее подходящее именно ему направление.

На занятиях йогой выполняются упражнения, которые направлены на снятие мышечных спазмов в тех зонах, где в первую очередь присутствует реакция на стресс (тазовый пояс, плечевой пояс, живот, мышцы вдоль позвоночника и др.) Эти упражнения смежны с техниками глубокой релаксации. Йога формирует нормальную физиологическую реакцию на стресс, завершая цепочку реакций, и избавляет таким образом от последствий стресса. Именно поэтому занятия йогой способствуют быстрому улучшению состояния при таких проблемах, как бессонница, нарушения менструального цикла, неврозы, головные боли, гастриты, язва желудка и двенадцатиперстной кишки, гипертония, вегетососудистая дистония, остеохондроз и грыжи позвоночника, заболевания нервной системы. Йога стабилизирует эмоциональный фон и обеспечивает профилактику стресса, позволяя противостоять ему в дальнейшем.

Следующий вид эффективной физической активности — спортивные игры. Они представляют собой ряд самостоятельных видов спорта, которые

связаны с игровым противоборством отдельных спортсменов или команд, и проводятся по определенным правилам. Существует большое разнообразие спортивных игр, которые рассчитаны на разное количество человек. В нашей стране к наиболее популярным спортивным играм можно отнести хоккей, футбол, волейбол, бадминтон, теннис, баскетбол и многие другие.

Во время игры человек может выплеснуть все накопившиеся негативные эмоции. Занятие любыми спортивными играми позволяет научиться быстрой реакции, снизить уровень агрессии, избавиться от чувства одиночества, повысить самооценку и многим другим важным навыкам, а также укрепить все мышцы тела.

Бег — вид физической активности, который практически не требует материальных затрат. Это один из способов передвижения человека и животных, отличающийся наличием так называемой «фазы полёта». Бег осуществляется в результате скоординированной сложной деятельности конечностей и скелетных мышц. Для него характерен тот же цикл движений, что и при ходьбе, те же действующие силы и функциональные группы мышц. Отличием бега от ходьбы является отсутствие при беге фазы двойной опоры. Во время бега задействованы практически все суставы и группы мышц.

Этот вид спорта практически не требует никакого снаряжения, но при этом приносит неоценимую пользу организму. При занятиях бегом улучшается работа сердечно сосудистой и дыхательной систем, укрепляется как мышечный скелет, так и иммунитет. Кроме того, он способствует выработке гормона счастья — эндорфина, благодаря этому повышается эмоциональный фон.

Из всего выше сказанного можно сделать вывод о том, что существует большое количество способов борьбы со стрессом. Одними из самых эффективных и в то же время доступных практически каждому способов являются спорт и физические нагрузки. Из огромного разнообразия существующих видов спорта каждый легко может подобрать наиболее подходящий для него. Регулярная физическая активность сможет привести

человека не только в отличную физическую форму, но и к стабильному эмоциональному состоянию. Она сможет значительно повысить стрессоустойчивость, что немаловажно в современных условиях жизни. Не обязательно прибегать к изнурительным и сложным тренировкам, вполне достаточно умеренных физических нагрузок. Но не стоит забывать о том, что бороться со стрессом следует комплексно, помимо регулярных занятий спортом следует также сбалансированно питаться, соблюдать режим дня и избегать стрессогенных факторов.

Заключение

В результате выполнения выпускной квалификационной работы была разработана система централизованного управления узлами сети с реализацией функций защиты от несанкционированного доступа.

Разработанная система состоит из компонентов:

- сервер авторизации;
- графическое программное обеспечение «Настройка сети», представляющее собой сторону клиента.

В ходе выполнения проектирования был разработан алгоритм работы сервера авторизации, был разработан протокол взаимодействия сервера с клиентской стороной, была написана программа на языке C++, реализующая данный алгоритм работы, был разработан алгоритм работы графического программного обеспечения «Настройка сети», были разработаны макеты окон данного графического программного обеспечения, была написана программа на языке C++ с использованием пакета Qt4, реализующая алгоритм работы для программы «Настройка сети».

Таким образом, задание на дипломное проектирование выполнено в полном объёме.

Список используемых источников

1 Константинова, М.Д. Система централизованного мониторинга узлов сети с реализацией функций защиты от угроз несанкционированного доступа / М.Д. Константинова, В.А. Мали // Информационные технологии в науке и образовании. Проблемы и перспективы : сб. ст. по материалам VIII Всерос. межвуз. науч.-прак. конф. Под редакцией Л.Р. Фионовой. - Пенза: Изд-во ПГУ, 2021. - С. 409-416.

2 Банк данных угроз безопасности информации ФСТЭК [Электронный ресурс] // <https://bdu.fstec.ru>: информ.-справочный портал. Режим доступа: <https://bdu.fstec.ru/threat> (свободный).

3 ГОСТ Р 34.11-94. Информационная технология. Криптографическая защита информации. Функция хэширования. Процедура вычисления хэш-функции [Электронный ресурс]. – М.: Издательство стандартов, 1994. – Режим доступа: <https://files.stroyinf.ru/Data2/1/4294824/4294824580.pdf>

4 ГОСТ Р ИСО/МЭК 7498-1-99. Информационная технология (ИТ). Взаимосвязь открытых систем. Базовая эталонная модель. Часть 1. Базовая модель. [Электронный ресурс]. – М.: МНИЦ, 1999. – Режим доступа: <http://docs.cntd.ru/document/1200028699>// – Загл. с экрана

5 Нейл Мэтью, Ричард Стоунс. Основы программирования в Linux // СПб.: «БХВ-Петербург»/2009. – С. 702

6 Леоненков А.В. Самоучитель UML // СПб.: «БХВ-Петербург»/2004. – С. 54

7 ГОСТ 12.0.003-2015. Система стандартов по безопасности труда. Основные и вредные производственные факторы. Классификация [Электронный ресурс]. – М.: Стандартинформ, 2016. – Режим доступа: <http://www.internet-law.ru/gosts/gosts/62075>// - Загл. С экрана.

8 Human med [Электронный ресурс] // human-med.info: информ.-справочный портал. Режим доступа: <http://human-med.info/7-prostykh-uprazhnenij-dlya-snyatiya-napryazheniya-i-stressa.htm> (свободный).

ПРИЛОЖЕНИЕ А

(обязательное)

Исходные файлы сервера авторизации, отвечающие за работу в терминале

Текст исходного файла main.cpp.

```
#include "params.h"
#include "client.h"
#include <netinet/in.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/time.h>
#include <sys/types.h>
#include <iostream>
#include <string.h>
#include <stdio.h>
#include <unistd.h>
#include <thread>

static void threadFunction( C_Client * cln ){
    cln->reqTreatment();
};

int main( const int _argc, const char ** _argv ){
    struct s_Params params;
    int ret;

    // Разбор параметров
    ret = ParseParam( _argc, _argv, &params );
    if( ret == -1 ){
        std::cout << "Параметры с ошибками!" <<
std::endl;
        std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
        return -1;
    };

    struct sockaddr_in addr;
    int master_socket;
    fd_set readfds;

    // создаю потоковый сокет на базе интернет-
    протоколов ARPA
    master_socket = socket(AF_INET, SOCK_STREAM, 0 );
    if( master_socket < 0 ){
```

```

        std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
        return -1;
    };

    memset( &addr, 0, sizeof( master_socket ) );
    addr.sin_family = AF_INET;
    addr.sin_addr.s_addr = 0;
    addr.sin_port = htons( params.port );

    int opt = 1;
    // устанавливаю уровень сокета - SOL_SOCKET (для
уровня API), и разрешаю повторное использование адресов
    if( setsockopt( master_socket, SOL_SOCKET,
SO_REUSEADDR, (char*)&opt, sizeof(opt) ) < 0 ){
        std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
        return -1;
    };
    if( bind( master_socket, (struct sockaddr*)&addr,
sizeof( addr ) ) < 0 ){
        std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
        return -1;
    };
    if( listen( master_socket, 3 ) < 0 ){
        std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
        return -1;
    };

    while(1){
        FD_ZERO(&readfds);
        FD_SET(master_socket, &readfds );

        int activity = select( master_socket + 1,
&readfds, NULL, NULL, NULL );
        if( activity <= 0 ){
            close( master_socket );
            std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
            return -1;
        };
        if( FD_ISSET( master_socket, &readfds ) ){
            struct sockaddr_in client;
            int clnlen = sizeof( client );
            int connect_fd = accept( master_socket,
(struct sockaddr*)&client, (socklen_t*)&clnlen);

```

```

        if( connect_fd < 0 ){
            continue;
        };
        C_Client    cln(    &client,    connect_fd,
params);

        std::thread th( threadFunction, &cln );
        th.detach();
    };

};

return 0;
};

```

Текст исходного файла params.h.

```

#ifndef __PARAMS_H
#define __PARAMS_H

#include <stdint.h>

//! Структура для хранения параметров программы
struct s_Params{
    uint16_t port;                //!< Порт на котором
ождается соединение
    char * authorization_file;    //!< Каталог для
сохранения файлов
    s_Params();
};

//! Функция разбора параметров при вызове программы
/*!
    @param _argc  Количество параметров
    @param _argv  Параметры
    @param _params Полученные в результате работы
функции параметры строки
    @return      0    - все параметры разобраны
успешно
                1    - вызов справки
               -1    - ошибка разбора параметров
*/
int ParseParam( const int _argc, const char *_argv[],
struct s_Params *_params );

#endif

```

Текст исходного файла params.cpp.

```

#include "params.h"
#include <iostream>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

s_Params::s_Params() {
    // порт по умолчанию
    port = 5000;
    authorization_file = strdup( "../log.ini" );
};

//! Функция разбора параметров при вызове программы
/*!
    @param _argc  Количество параметров
    @param _argv  Параметры
    @param _params Полученные в результате работы
функции параметры строки
    @return      0      - все параметры разобраны
успешно
                1      - вызов справки
                -1     - ошибка разбора параметров
*/
int ParseParam( const int _argc, const char *_argv[],
struct s_Params *_params ){
    int pos = 1;

    // TRUE - если одна из двух строк равна текущему
параметру
#define STREQ2( _str1, _str2 ) (( 0 == strcmp(
_argv[pos], _str1 ) )||( 0 == strcmp( _argv[pos], _str2 )
))

    while( _argc > pos ){

        if( STREQ2( "-p", "--port" ) ){
            if( _argc <= ( pos + 1 ) ){
                std::cout << "Недостаточно параметров
для " << _argv[pos] << std::endl;
                return -1;
            }
            pos++;
            //--
            char *endptr;
            _params->port = strtol( _argv[pos],
&endptr, 10 );
            if(( 0 != errno )||( _argv[pos] == endptr
)||( '\0' != *endptr )){

```

```

                                printf( "Неправильное значение --port
'%s'!\n", _argv[pos] );
                                return -1;
                                };
                                //--
                                }
                                if( STREQ2( "-a", "--authorization_file" ) ){
                                    if( _argc <= ( pos + 1 ) ){
                                        std::cout << "Недостаточно параметров
для " << _argv[pos] << std::endl;
                                        return -1;
                                    }
                                    pos++;
                                    free( _params->authorization_file );
                                    // Каталог для сохранения файлов
                                    _params->authorization_file = strdup(
_argv[pos] );
                                }
                                pos++;
                                }
                                return 0;
                                };

```

ПРИЛОЖЕНИЕ Б

(обязательное)

Исходные файлы сервера авторизации, отвечающие за обработку запросов

Текст исходного файла client.h.

```
#ifndef __CLIENT_H
#define __CLIENT_H

#include "params.h"
#include <netinet/in.h>

class C_Client{

    public:
        C_Client( struct sockaddr_in *_client, int
_connect_fd, s_Params _params );
        void reqTreatment();

    private:
        char ip[INET_ADDRSTRLEN];
        struct sockaddr_in * client;
        int connect_fd;
        s_Params params;

        void getIP();

};

#endif
```

Текст исходного файла client.cpp.

```
#include "client.h"
#include "workers.h"
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <unistd.h>
#include <iostream>
#include <string.h>
#include <stdio.h>

C_Client::C_Client ( struct sockaddr_in * _client, int
_connect_fd, s_Params _params ){
    client = _client;
```

```

        connect_fd = _connect_fd;
        params = _params;
        getIP();
    }

    void C_Client::getIP(){
        // struct sockaddr_in * sin = (struct sockaddr_in*)
client;
        inet_ntop(AF_INET,      &client->sin_addr,      ip,
sizeof(ip) );
        std::cout << "IP client: " << (std::string)ip <<
std::endl;
    }

    void C_Client::reqTreatment(){
        std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
        while(1){
            unsigned char operation;
            int r_size = recv( connect_fd, &operation, 1,
0 );
            if( r_size == 1 ){
                CWorkers      workers(      connect_fd,
params.authorization_file, (std::string)ip );
                if( operation == 0x00 ){
                    std::cout << "go to authorize" <<
std::endl;
                    workers.authorization();
                }else if( operation == 0x01 ){
                    std::cout << "go to ch passwd" <<
std::endl;
                    workers.chPasswd();
                }else if( operation == 0x02 ){
                    std::cout << "go to add user" <<
std::endl;
                    workers.addUser();
                }else if( operation == 0x03 ){
                    std::cout << "go to ch mod" <<
std::endl;
                    workers.chMod();
                }else if( operation == 0x04 ){
                    std::cout << "go to cill user" <<
std::endl;
                    workers.cillUser();
                }else if( operation == 0x05 ){
                    std::cout << "go to gen salt" <<
std::endl;
                    workers.genSalt();

```

```

        }else if( operation == 0x06 ){
            std::cout << "go to get all users" <<
std::endl;
            workers.getAllUsers();
        }else if( operation == 0x07 ){
            std::cout << "initalize" <<
std::endl;
            const char * answer = "ok\0";
            send( connect_fd, answer,
strlen(answer), 0);
            std::cout << __FILE__ << " " <<
__LINE__ << std::endl;
        }else if( operation == 0x08 ){
            std::cout << "close client" <<
std::endl;
            break;
        }
    }
}
shutdown( connect_fd, SHUT_RD );
close( connect_fd );
}

```

Текст исходного файла headers.h.

```

#ifndef __HEADERS_H
#define __HEADERS_H

#include "openssl/sha.h"
#include <string>
#include <stdio.h>

struct s_autorHeader{
    int size_userName;
    char passwd[65];
} __attribute__((packed));

struct s_chPasswdHeader{
    int size_userName;
    char passwd_old[65];
    char passwd_new[65];
} __attribute__((packed));

struct s_addUserHeader{
    int size_userName;
    char passwd[65];
    char accessType;
    char ip[16];
} __attribute__((packed));

```

```

struct s_chModHeader{
    int size_userName;
    char accessType;
} __attribute__((packed));

```

```

struct s_cillUserHeader{
    int size_userName;
} __attribute__((packed));

```

```

#endif

```

Текст исходного файла workers.h.

```

#ifndef __WORKERS_H
#define __WORKERS_H

#include "autorbd.h"
#include <string>

class CWorkers : CAutorBD
{
    public:
        CWorkers(    int    _fd,    std::string    _bdFile,
std::string _ip );
        int authorization();
        int chPasswd();
        int addUser();
        int chMod();
        int cillUser();
        int genSalt();
        int getAllUsers();

    private:
        int fd;
        std::string bdFile;
        std::string ip;

        bool startWork();
        bool checkName( std::string userName );
        bool    checkPasswd(    std::string    userName,
std::string passwd );
        bool checkIP( std::string userName );
};

#endif

```

Текст исходного файла workers.cpp.

```
#include "workers.h"
#include "headers.h"
#include "autorbd.h"
#include <iostream>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <stdio.h>
#include <string.h>

CWorkers::CWorkers( int _fd, std::string _bdFile,
std::string _ip ){
    fd = _fd;
    bdFile = _bdFile;
    ip = _ip;
};

bool CWorkers::startWork(){
    initFile( bdFile );
    int res = readAutorFile();
    if( res != 0 ){
        return false;
    }
    return true;
};

bool CWorkers::checkName( std::string userName ){
    CAutorData::iterator it;
    it = autor.find(userName);
    if( it != autor.end() ){
        return true;
    }
    return false;
};

bool CWorkers::checkPasswd( std::string userName,
std::string passwd ){
    CAutorData::iterator it;
    it = autor.find(userName );
    if( it->second.passwd != passwd ){
        return false;
    }
    return true;
};

bool CWorkers::checkIP( std::string userName ){
```

```

        CAutorData::iterator it;
        it = autor.find(userName);
        if( it->second.ip != ip ){
            return false;
        }
        return true;
    };

    int CWorkers::authorization(){
        char res_code = 0x01;
        s_autorHeader header;
        recv( fd, &header, sizeof(header), 0);
        if( startWork() != true ){
            send( fd, &res_code, 1, 0);
            std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
            return -1;
        }

        char * userName = new char[header.size_userName];
        memset(userName, header.size_userName, 0);
        recv( fd, userName, header.size_userName, 0);

        if( checkName( userName ) != true ){
            res_code = 0x02;
            send( fd, &res_code, 1, 0);
            std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
            return -1;
        }

        std::cout << "authorize passwd: " << header.passwd
<< std::endl;
        if( checkPasswd( userName, header.passwd ) != true
){
            res_code = 0x02;
            send( fd, &res_code, 1, 0);
            std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
            return -1;
        }

        if( checkIP( userName ) != true ){
            res_code = 0x03;
            send( fd, &res_code, 1, 0);
            std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
            return -1;
        }
    }

```

```

    }

    res_code = 0x00;
    send( fd, &res_code, 1, 0);
    CAutorData::iterator it;
    it = autor.find(userName );
    int mod = it->second.mod;
    send( fd, &mod, 4, 0);
    return 0;
};

int CWorkers::addUser(){
    char res_code = 0x01;
    s_addUserHeader header;
    recv( fd, &header, sizeof(header), 0);
    if( startWork() != true ){
        send( fd, &res_code, 1, 0);
        return -1;
    }

    char * userName = new char[header.size_userName];
    recv( fd, userName, header.size_userName, 0);
    std::cout << "user name add: " << userName <<
std::endl;

    if( checkName( userName ) != false ){
        res_code = 0x02;
        send( fd, &res_code, 1, 0);
        return -1;
    }
    std::string ip = (std::string)header.ip;
    int i = 0;
    while (ip[i] == '0'){
        ip.erase(i,1);
        i++;
    }
    writeToFile(          userName,          header.passwd,
(int)header.accessType, ip );

    res_code = 0x00;
    send( fd, &res_code, 1, 0);
    return 0;
};

int CWorkers::chMod(){
    char res_code = 0x01;
    s_chModHeader header;
    recv( fd, &header, sizeof(header), 0);

```

```

        if( startWork() != true ){
            send( fd, &res_code, 1, 0);
            return -1;
        }

        char * userName = new char[header.size_userName];
        recv( fd, userName, header.size_userName, 0);

        CAutorData::iterator it;
        it = autor.find(userName );

        if( remove( bdFile.c_str() ) != 0 ){
            send( fd, &res_code, 1, 0);
            return -1;
        }
        CAutorData::iterator rewrite;
        for( rewrite = autor.begin(); rewrite !=
autor.end(); rewrite++){
            int res = 0;

            if( rewrite->first == it->first ){
                res = writeToFile( it->first, rewrite-
>second.passwd, (int)header.accessType, rewrite->second.ip
);
            } else {
                res = writeToFile( rewrite->first,
rewrite->second.passwd, rewrite->second.mod, rewrite-
>second.ip );
            }
            if ( res < 0 ){
                send( fd, &res_code, 1, 0);
                return -1;
            }
        }
        res_code = 0x00;
        send( fd, &res_code, 1, 0);
        return 0;
    };

int CWorkers::chPasswd(){
    char res_code = 0x01;
    s_chPasswdHeader header;
    recv( fd, &header, sizeof(header), 0);
    if( startWork() != true ){
        send( fd, &res_code, 1, 0);
        return -1;
    }
}

```

```

char * userName = new char[header.size_userName];
recv( fd, userName, header.size_userName, 0);

if( checkPasswd(  userName, header.passwd_old) !=
true ){
    res_code = 0x02;
    send( fd, &res_code, 1, 0);
    return -1;
}
CAutorData::iterator it;
it = autor.find( userName );

if( remove( bdFile.c_str() ) != 0 ){
    send( fd, &res_code, 1, 0);
    return -1;
}
CAutorData::iterator rewrite;
for(  rewrite  =  autor.begin();  rewrite  !=
autor.end(); rewrite++ ){
    int res = 0;

    if( rewrite->first == it->first ){
        res      =      writeToFile(      it->first,
header.passwd_new,  rewrite->second.mod,  rewrite->second.ip
);
    } else {
        res      =      writeToFile(      rewrite->first,
rewrite->second.passwd,      rewrite->second.mod,      rewrite-
>second.ip );
    }
    if ( res < 0 ){
        send( fd, &res_code, 1, 0);
        return -1;
    }
}
res_code = 0x00;
send( fd, &res_code, 1, 0);
return 0;
};

int CWorkers::cillUser(){
    char res_code = 0x01;
    s_cillUserHeader header;
    recv( fd, &header, sizeof(header), 0);
    if( startWork() != true ){
        send( fd, &res_code, 1, 0);

```

```

        std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
        return -1;
    }

    char * userName = new char[header.size_userName];
    recv( fd, userName, header.size_userName, 0);

    CAutorData::iterator it;
    it = autor.find( userName );

    if( remove( bdFile.c_str() ) != 0 ){
        send( fd, &res_code, 1, 0);
        std::cout << __FILE__ << " " << __LINE__ <<
std::endl;
        return -1;
    }
    CAutorData::iterator rewrite;
    for( rewrite = autor.begin(); rewrite !=
autor.end(); rewrite++ ){
        int res = 0;

        if( rewrite->first != it->first ){
            res = writeToFile( rewrite->first,
rewrite->second.passwd, rewrite->second.mod, rewrite-
>second.ip );
        }
        if ( res < 0 ){
            send( fd, &res_code, 1, 0);
            std::cout << __FILE__ << " " << __LINE__
<< std::endl;
            return -1;
        }
    }
    res_code = 0x00;
    send( fd, &res_code, 1, 0);
    return 0;
};

int CWorkers::genSalt(){
    const char * salt = "IBST\0";
    send( fd, salt, 5, 0);
    return 0;
};

int CWorkers::getAllUsers(){
    if( startWork() != true ){
        char res_code = 0xFF;

```

```

        send( fd, &res_code, 1, 0);
        return -1;
    }
    CAutorData::iterator it;
    int cnt = 0;
    for( it = autor.begin(); it != autor.end(); it++)
){
        cnt++;
    }
    std::cout << __FILE__ << ": send cnt = " << cnt
<<std::endl;
    send( fd, &cnt, 4, 0);
    for( it = autor.begin(); it != autor.end(); it++)
){
        int sUserName = it->first.length() + 1;
        send( fd, &sUserName, 4, 0);
        std::string userName = it->first + "\0";
        send( fd, userName.c_str(), sUserName, 0);
    }
    return 0;
};

```

ПРИЛОЖЕНИЕ В

(обязательное)

Исходные файлы сервера авторизации, отвечающие за работу с базой
данных авторизации

Текст исходного файла autorbd.h.

```
#ifndef __AUTORBD_H
#define __AUTORBD_H

#include <string>
#include <map>

class CUserData{
public:
    std::string passwd;
    int mod;
    std::string ip;
};

typedef std::map<std::string, CUserData> CAutorData;

class CAutorBD {
public:
    CAutorBD(){};

protected:
    std::string bdFile;
    CAutorData autor;

    void initFile( std::string str);
    int readAutorFile();
    bool fillAutorMap( std::string str );
    int writeToFile ( std::string login,
std::string passwd, int mod, std::string ip);
};

#endif
```

Текст исходного файла autorbd.cpp.

```
#include "autorbd.h"
#include "openssl/sha.h"
#include <fstream>
#include <iostream>
#include <QRegExp>
```

```

void CAutorBD::initFile( std::string str ){
    bdFile = str;
};

int CAutorBD::readAutorFile(){
    autor.clear();
    std::ifstream filestream( bdFile );
    if( !filestream ){
        return -1;
    }
    std::string str;
    int str_pos = 0;

    while(filestream ){
        getline( filestream, str );
        if( fillAutorMap(str) == false ){
            return -1;
        }
        str_pos++;
    }

    if( filestream.eof() ){
        filestream.close();
        return 0;
    };

    filestream.close();

    return str_pos;
};

bool CAutorBD::fillAutorMap( std::string str ){
    int str_len = str.length();
    if( str_len == 0 ){
        return true;
    };

    int null = 0;
    for( int i=0; i<str_len; i++ ){
        if(( str[i] == ' ' )||( str[i] == '(' ) ){
            null++;
        };
    };
    if( null == str_len ){
        return true;
    };

    if( 2 < str_len ){

```

```

        QRegExp rx("^[\t]*([A-Za-z0-9\\-_]*)[\t]*([A-Za-z0-9:\\\\.,@\\+\\*#&\\^!\\|?%=\"\\{\\}\\|\\|\\|\\|'<>_]*)[\t]*([0-2]*)[\t]*([\\.0-9]*)[\t]*$");
        if( -1 != rx.indexIn( QString::fromStdString(
str ) ) ){
            class CUserData userData;
            userData.passwd = rx.cap( 2
).toStdString();
            userData.mod = rx.cap( 3 ).toInt();
            userData.ip = rx.cap( 4 ).toStdString();
            autor.insert( std::make_pair(rx.cap( 1
).toStdString(), userData ) );
            return true;
        }
    }

    return false;
};

int CAutorBD::writeToFile( std::string userName,
std::string passwd, int mod, std::string ip ){

    std::ofstream filestream( bdFile, std::ios::app );
    if( !filestream ){
        return -1;
    }
    filestream << userName << " " << passwd << " " <<
std::to_string(mod) << " " << ip << std::endl;
    filestream.close();
    return 0;
};

```

ПРИЛОЖЕНИЕ Г

(обязательное)

Исходный файл графического программного обеспечения «Настройка сети», отвечающие за установку соединения с сервером авторизации

Текст исходного файла main.cpp.

```
#include "maindlg.h"
#include "configdata.h"
#include "read_config.h"
#include <netinet/in.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/time.h>
#include <netdb.h>
#include <QApplication>
#include <QDialog>
#include <QMessageBox>
#include <iostream>

int main( int argc, char *argv[] ){
    CConfigData conf;

    char conf_fname[] = "net.ini";
    char * fname = conf_fname;
    int err_pos = ReadConfig( &conf, conf_fname );
    CDeviceData dev;
    char dev_fname[] = "devicelist.ini";
    int d_err_pos = ReadDevices( &dev, dev_fname );
    if( d_err_pos > err_pos ){
        err_pos = d_err_pos;
        fname = dev_fname;
    }

    QApplication app( argc, argv );
    if( 0 != err_pos ){
        if( -1 == err_pos ){
            // Файл с конфигом не найден
            //      или      нет      доступа      на      чтение
конфигурационного файла
            QString text = QObject::trUtf8( "Ошибка
открытия '"
                                + fname
                                + QObject::trUtf8( "'.\nФайл не
найден или нет доступа на чтение." );
            QMessageBox::critical(                                0,
QObject::trUtf8( "Ошибка открытия файла!" ), text );
```

```

        }else{
            // Ошибки при разборе содержимого
            конфигурационного файла
            QString text = QObject::trUtf8( "Ошибка
            чтения " )
                + fname
                + QObject::trUtf8( "'" в строке " )
                + QString::number( err_pos, 10 )
                + QObject::trUtf8(
".\nЗакомментируйте/удалите/исправьте строку." );
            QMessageBox::critical(                                0,
            QObject::trUtf8( "Ошибка чтения файла!" ), text );
        };
        return 0;
    }
    int sock = socket(AF_INET, SOCK_STREAM, 0);
    if( sock < 0 ){
        return -1;
    }

    char ac[80];
    gethostname(ac, sizeof(ac));
    struct hostent *phe = gethostbyname(ac);
    struct in_addr addr_in;
    memcpy(&addr_in,                                phe->h_addr_list[0],
    sizeof(addr_in));

    struct sockaddr_in addr;
    memset(&addr, '0', sizeof( sock ) );
    addr.sin_family = AF_INET;
    // addr.sin_addr.s_addr = inet_addr("127.0.0.1");
    addr.sin_addr.s_addr                                =
    inet_addr(inet_ntoa(addr_in));
    addr.sin_port = htons(5000);
    connect(sock, (struct sockaddr *)&addr,
    sizeof(addr) );

    char operation = 0x07;
    send( sock, &operation, 1, 0);
    char * answer = new char[3];
    memset( answer, 0, 3);
    recv( sock, answer, 3, 0);
    if( strcmp(answer, "ok") != 0){
        std::cout << "Ошибка взаимодействия с
сервером" << std::endl;
        return -1;
    }
}

```

```
        CMainDlg *mdlg = new CMainDlg( sock, &dev, &conf,  
NULL );  
        mdlg->show();  
        // close( sock );  
        return app.exec();  
    };
```

ПРИЛОЖЕНИЕ Д

(обязательное)

Исходные файлы графического программного обеспечения «Настройка сети», отвечающие за работу главного окна

Текст исходного файла maindlg.h.

```
#ifndef __MAINDLG_H
#define __MAINDLG_H

#include "ui_main_netconf.h"
#include "read_config.h"
#include <QMainWindow>
#include <QListView>

//! Класс описывающий основное окно
class CMainDlg : public QMainWindow, public
Ui_mainWindow
{
    Q_OBJECT

public:
    CMainDlg( int fd, CDeviceData *_dev,
CConfigData *_conf, QWidget *_p = 0 );

private:
    QString login;
    int accessType;
    CDeviceData *dev; //!< Устройства вычитанные
из файла
    CConfigData *conf; //!< Профайлы вычитанные из
файла
    int sock;

    QAction *actChuser; //!< Действие смена
пользователя
    QAction *actChpasswd; //!< Действие смена
пароля
    QAction *actAdduser; //!< Действие
добавление пользователя
    QAction *actChmod; //!< Действие изменить права
доступа
    QAction *actCillUser; //!< Действие удаление
пользователя
```

```

        QComboBox *cmbDevices; //!< Выпадающий список
с адресами устройств
        QComboBox *cmbProfile; //!< Выпадающий список
с профайлами из конфига
        QComboBox *cmbPing;      //!< Выпадающий список
с профайлами для пинга

        QListView *listView;      //!< Указатель на
список элементов cmbPing

        QTableWidget *tblProfile; //!< Таблица со
списком записей выбранного профайла
        QTableWidget *tblIfconfig; //!< Таблица с
текущими настройками сетевых интерфейсов
        QTableWidget *tblPing;      //!< Таблица с
состоянием доступности узлов из выбранного профайла

        QPushButton *btnConnect;      //!< Кнопка
подключения к другому устройству
        QPushButton *btnSetNetConfig; //!< Кнопка
применения сетевых настроек выбранного профайла
        QPushButton *btnClearNetConfig; //!< Кнопка
сброса сетевых настроек
        QPushButton *btnPing;          //!< Кнопка
проверки доступности узлов из выбранного профайла

        int numberClean;      //!< Номер ноды для
удаления из комбобокса в пинге

    private:
        //!< Чтение конфигурации сети и заполнение
таблицы tblIfconfig
        void TableIfconfigUpdate();
        //!< Подсчет количества бит в маске
        int Msk2BitCnt( const unsigned int _s_addr );
        //!< Разбор IP/MASKA в отдельные строки
        void ParseIpMsk( const QString &_ipmsk,
QString &_ip, QString &_mask );
        //!< Сеть1 -> eth0
        void EthEn2Ru( const QString &_en, QString
&_ru );
        //!< eth0 -> Сеть1
        void EthRu2En( const QString &_ru, QString
&_en );
        void DistributionOfSecurityRoles();

    public slots:

```

```

        void ChangeUser();
        void AddUser();
        void ChangeMod();
        void ChangePassword();
        void CillUser();
        ///! Изменение таблицы с выбранным профайлом в
cmbProfile
        void TableProfileUpdate( const QString & );
        ///! Изменение таблицы с выбранным профайлом в
cmbPing
        void TablePingUpdate( const QString & );
        ///! Применение настроек из профайла
        void BtnSetNetConfig_Click();
        ///! Сброс настроек сетевых интерфейсов
        void BtnClearNetConfig_Click();
        ///! Пинг выбранных узлов
        void BtnPing_Click();
        ///! Открытие выпадающего списка для Применить
        void OpenCmbProfile();
        ///! Открытие выпадающего списка для Проверить
        void OpenCmbPing();

protected:
        void closeEvent(QCloseEvent *);
};

#endif

```

Текст исходного файла maindlg.cpp.

```

#include "maindlg.h"
#include "autordlg.h"
#include "adduserdlg.h"
#include "chmoddlg.h"
#include "chpasswdlg.h"
#include "cilluserdlg.h"
#include "ipchange.h"
#include "ping.h"

#include <QMessageBox>
#include <QShortcut>
#include <iostream>
//#include <ifconf.h>
#include <arpa/inet.h>
#include <ifaddrs.h>
#include <net/if.h>

///! Подсчет количества бит в маске
int CMainDlg::Msk2BitCnt( const unsigned int _s_addr ){

```

```

        int cnt = 0;
        for( int i=0; i<4; i++ ){
            unsigned char val = 0xFF;
            int n = 8;
            unsigned char byte = ( (_s_addr >> (8*i)) &
0xFF );

            if( byte == 0 ){
                break;
            };

            while( n && (byte != val) ){
                val <<= 1;
                n--;
            };

            cnt += n;
        };
        return cnt;
    };

    //! Разбор IP/MASKA в отдельные строки
    void CMainDlg::ParseIpMsk( const QString &_ipmsk,
    QString &_ip, QString &_mask ){
        int index = _ipmsk.indexOf('/');
        _ip = _ipmsk.left(index);
        _mask = _ipmsk.right( _ipmsk.length() - index - 1
    );
        return;
    };

    //! Сеть1 -> eth0
    void CMainDlg::EthEn2Ru( const QString &_en, QString
    &_ru ){
        if( 0 == _en.left(3).compare( "eth" ) ){
            bool ok;
            int digit = _en.mid( 3 ).toInt( &ok );
            if( true == ok ){
                _ru = QObject::trUtf8( "Сеть" );
                //_ru.append( QString::setNum( digit + 1
    ) );
                _ru.append( QString::number( digit + 1 )
    );
                return;
            };
        };
        _ru = _en;
        return;
    };

```

```

};

//! eth0 -> Сеть1
void CMainDlg::EthRu2En( const QString &_ru, QString
&_en ){
    if( 0 == _ru.left( 4 ).compare( QObject::trUtf8(
"Сеть" ) ) ){
        bool ok;
        int digit = _ru.mid( 4 ).toInt( &ok );
        if( true == ok ){
            _en = "eth";
            //_en.append( QString::setNum( digit - 1
) );
            _en.append( QString::number( digit - 1 )
);
            return;
        };
    };
    _en = _ru;
    return;
};

void CMainDlg::DistributionOfSecurityRoles() {
    btnConnect->setEnabled(true);
    cmbDevices->setEnabled(true);
    tblProfile->setEnabled(true);
    btnSetNetConfig->setEnabled(true);
    cmbProfile->setEnabled(true);
    btnClearNetConfig->setEnabled(true);
    menu_3->setEnabled(true);
    if( accessType == 0 ){
        btnConnect->setEnabled(false);
        cmbDevices->setEnabled(false);
        tblProfile->setEnabled(false);
        btnSetNetConfig->setEnabled(false);
        cmbProfile->setEnabled(false);
        btnClearNetConfig->setEnabled(false);
        menu_3->setEnabled(false);
    } else if( accessType == 1 ){
        btnConnect->setEnabled(false);
        cmbDevices->setEnabled(false);
    }
};

//! Конструктор главного окна
CMainDlg::CMainDlg( int fd, CDeviceData *_dev,
CConfigData *_conf, QWidget *_p ) : QMainWindow( _p ){

```

```

        setupUi(this);
        dev = _dev;
        conf = _conf;
        sock = fd;

        //  std::cout << __LINE__ << ": main sock: " << sock
        << std::endl;
        //  CAutorDlg *adlg = new CAutorDlg( sock, NULL );
        //  adlg->exec();
        //  std::cout << __LINE__ << ": main sock: " << sock
        << std::endl;

        // Приводим именованное из ui файла к собственному
        actChuser = action_4;
        actChpasswd = action_5;
        actAdduser = action;
        actChmod = action_2;
        actCillUser = action_3;
        cmbDevices = comboBox_3;
        cmbProfile = comboBox;
        cmbPing = comboBox_2;
        tblProfile = tableWidget_2;
        tblIfconfig = tableWidget;
        tblPing = tableWidget_3;
        btnConnect = pushButton_4;
        btnSetNetConfig = pushButton;
        btnClearNetConfig = pushButton_2;
        btnPing = pushButton_3;

        //  Говорим, что ещё не было сделано никаких
        применений настроек
        numberClean = -1;

        //  Создаём список для элементов комбобокса в ПИНГ
        listView = new QListView( cmbPing );
        cmbPing->setView( listView );

        //  Настраиваем таблицы
        tblProfile->horizontalHeader()->setResizeMode( 1,
        QHeaderView::Stretch );
        tblIfconfig->horizontalHeader()->setResizeMode( 1,
        QHeaderView::Stretch );
        tblPing->horizontalHeader()->setResizeMode( 2,
        QHeaderView::Stretch );
        tblPing->horizontalHeader()->setResizeMode( 3,
        QHeaderView::Stretch );
        tblIfconfig->setSortingEnabled( true );

```

```

tblProfile->setSortingEnabled( true );
tblPing->setSortingEnabled( true );

// Устанавливаем порядок обхода элементов по
часовой стрелке
setTabOrder( cmbDevices, btnConnect );
setTabOrder( btnConnect, cmbProfile );
setTabOrder( cmbProfile, btnSetNetConfig );
setTabOrder( btnSetNetConfig, btnClearNetConfig );
setTabOrder( btnClearNetConfig, btnPing );
setTabOrder( btnPing, tblPing );
setTabOrder( tblPing, cmbPing );
setTabOrder( cmbPing, tblIfconfig );
setTabOrder( tblIfconfig, tblProfile );
setTabOrder( tblProfile, cmbProfile );

// Заполняем выпадающие списки
QString str;
CDeviceData::iterator it_d;
CConfigData::iterator it;

cmbDevices->clear();
for( it_d = _dev->begin(); it_d != _dev->end();
it_d = _dev->upper_bound( it_d->first ) ){
    str = QObject::trUtf8( it_d->first.c_str() );
    cmbDevices->addItem( str );
};

cmbProfile->clear();
cmbPing->clear();
cmbPing->addItem( QObject::trUtf8( "Все" ) );
for( it = _conf->begin(); it != _conf->end(); it =
_conf->upper_bound( it->first ) ){
    str = QObject::trUtf8( it->first.c_str() );
    cmbProfile->addItem( str );
    cmbPing->addItem( str );
};

// Заполняем таблицы
// Профайл
it = _conf->begin();
if( it != _conf->end() ){
    TableProfileUpdate(      QObject::trUtf8(      it-
>first.c_str() ) );
};
// Сетевые настройки
TableIfconfigUpdate();

```

```

// Пинг
TablePingUpdate( QObject::trUtf8( "Все" ) );

//Создаём акселератор для кнопки Применить
QShortcut * keyAltG;
keyAltG = new QShortcut( this );
keyAltG->setKey( Qt::ALT + Qt::Key_G );

//Создаём акселератор для кнопки Сброс
QShortcut * keyAltC;
keyAltC = new QShortcut( this );
keyAltC->setKey( Qt::ALT + Qt::Key_C );

//Создаём акселератор для кнопки Пинг
QShortcut * keyAltH;
keyAltH = new QShortcut( this );
keyAltH->setKey( Qt::ALT + Qt::Key_H );

//Создаём акселератор для первого комбобокса
QShortcut * keyAltJ;
keyAltJ = new QShortcut( this );
keyAltJ->setKey( Qt::ALT + Qt::Key_J );

//Создаём акселератор для второго комбобокса
QShortcut * keyAltD;
keyAltD = new QShortcut( this );
keyAltD->setKey( Qt::ALT + Qt::Key_D );

//
// if( adlg->result == true ){
//     login = adlg->lineLogin->text();
//     accessType = adlg->accessType;
//     DistributionOfSecurityRoles();
// }
//
ChangeUser();

// Соединяем сигналы и слоты
connect( actChuser, SIGNAL( triggered()), this,
SLOT(ChangeUser()));
connect( actAdduser, SIGNAL( triggered()), this,
SLOT(AddUser()));
connect( actChmod, SIGNAL( triggered()), this,
SLOT(ChangeMod()));
connect( actChpasswd, SIGNAL( triggered()), this,
SLOT(ChangePassword()));
connect( actCillUser, SIGNAL( triggered()), this,
SLOT(CillUser()));

```

```

        connect(      cmbProfile,      SIGNAL(activated(const
QString &)),  this,  SLOT(TableProfileUpdate(const  QString
&)));
        connect( keyAltJ, SIGNAL(activated()), this, SLOT(
OpenCmbProfile()));
        connect( cmbPing, SIGNAL(activated(const  QString
&)),  this,  SLOT(TablePingUpdate( const  QString & )));
        connect( keyAltD, SIGNAL(activated()), this, SLOT(
OpenCmbPing()));
        connect( btnSetNetConfig, SIGNAL(clicked()), this,
SLOT(BtnSetNetConfig_Click()));
        connect( keyAltG, SIGNAL(activated()), this, SLOT(
BtnSetNetConfig_Click()));
        connect(      btnClearNetConfig,      SIGNAL(clicked()),
this,  SLOT(BtnClearNetConfig_Click()));
        connect(      keyAltC,      SIGNAL(activated()),      this,
SLOT(BtnClearNetConfig_Click()));
        connect(      btnPing,      SIGNAL(clicked()),      this,
SLOT(BtnPing_Click()));
        connect(      keyAltH,      SIGNAL(activated()),      this,
SLOT(BtnPing_Click()));

```

```

};

```

```

void CMainDlg::closeEvent(QCloseEvent * event){
    char a = 0x08;
    send(sock, &a, 1, 0);
};

```

```

void CMainDlg::ChangeUser(){
    CAutorDlg *adlg = new CAutorDlg( sock, NULL );
    adlg->exec();

```

```

    if( adlg->result == true ){
        login = adlg->lineLogin->text();
        accessType = adlg->accessType;
        DistributionOfSecurityRoles();
        repaint();
    }

```

```

};

```

```

void CMainDlg::AddUser(){
    CAddUserDlg *adlg = new CAddUserDlg( sock, NULL );
    adlg->show();
};

```

```

void CMainDlg::ChangeMod(){
    CChmodDlg *cmdlg = new CChmodDlg( sock, NULL );

```

```

        cmdlg->show();
    };

    void CMainDlg::ChangePassword() {
        CChPasswdDlg *cpdlg = new CChPasswdDlg(
sock, login, NULL );
        cpdlg->show();
    };

    void CMainDlg::CillUser() {
        CCillUserDlg *cudlg = new CCillUserDlg( sock,
login, NULL );
        cudlg->show();
    };

    //! Открытие выпадающего списка для Применить
    void CMainDlg::OpenCmbProfile() {
        cmbProfile->showPopup();
    };

    //! Открытие выпадающего списка для Проверить
    void CMainDlg::OpenCmbPing() {
        cmbPing->showPopup();
    };

    //! Изменение таблицы с выбранным профайлом в
    cmbProfile
    void CMainDlg::TableProfileUpdate( const QString &_text
) {
        // Очистили таблицу
        tblProfile->clearContents();
        tblProfile->setRowCount( 0 );

        // Выделили записи относящиеся к профайлу с ключем
_text
        std::pair<CConfigData::iterator, CConfigData::itera
tor> p = conf->equal_range( _text.toLocal8Bit().data() );
        CConfigData::iterator it = p.first;

        // Устанавливаем число строк в таблице
        tblProfile->setRowCount( conf->count(
_text.toLocal8Bit().data() ) );

        // Отключаем сортировку на время добавления
элементов
        tblProfile->setSortingEnabled( false );

        // Заполняем таблицу

```

```

        int i=0; // номер строки в таблице
        while( it != p.second ){
            // Преобразование eth0 -> Сеть1
            QString ru;
            QString en( it->second.ename.c_str() );
            EthEn2Ru( en, ru );
            // добавляем имя интерфейса
            QTableWidgetItem * item0 = new
QTableWidgetItem(ru);
            item0->setFlags( Qt::ItemIsSelectable ); //
столбец только для чтения
            tblProfile->setItem( i, 0, item0 );
            // добавляем IP адрес
            std::string ip_msk = it->second.ip + '/' + it-
>second.mask;
            tblProfile->setItem( i, 1, new
QTableWidgetItem( ip_msk.c_str() ) );
            // берем следующее значение
            i++;
            it++;
        };

        // Включаем сортировку
        tblProfile->setSortingEnabled( true );

        return;
    };

    //! Чтение конфигурации сети и заполнение таблицы
tblIfconfig
    void CMainDlg::TableIfconfigUpdate(){
        // Очистили таблицу
        tblIfconfig->clearContents();
        tblIfconfig->setRowCount( 0 );

        // Отключаем сортировку на время добавления
элементов
        tblIfconfig->setSortingEnabled( false );

        // Читаем конфигурацию сети
        struct ifaddrs *ifap, *ifa;
        getifaddrs( &ifap );

        // Перебираем все сетевые элементы
        int i = 0;
        for( ifa = ifap; ifa; ifa = ifa->ifa_next ){
            // Только IPv4

```

```

        if(( ifa->ifa_addr->sa_family == AF_INET
)&&( 0 != strcmp( ifa->ifa_name, "lo" ) )){
        // Преобразование eth0 -> Сеть1
        QString ru;
        QString en( ifa->ifa_name );
        EthEn2Ru( en, ru );
        // Устанавливаем число строк в таблице
        tblIfconfig->setRowCount( i + 1 );
        // добавляем имя интерфейса
        QTableWidgetItem * item0 = new
QTableWidgetItem(ru);
        item0->setFlags(Qt::ItemIsSelectable); //
столбец только для чтения
        tblIfconfig->setItem( i, 0, item0 );

        // добавляем IP адрес
        QString ip_msk;
        ip_msk = inet_ntoa( ((struct sockaddr_in
*)ifa->ifa_addr)->sin_addr );
        ip_msk += '/';
        ip_msk += QString::number( Msk2BitCnt(
((struct sockaddr_in *)ifa->ifa_netmask)->sin_addr.s_addr )
);
        QTableWidgetItem * item1 = new
QTableWidgetItem( ip_msk );
        item1->setFlags(Qt::ItemIsSelectable); //
столбец только для чтения
        tblIfconfig->setItem( i, 1, item1 );
        // берем следующее значение
        i++;
    };
};
freeifaddrs( ifap );

// Включаем сортировку
tblIfconfig->setSortingEnabled( true );

return;
};

//! Изменение таблицы с выбранным профайлом в smbPing
void CMainDlg::TablePingUpdate( const QString &_text ){
    // Очистили таблицу
    tblPing->clearContents();
    tblPing->setRowCount( 0 );

    // Выделили записи, относящиеся к профайлу с
ключем _text

```

```

std::pair<CConfigData::iterator,CConfigData::itera
tor> p;
if( _text == QObject::trUtf8( "Все" ) ){
    p = std::make_pair( conf->begin(), conf->end()
);
    // Устанавливаем число строк в таблице
    if( numberClean != -1 ){
        //Удаляем место в таблице от элементов
установленной конфигурации
        tblPing->setRowCount(      conf->size()      -
conf->count(      cmbPing->itemText(      numberClean
).toLocal8Bit().data()));
    } else {
        tblPing->setRowCount( conf->size() );
    };
} else {
    p = conf->equal_range(
_text.toLocal8Bit().data() );
    // Устанавливаем число строк в таблице
    tblPing->setRowCount(      conf->count(
_text.toLocal8Bit().data() ) );
};
CConfigData::iterator it = p.first;

// Отключаем сортировку на время добавления
элементов
tblPing->setSortingEnabled( false );

// Заполняем таблицу
int i=0; // номер строки в таблице
while( it != p.second ){
    QTableWidgetItem *qtwi;
    // Преобразование eth0 -> Сеть1
    QString ru;
    QString en( it->second.ename.c_str() );
    EthEn2Ru( en, ru );
    // Имя профайла
    if( ( numberClean != -1 ) && ( it->first ==
cmbPing->itemText( numberClean ).toLocal8Bit().data())){
        it++; // Удаляем из таблицы элементы
установленной конфигурации
    } else {
        qtwi = new QTableWidgetItem(
QObject::trUtf8( it->first.c_str() ) );
        qtwi->setFlags( Qt::ItemIsSelectable );
        // столбец только для чтения
        tblPing->setItem( i, 0, qtwi );

```

```

        // добавляем имя интерфейса
        qtwi = new QTableWidgetItem( ru );
        qtwi->setFlags( Qt::ItemIsSelectable );
// столбец только для чтения
        tblPing->setItem( i, 1, qtwi );
        // добавляем IP адрес
        std::string ip_msk = it->second.ip + '/'
+ it->second.mask;
        tblPing->setItem( i, 2, new
QTableWidgetItem( ip_msk.c_str() ) );
        // доавбляем пустой виджет состояние
        qtwi = new QTableWidgetItem();
        qtwi->setFlags( Qt::ItemIsSelectable );
// столбец только для чтения
        tblPing->setItem( i, 3, qtwi );
        // берем следующее значение
        i++;
        it++;
    }
};

// Включаем сортировку
tblPing->setSortingEnabled( true );

return;
};

//! Применение настроек из профайла
void CMainDlg::BtnSetNetConfig_Click() {
    int isError = 0; //счётчик ошибок
    for( int i = 0; i < tblProfile->rowCount(); i++ ){
        QTableWidgetItem *qtwi_ename = tblProfile-
>item( i, 0 );
        QTableWidgetItem *qtwi_ip = tblProfile->item(
i, 1 );
        if(( NULL != qtwi_ename )&&( NULL != qtwi_ip
)){
            // Разбираем строку на составляющие
            QString ename, ip, mask;
            ename = qtwi_ename->text();
            ParseIpMsk( qtwi_ip->text(), ip, mask );
            // Преобразование Сеть1 -> eth0
            QString en;
            EthRu2En( ename, en );
            if( 0 != IpChange(
en.toStdString().c_str(), ip.toStdString().c_str() ) ){
                isError++;
                QMessageBox::warning(

```

```

        this,
        QObject::trUtf8( "Ошибка
настройки сети" ),
        QObject::trUtf8( "Неудачная
смена IP адреса в " ) + ename
    );
    } else if( 0 != MaskChange(
en.toStdString().c_str(), mask.toStdString().c_str() ) ){
        isError++;
        QMessageBox::warning(
            this,
            QObject::trUtf8( "Ошибка
настройки сети" ),
            QObject::trUtf8( "Неудачная
смена маски адреса в " ) + ename
        );
    } else if( 0 != SetUpDown(
en.toStdString().c_str(), (IFF_UP | IFF_RUNNING)) ){
        isError++;
        QMessageBox::warning(
            this,
            QObject::trUtf8( "Ошибка
настройки сети" ),
            QObject::trUtf8( "Неудачное
включение интерфейса " ) + ename
        );
    };
};

};

if( isError == 0 ) {
    if( numberClean != -1 ){
        listView->setRowHidden( numberClean ,
false ); // Возвращаем в выпадающий список Пинг ранее
применённую конфигурацию
    };
    numberClean = cmbProfile->currentIndex() + 1;
    listView->setRowHidden( numberClean , true );
    // Удаляем из выпадающего списка Пинг новую применённую
конфигурацию
    TablePingUpdate( QObject::trUtf8( "Все" ) );
};

TableIfconfigUpdate();

return;
};

```

```

    //! Сброс настроек сетевых интерфейсов
void CMainDlg::BtnClearNetConfig_Click() {
    // Количество правильно сброшенных адресов
    int isGood = 0;
    for( int i = 0; i < tblIfconfig->rowCount(); i++
){
        QTableWidgetItem *qtwi = tblIfconfig->item( i,
0 );
        if( NULL != qtwi ){
            // Преобразование Сеть1 -> eth0
            QString en;
            EthRu2En( qtwi->text(), en );
            if( 0 != SetupDown(
en.toStdString().c_str(), IFF_UP )){
                QMessageBox::warning(
                    this,
                    QObject::trUtf8( "Ошибка сброса сети"
),
                    QObject::trUtf8( "Неудачное
выключение интерфейса в " ) + qtwi->text()
                );
            } else
            if( 0 != IpChange(
en.toStdString().c_str(), "0.0.0.0" ) ){
                QMessageBox::warning(
                    this,
                    QObject::trUtf8( "Ошибка сброса
сети" ),
                    QObject::trUtf8( "Неудачная
смена IP адреса в " ) + qtwi->text()
                );
            } else {
                isGood++;
            }
        };
    };
    if ( isGood == tblIfconfig->rowCount() ){
        if( numberClean != -1 ){
            listView->setRowHidden( numberClean ,
false ); // Возвращаем в выпадающий список Пинг ранее
применённую конфигурацию
        };
        numberClean = -1;
        TablePingUpdate(QObject::trUtf8( "Все" )); //
Восстанавливаем полную таблицу Пинг
    };

    TableIfconfigUpdate();
}

```

```

        return;
    };

    //! Пинг выбранных узлов
    void CMainDlg::BtnPing_Click() {
        cmbPing->setEnabled(false);
        btnPing->setEnabled(false);
        // Цвета строк
        QColor clr_check( 190, 190, 190, 200 );          //
Проверка
        QColor clr_access( 142, 251, 152, 200 ); //
Доступен
        QColor clr_notaccess( 255, 120, 120, 180 ); //
Недоступен
        QColor clr_error( 255, 0, 0 );                  // Ошибка

        // Отключаем сортировку на время изменения
элементов
        tblPing->setSortingEnabled( false );

        PingDataType host;
        // Выставляем "Проверка" в колонке "Состояние"
        for( int i = 0; i < tblPing->rowCount(); i++ ){
            for( int j=0; j<tblPing->columnCount(); j++ ){
                QTableWidgetItem *qtwi = tblPing->item(
i, j );

                if( NULL != qtwi ){
                    qtwi->setBackground( clr_check );
                    if( j == 2 )
                    {
                        host.insert(      std::make_pair(
qtwi->text().section('/',0,0).toStdString(),    PR_UNAVAILABLE
) );
                    } else if ( j == 3 ) {
                        qtwi->setText(    QObject::trUtf8(
"Проверка..." ) );
                    }
                }
            };
        };

        // Обновляем экран
        // Почему-то после первого не отрисовывает
        repaint();
        repaint();

        // Пингуем
        int ret = ping( host );
    }

```

```

        if( ret == 0 ){
            for( int i = 0; i < tblPing->rowCount(); i++
){
                QTableWidgetItem *qtwi_ipmsk = tblPing-
>item( i, 2 ); // IP Адрес/Маска
                QString state = QObject::trUtf8( "Ошибка"
);
                QColor *color = &clr_error;
                PingDataType::iterator it = host.find(
qtwi_ipmsk->text().section('/',0,0).toString());
                if( it != host.end() ){
                    // все хорошо
                    if( PR_AVALIBLE == it->second ){
                        state = QObject::trUtf8(
"Доступен" );
                        color = &clr_access;
                        // недоступен
                    }else if( PR_UNAVALIBLE == it->second
){
                        state = QObject::trUtf8(
"Недоступен" );
                        color = &clr_notaccess;
                    };
                };
                for( int j=0; j<tblPing->columnCount();
j++ ){
                    QTableWidgetItem *qtwi = tblPing->item(
i, j );
                    if( NULL != qtwi ){
                        // Устанавливаем цвет строки
                        qtwi->setBackground( *color );
                        if( j == 3 ) {
                            // Устанавливаем текст
                            qtwi->setText( state );
                        };
                    };
                };
                // Обновляем экран
                repaint();
                repaint();
            } else {
                for( int i = 0; i < tblPing->rowCount(); i++
){
                    for( int j=0; j<tblPing->columnCount();
j++ ){

```

```

        QTableWidgetItem *qtwi = tblPing-
>item( i, j );

        if( NULL != qtwi ){
            // Устанавливаем цвет строки
            qtwi->setBackground( clr_error);
            if( j == 3 ){
                // ошибка
                qtwi->setText(
QObject::trUtf8( "Ошибка" ) );
            }
        };

    };

    }
    // Обновляем экран
    repaint();
    repaint();
    QMessageBox::warning(
        this,
        QObject::trUtf8( "Ошибка выполнения
операции пинг" ),
        QObject::trUtf8( "Недостаточно прав для
выполнения запроса" )
    );
}

// Включаем сортировку
tblPing->setSortingEnabled( true );
cmbPing->setEnabled(true);
btnPing->setEnabled(true);

return;
};

```

ПРИЛОЖЕНИЕ Е

(обязательное)

Исходные файлы графического программного обеспечения «Настройка сети», отвечающие за работу диалоговых окон

Текст исходного файла autordlg.h.

```
#ifndef __AUTORDLG_H
#define __AUTORDLG_H

#include "ui_authorized.h"
#include <QDialog>
#include <QCloseEvent>

class CAutorDlg : public QDialog,
Ui_Dialog_authorized
{
    Q_OBJECT

public:
    CAutorDlg( int fd, QWidget *_p = 0 );
    QLineEdit *lineLogin;
    int accessType;
    bool result;

private:
    int sock;
    QString truePasswd;

    QLineEdit *linePasswd;
    QPushButton * btnOk;
    QPushButton * btnConcel;

private:
    bool Inicialize( const QString login );
    bool PassCompare( QString cryptoPasswd );

protected:
    void closeEvent(QCloseEvent *);

public slots:
    void PushBtnCheckAutor();
    void PushBtnConsel();

};
```

```
#endif
```

Текст исходного файла autordlg.cpp.

```
#include "autordlg.h"
#include "client.h"
#include "cripto.h"
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <QMessageBox>
#include <iostream>

CAutorDlg::CAutorDlg( int fd, QWidget *_p ) : QDialog(
_p ){
    setupUi( this );

    sock = fd;
    lineLogin = lineEdit;
    linePasswd = lineEdit_2;
    btnOk = pushButton;
    btnConcel = pushButton_2;

    linePasswd->setEchoMode( QLineEdit::Password );
    result = false;

    connect( btnOk, SIGNAL( clicked() ), this, SLOT(
PushBtnCheckAutor() ) );
    connect( btnConcel, SIGNAL( clicked() ), this,
SLOT( PushBtnConsel() ) );
};

void CAutorDlg::PushBtnCheckAutor(){
    char code = 0x05;
    if( send( sock, &code, 1, 0 ) < 0 ){
        std::cout << "Error send" << std::endl;
    }
    char * salt = new char[5];
    recv( sock, salt, 5, 0);

    s_autorHeader header;
    char * userName = (char*)lineLogin-
>text().append("\0").toUtf8().constData();
    header.size_userName = strlen(userName)+1;
    std::string passwd = PassCript(linePasswd-
>text().toStdString(), salt ).append("\0");
    for( int i = 0; i < 65; i++ ){
        header.passwd[i] = passwd.c_str()[i];
```

```

    }

    code = 0x00;
    send( sock, &code, 1, 0);
    if (send( sock, &header, sizeof(header), 0) < 0){
        std::cout << "Error send" << std::endl;
    }
    send( sock, userName, header.size_userName, 0);
    char res_code = 0xFF;
    recv( sock, &res_code, 1, 0);

    if( res_code == 0x00){
        result = true;
        int accessType_c;
        recv( sock, &accessType_c, 4, 0);
        accessType = accessType_c;
        this->hide();
    } else if( res_code == 0x02){
        QMessageBox::warning(
            this,
            QObject::trUtf8( "Ошибка авторизации" ),
            QObject::trUtf8( "Неверный логин или пароль" )
        );
        lineLogin->clear();
        linePasswd->clear();
    } else if( res_code == 0x01 ){
        QMessageBox::warning(
            this,
            QObject::trUtf8( "Ошибка авторизации" ),
            QObject::trUtf8( "Ошибка работы сервера" )
        );
        lineLogin->clear();
        linePasswd->clear();
    } else if( res_code == 0x03){
        QMessageBox::warning(
            this,
            QObject::trUtf8( "Ошибка авторизации" ),
            QObject::trUtf8( "Для входа в систему  
воспользуйтесь вашим персональным компьютером" )
        );
        lineLogin->clear();
        linePasswd->clear();
    }
};

void CAutorDlg::PushBtnConsel() {
    exit(0);
};

```

```
void CAutorDlg::closeEvent(QCloseEvent * event) {
    exit(0);
}
```

Текст исходного файла chpasswdDlg.h.

```
#ifndef __CHPASSWDDL_H
#define __CHPASSWDDL_H

#include "ui_chpasswd.h"
#include <QDialog>

class CChPasswdDlg : public QDialog, Ui_Dialog_chpasswd
{
    Q_OBJECT

public:
    CChPasswdDlg( int fd, QString log, QWidget *_p
= 0 );

private:
    int sock;
    QString login;

    QLabel * labelUser;
    QLineEdit *lePasswdOld;
    QLineEdit *lePasswdNew;
    QLineEdit *lePasswdRepid;

public slots:
    void ClickedOkBtn();

};

#endif
```

Текст исходного файла chpasswdDlg.cpp.

```
#include "chpasswdDlg.h"
#include "client.h"
#include "cripto.h"
#include <QMessageBox>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
```

```

#include <sys/socket.h>
#include <iostream>

CChPasswdDlg::CChPasswdDlg( int fd, QString log,
QWidget *_p ) : QDialog(_p){
    setupUi(this);

    sock = fd;
    login = log;

    labelUser = label_5;
    labelUser->setText(login);

    lePasswdOld = lineEdit;
    lePasswdNew = lineEdit_2;
    lePasswdRepid = lineEdit_3;

    connect(buttonBox, SIGNAL(accepted()), this,
SLOT(ClickedOkBtn()) );
};

void CChPasswdDlg::ClickedOkBtn(){
    if( QString::compare( lePasswdNew->text(),
lePasswdRepid->text() ) != 0 ){
        QMessageBox::warning(
            this,
            QObject::trUtf8( "Ошибка смены пароля" ),
            QObject::trUtf8( "Новые пароли не совпадают" )
        );
        lePasswdOld->clear();
        lePasswdNew->clear();
        lePasswdRepid->clear();
    } else {
        char code = 0x05;
        send( sock, &code, 1, 0);
        char * salt = new char[5];
        recv( sock, salt, 5, 0);

        char * userName =
(char*)login.append("\0").toUtf8().constData();
        s_chPasswdHeader header;
        header.size_userName = strlen(userName)+1;
        std::string passwd_old =
PassCript(lePasswdOld->text().toStdString(), salt
).append("\0");
        std::string passwd_new =
PassCript(lePasswdNew->text().toStdString(), salt
).append("\0");
    }
}

```

```

        for( int i = 0; i < 65; i++ ){
            header.passwd_old[i]
passwd_old.c_str()[i];
        }
        for( int i = 0; i < 65; i++ ){
            header.passwd_new[i]
passwd_new.c_str()[i];
        }

        code = 0x01;
        send( sock, &code, 1, 0);
        send( sock, &header, sizeof(header), 0);
        send( sock, userName, header.size_userName,
0);

        char res_code;
        recv( sock, &res_code, 1, 0);

        if( res_code == 0x02 ){
            QMessageBox::warning(
                this,
                QObject::trUtf8( "Ошибка смены пароля" ),
                QObject::trUtf8( "Неверный старый пароль"
)
            );
            lePasswdOld->clear();
            lePasswdNew->clear();
            lePasswdRepid-> clear();
        } else if ( res_code == 0x01 ){
            QMessageBox::warning(
                this,
                QObject::trUtf8( "Ошибка смены пароля" ),
                QObject::trUtf8( "Ошибка работы сервера"
)
            );
            lePasswdOld->clear();
            lePasswdNew->clear();
            lePasswdRepid-> clear();
        } else if ( res_code == 0x00 ){
            this->hide();
        }
    }

};

```

Текст исходного файла adduserdlg.h.

```

#ifndef __ADDUSERDLG_H
#define __ADDUSERDLG_H

```

```

#include "ui_adduser.h"
#include <QDialog>

class CAddUserDlg : public QDialog, Ui_Dialog
{
    Q_OBJECT

public:
    CAddUserDlg( int fd, QWidget *_p = 0 );

private:
    int sock;
    QString userName;
    QString passwdHash;

    QLineEdit *leUserName;
    QLineEdit *lePasswd;
    QLineEdit *lePasswdRepid;
    QComboBox *cmbAccessType;
    QLineEdit *leIP;

    bool CheckName( QString userName );

public slots:
    void PushBtnOk();

};

#endif

```

Текст исходного файла adduserdlg.cpp.

```

#include "adduserdlg.h"
#include "client.h"
#include "cripto.h"
#include <QMessageBox>
#include "autordlg.h"
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <fstream>
#include <iostream>

CAddUserDlg::CAddUserDlg( int fd, QWidget *_p ):
QDialog( _p ){
    setupUi( this );

    sock = fd;

```

```

        leUserName = lineEdit;
        lePasswd = lineEdit_2;
        lePasswdRepid = lineEdit_3;
        cmbAccessType = comboBox;
        leIP = lineEdit_5;

        connect( buttonBox, SIGNAL( accepted() ), this,
SLOT( PushBtnOk() ) );
    };

void CAddUserDlg::PushBtnOk() {
    int err = 0;
    if( leUserName->text().length() == 0 ){
        err = 1;
    }
    if( lePasswd->text().length() == 0 ){
        err = 1;
    }
    if( lePasswdRepid->text().length() == 0 ){
        err = 1;
    }
    if(      QString::compare(      lePasswd->text(),
lePasswdRepid->text() ) != 0 ){
        err = 2;
    }

    char accessType;
    if( QString::compare(cmbAccessType->currentText(),
QString::fromUtf8("Пользователь") ) == 0 ){
        accessType = 0;
    } else if(      QString::compare(cmbAccessType-
>currentText(),      QString::fromUtf8("Локальный
администратор") ) == 0 ){
        accessType = 1;
    } else if(      QString::compare(cmbAccessType-
>currentText(),      QString::fromUtf8("Системный
администратор") ) == 0 ){
        accessType = 2;
    }

    if( err == 1 ){
        QMessageBox::warning(
            this,
            QObject::trUtf8(      "Ошибка      добавления
пользователя" ),
            QObject::trUtf8( "Заполнены не все поля" )
        );
        leUserName->clear();
    }
}

```

```

        lePasswd->clear();
        lePasswdRepid->clear();
        cmbAccessType->setCurrentIndex(0);
    } else if ( err == 2 ){
        QMessageBox::warning(
            this,
            QObject::trUtf8( "Ошибка добавления
пользователя" ),
            QObject::trUtf8( "Пароли не совпадают" )
        );
        leUserName->clear();
        lePasswd->clear();
        lePasswdRepid->clear();
        cmbAccessType->setCurrentIndex(0);
    } else {
        char code = 0x05;
        send( sock, &code, 1, 0);
        char * salt = new char[5];
        recv( sock, salt, 5, 0);

        s_addUserHeader header;
        char * username = (char*)leUserName-
>text().append("\0").toUtf8().constData();
        header.size_username = strlen(username)+1;
        std::string passwd = PassCrypt(lePasswd-
>text().toStdString(), salt ).append("\0");
        for( int i = 0; i < 65; i++ ){
            header.passwd[i] = passwd.c_str()[i];
        }
        header.accessType = accessType;
        const char * adress = leIP-
>text().append("\0").toUtf8().constData();
        int slen = strlen(adress);
        for( int i = 0; i < (16 - slen - 1); i++){
            header.ip[i] = '0';
        }
        for( int i = (16 - slen); i < 16; i++){
            header.ip[i] = adress[i];
        }

        code = 0x02;
        send( sock, &code, 1, 0);
        send( sock, &header, sizeof(header), 0);
        send( sock, username, header.size_username,
0);

        char res_code;
        recv( sock, &res_code, 1, 0);

```

```

        if( res_code == 0x02 ){
            QMessageBox::warning(
                this,
                QObject::trUtf8(      "Ошибка      добавления
пользователя" ),
                QObject::trUtf8(      "Пользователь с таким
именем уже существует" )
            );
            leUserName->clear();
            lePasswd->clear();
            lePasswdRepid->clear();
            cmbAccessType->setCurrentIndex(0);
        }else if( res_code == 0x01){
            QMessageBox::warning(
                this,
                QObject::trUtf8(      "Ошибка      добавления
пользователя" ),
                QObject::trUtf8( "Ошибка сервера" )
            );
            leUserName->clear();
            lePasswd->clear();
            lePasswdRepid->clear();
            cmbAccessType->setCurrentIndex(0);
        }else{
            QMessageBox::information(
                this,
                QObject::trUtf8( "Информация" ),
                QObject::trUtf8(      "Успешное      добавление
пользователя" )
            );
            leUserName->clear();
            lePasswd->clear();
            lePasswdRepid->clear();
            cmbAccessType->setCurrentIndex(0);
        }
    }
}

```

Текст исходного файла chmoddlg.h.

```

#ifndef __CHMODDLG_H
#define __CHMODDLG_H

#include "ui_chmod.h"
#include <QDialog>

class CChmodDlg : public QDialog, Ui_Dialog_chmod
{
    Q_OBJECT

```

```

public:
    CChmodDlg( int fd, QWidget *_p = 0 );

private:
    int sock;
    QComboBox *cmbUserName;
    QComboBox *cmbAccessType;

    bool FillCmbUserName();
    bool CorrectBDFFile(QString userName, int
accessType);

    public slots:
        void PushBtnOk();
};

#endif

```

Текст исходного файла chmoddlg.cpp.

```

#include "chmoddlg.h"
#include "client.h"
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <QMessageBox>
#include <iostream>

bool CChmodDlg::FillCmbUserName(){
    char code = 0x06;
    send( sock, &code, 1, 0);

    int cnt;
    recv( sock, &cnt, 4, 0);
    for( int i = 0; i < cnt; i++ ){
        int sUserName;
        recv( sock, &sUserName, 4, 0);
        char * userName = new char[sUserName];
        recv( sock, userName, sUserName, 0);
        cmbUserName->addItem(
QString::fromUtf8(userName) );
    }

    return true;
};

CChmodDlg::CChmodDlg( int fd, QWidget *_p ):
QDialog(_p){

```

```

        setupUi( this );

        sock = fd;
        cmbUserName = comboBox;
        cmbAccessType = comboBox_2;

        FillCmbUserName();

        connect( buttonBox, SIGNAL( accepted() ), this,
        SLOT( PushBtnOk() ) );
    };

    void CChmodDlg::PushBtnOk() {
        char accessType;
        if( QString::compare(cmbAccessType->currentText(),
        QString::fromUtf8("Пользователь") ) == 0 ){
            accessType = 0;
        } else if( QString::compare(cmbAccessType->currentText(),
        QString::fromUtf8("Локальный администратор") ) == 0 ){
            accessType = 1;
        } else if( QString::compare(cmbAccessType->currentText(),
        QString::fromUtf8("Системный администратор") ) == 0 ){
            accessType = 2;
        }

        s_chModHeader header;
        char * userName = (char*)cmbUserName->currentText().append("\0").toUtf8().constData();
        header.size_userName = strlen(userName)+1;
        header.accessType = accessType;

        char code = 0x03;
        send( sock, &code, 1, 0);
        send( sock, &header, sizeof(header), 0);
        send( sock, userName, header.size_userName, 0);
        char res_code;
        recv( sock, &res_code, 1, 0);

        if( res_code == 0x01 ){
            QMessageBox::warning(
            this,
            QObject::trUtf8( "Ошибка изменения прав пользователя" ),
            QObject::trUtf8( "Ошибка обработки данных" )
            );
        } else if( res_code == 0x00 ) {

```

```

        QMessageBox::information(
            this,
            QObject::trUtf8( "Информация" ),
            QObject::trUtf8( "Успешная смена прав
пользователя" )
        );
    }

};

```

Текст исходного файла cilluserdlg.h.

```

#ifndef __CILLUSERDLG_H
#define __CILLUSERDLG_H

#include "ui_cilluser.h"
#include <QDialog>
#include <QCloseEvent>

class CCillUserDlg : public QDialog, Ui_Dialog_cilluser
{
    Q_OBJECT

public:
    CCillUserDlg( int fd, QString _login, QWidget
*_p = 0 );

private:
    int sock;
    QString login;
    QString truePasswd;
    QComboBox * cmbUserName;
    QLineEdit * lePasswdAdmin;

    bool FillCmbUserName();
    bool CillUser();

private slots:
    void PushBtnOk();
};

#endif

```

Текст исходного файла cilluserdlg.cpp.

```

#include "cilluserdlg.h"
#include "client.h"
#include "cripto.h"
#include <sys/stat.h>

```

```

#include <sys/types.h>
#include <sys/socket.h>
#include <QMessageBox>
#include <iostream>

bool CCillUserDlg::FillCmbUserName(){
    cmbUserName->clear();
    char code = 0x06;
    send( sock, &code, 1, 0);

    int cnt = 0;
    int r_size = recv( sock, &cnt, 4, 0);
    std::cout << "r_size: " << r_size << std::endl;
    std::cout << "cnt: " << cnt << std::endl;
    for( int i = 0; i < cnt; i++ ){
        int sUserName;
        recv( sock, &sUserName, 4, 0);
        char * userName = new char[sUserName];
        recv( sock, userName, sUserName, 0);
        cmbUserName->addItem(
QString::fromUtf8(userName) );
    }

    return true;
};

CCillUserDlg::CCillUserDlg( int fd, QString _login,
QWidget *_p) :QDialog( _p ){
    setupUi(this);
    sock = fd;
    login = _login;

    cmbUserName = comboBox_2;
    lePasswdAdmin = lineEdit;

    FillCmbUserName();

    connect( buttonBox, SIGNAL( accepted() ), this,
SLOT( PushBtnOk() ) );
};

void CCillUserDlg::PushBtnOk(){

    char code = 0x05;
    send( sock, &code, 1, 0);
    char * salt = new char[5];
    int r_size = recv( sock, salt, 5, 0);
    std::cout << "r_size: " << r_size << std::endl;

```

```

std::cout << "salt: " << salt << std::endl;

s_authorizedHeader header;
char * userName =
(char*)login.append("\0").toUtf8().constData();
header.size_userName = strlen(userName)+1;
std::string passwd = PassCrypt(lePasswdAdmin-
>text().toStdString(), salt ).append("\0");
std::cout << "passwd: " << lePasswdAdmin-
>text().toStdString() << std::endl;
for( int i = 0; i < 65; i++ ){
    header.passwd[i] = passwd.c_str()[i];
}

std::cout << "authorize passwd: " << header.passwd
<< std::endl;
std::cout << "socket: " << sock << std::endl;
code = 0x00;
send( sock, &code, 1, 0);
send( sock, &header, sizeof(header), 0);
send( sock, userName, header.size_userName, 0);
char res_code;
recv( sock, &res_code, 1, 0);
int mod;
recv( sock, &mod, 1, 0);

if( res_code != 0x00 ){
    QMessageBox::warning(
        this,
        QObject::trUtf8( "Ошибка удаления
пользователя" ),
        QObject::trUtf8( "Неверный пароль
администратора" )
    );
    cmbUserName->setCurrentIndex(0);
    lePasswdAdmin->clear();
} else {
    s_cillUserHeader header_cill;
    char * userName = (char*)cmbUserName-
>currentText().append("\0").toUtf8().constData();
    header_cill.size_userName =
strlen(userName)+1;

    code = 0x04;
    send( sock, &code, 1, 0);
    send( sock, &header_cill, sizeof(header_cill),
0);

```

```

        send( sock, userName,
header_cill.size_userName, 0);
        char r_code = 0;
        recv( sock, &r_code, 1, 0);
        printf( "r_code: %d\n", r_code);

        if( r_code == 0x00 ){
            this->repaint();
            QMessageBox::information(
                this,
                QObject::trUtf8( "Информация" ),
                QObject::trUtf8( "Успешное удаление
пользователя" )
            );
            // FillCmbUserName();
            cmbUserName->removeItem(cmbUserName->
>currentIndex());
            cmbUserName->setCurrentIndex(0);
            lePasswdAdmin->clear();
        } else {
            QMessageBox::warning(
                this,
                QObject::trUtf8( "Ошибка" ),
                QObject::trUtf8( "Ошибка удаления
пользователя" )
            );
            cmbUserName->setCurrentIndex(0);
            lePasswdAdmin->clear();
        }
    }
};

```

ПРИЛОЖЕНИЕ Ж

(обязательное)

Исходные файлы графического программного обеспечения «Настройка сети», отвечающие за криптографические преобразования.

Текст исходного файла `cripto.h`.

```
#ifndef __CRIPTO_H
#define __CRIPTO_H

#include <string>

std::string PassCrypt( std::string passwd, const char *
salt );

#endif
```

Текст исходного файла `cripto.cpp`.

```
#include "cripto.h"
#include "openssl/sha.h"
#include "string.h"

std::string PassCrypt( std::string passwd, const char *
salt ){
    passwd+=salt;
    for( int i = 0; i <= 1; i++ ){
        unsigned char
sha256digest[SHA256_DIGEST_LENGTH];
        SHA256_CTX sha256handler;
        SHA256_Init(&sha256handler);
        SHA256_Update( &sha256handler, passwd.c_str(),
strlen(passwd.c_str()) );
        SHA256_Final(sha256digest, &sha256handler);
        char result[SHA256_DIGEST_LENGTH * 2] = "";
        for( int j = 0; j < SHA256_DIGEST_LENGTH; j++
){
            sprintf( result, "%s%02x", result,
sha256digest[j] );
        }
        passwd = result;
    }
    return passwd;
};
```

ПРИЛОЖЕНИЕ И

(обязательное)

Исходные файлы графического программного обеспечения «Настройка сети», отвечающие за работу функций настройки сети.

Текст исходного файла configdata.h.

```
#ifndef __CONFIGDATA_H
#define __CONFIGDATA_H

#include <map>
#include <string>

//! Тип данных для одной записи раздела
class CRec{
public:
    //! Имя сетевого интерфейса
    std::string ename;
    //! IP
    std::string ip;
    //! Маска
    std::string mask;
};

//! Тип данных для конфигурации
typedef std::multimap< std::string, class CRec >
CConfigData;

//! Тип данных для устройств
typedef std::map< std::string, std::string >
CDeviceData;
#endif
```

Текст исходного файла ipchange.h.

```
#ifndef __IPCHANGE_H
#define __IPCHANGE_H

//! Смена IP адреса интерфейса
/*!
    Запрос ядра на смену IP адреса интерфейса
    IpChange( "eth0", "192.168.1.1" );

    @param _ethname [i] - Имя сетевого интерфейса
    @param _ip [i] - IP адрес (формат:
"0.0.0.0" .. "255.255.255.255")
    @return 0 - все хорошо
```

```

        1 - Ошибки в ходе работы
*/
int IpChange( const char *_ethname, const char *_ip );

//! Смена маски интерфейса
/*!
    Запрос ядра на смену маски интерфейса
    MaskChange( "eth0", "8" );

    @param _ethname [i] - Имя сетевого интерфейса
    @param _mask [i] - Маска, количество бит (формат:
"0" .. "32")
    @return 0 - все хорошо
        1 - Ошибки в ходе работы
*/
int MaskChange( const char *_ethname, const char *_mask
);

//! Включение/выключение интерфейса
/*!
    @param ifname - имя интерфейса
    @param flag - флаг вкл/выкл
    @return 0 - всё хорошо
        1 - ошибки при установке флага
*/
int SetUpDown(const char *ifname, short flag);

#endif

```

Текст исходного файла ipchange.cpp.

```

#include "ipchange.h"
#include <unistd.h>
#include <sys/socket.h>
#include <sys/ioctl.h>
#include <netinet/in.h>
#include <net/if.h>
#include <arpa/inet.h>
#include <string.h>

#include <stdio.h>
#include <stdlib.h>

#define MIN( _a, _b ) ( ( (_a) < (_b) ) ? (_a) : (_b) )

//! Подготовка переменной со структурой struct ifreq
#define __IFREQ_INIT( _ifr, _af, _ethname ) ({
\

```

```

        /* Очищаем структуру */
        memset( (_ifr), 0, sizeof( struct ifreq ) ); \
        /* AF_INET - IPv4 */
        (_ifr)->ifr_addr.sa_family = (_af); \
        /* Имя интерфейса */
        memcpy( (_ifr)->ifr_name, (_ethname), MIN(
strlen(_ethname), IFNAMSIZ-1 ) ); \
    })

    /* Преобразую строку с количеством бит в маске в U32
    /*!
    @param _mskBitCnt - количество бит в маске
    @param _s_addr - 32х битное представление
маски
    @return 0 - все хорошо
            -1 - ошибки при конвертации
    */
    static int __bitCnt2s_addr( const char *_mskBitCnt,
unsigned int *_s_addr ){
        // проверка параметров
        if(( NULL == _mskBitCnt )||( NULL == _s_addr )){
            // Плохие параметры
            return -1;
        };
        int mskBitCntLen = strlen( _mskBitCnt );
        if(( 1 > mskBitCntLen )||( 2 < mskBitCntLen )){
            // Неправильный формат _mskBitCnt
            // длина строки = 1 или 2
            return -1;
        };

        // Преобразуем строчку с количеством бит в число
        char *end;
        int bitCnt = strtoul( _mskBitCnt, &end, 10 );
        if( mskBitCntLen != ( end - _mskBitCnt ) ){
            // Неправильный формат _mskBitCnt
            // Число преобразованных символов не совпадает
с длиной строки
            return -1;
        };
        if( 32 < bitCnt ){
            // чило бит: [0..32]
            return -1;
        };

        // Преобразуем число бит в s_addr
        *_s_addr = 0;
        unsigned char* ch = (unsigned char*)_s_addr;

```

```

        for( int i=0; i<4; i++ ){
            if( 8 >= bitCnt ){
                *ch = 0;
                for( int i=0; i<bitCnt; i++ ){
                    *ch = ((*ch) >> 1) | 0x80;
                };
                break;
            };
            *ch = 0xFF;
            ch++;
            bitCnt -= 8;
        };

        return 0;
    };

    //! Запрос к ядру
    /*!
        Перечень доступных ioctl вызовов тут
        /usr/include/x86_64-linux-gnu/bits/ioctls.h

        @param _request [i]      - Номер запроса
        @param _data [o]   - Данные запроса
        @return 0 - все хорошо
                1 - Ошибки в ходе работы
    */
    static int __SendIOCtl( unsigned long _request, struct
ifreq *_data ){
        // Открываем сокет
        int fd = socket( AF_INET, SOCK_DGRAM, 0 );

        // Проверка
        if( 0 > fd ){
            return 1;
        };

        // Запрос к ядру о смене маски интерфейса
        int ret = ioctl( fd, _request, _data );

        // Зарываем сокет
        close( fd );

        // Проверка результата запроса
        if( 0 != ret ){
            // ошибка
            return 1;
        };
    };

```

```

        // все ок
        return 0;
    };

    //! Смена IP адреса интерфейса
    /*!
        Запрос ядра на смену IP адреса интерфейса
        IpChange( "eth0", "192.168.1.1" );

        @param _ethname [i]      - Имя сетевого интерфейса
        @param _ip [i]           - IP адрес (формат:
"0.0.0.0" .. "255.255.255.255")
        @return 0 - все хорошо
                1 - Ошибки в ходе работы
    */
    int IpChange( const char *_ethname, const char *_ip ){
        // Подготовка переменной ip
        struct ifreq ip;
        __IFREQ_INIT( &ip, AF_INET, _ethname );

        // Преобразуем строку с IP адресом в корректный
формат
        struct sockaddr_in *addr;
        addr = (struct sockaddr_in *)&ip.ifr_addr;
        inet_pton( AF_INET, _ip, &addr->sin_addr );

        // Запрос на смену IP адреса
        //set_flag(_ethname, (IFF_UP | IFF_RUNNING));
        return __SendIOCtl( SIOCSIFADDR, &ip );
    }

    //! Смена маски интерфейса
    /*!
        Запрос ядра на смену маски интерфейса
        MaskChange( "eth0", "8" );

        @param _ethname [i]      - Имя сетевого интерфейса
        @param _mask [i]         - Маска, количество бит (формат:
"0" .. "32")
        @return 0 - все хорошо
                1 - Ошибки в ходе работы
    */
    int MaskChange( const char *_ethname, const char *_mask
){
        // Подготовка переменной mask
        struct ifreq mask;
        __IFREQ_INIT( &mask, AF_INET, _ethname );

```

```

        // Преобразуем строку с Маской в корректный формат
        struct sockaddr_in *addr;
        addr = (struct sockaddr_in *)&mask.ifr_netmask;
        __bitCnt2s_addr( _mask, &addr->sin_addr.s_addr );

        // Запрос на смену маски
        return __SendIOCtl( SIOCSIFNETMASK, &mask );
    }

    //! Включение/выключение интерфейса
    /*!
        @param ifname - имя интерфейса
        @param flag    - флаг IFF_UP (вкл/выкл)
        @return 0- всё хорошо
                1    - ошибки при установке флага
    */
    int SetUpDown(const char *ifname, short flag)
    {
        struct ifreq ifr;
        // Подготавливаем переменную ifr
        __IFREQ_INIT( &ifr, AF_INET, ifname );
        if( __SendIOCtl( SIOCGIFFLAGS, &ifr ) ){
            return 1;
        };

        ifr.ifr_flags |= flag;

        return __SendIOCtl( SIOCSIFFLAGS, &ifr );
    }

```

Текст исходного файла ping.h.

```

#ifndef __PING_H
#define __PING_H

#include <map>
#include <string>

//результат пинга
enum    e_PingResult    {    PR_AVALIBLE,    PR_UNAVALIBLE,
PR_ERROR };

//тип данных для функции пинг
typedef    std::map<std::string,enum    e_PingResult>
PingDataType;

//! Опрос доступности указанных ip адресов

```

```

    /*! @param host[i][o] - список хостов для проверки
        @return ret      0 - успех
                        1 - ошибка доступа
    */
    int ping( PingDataType &_host );

#endif

```

Текст исходного файла ping.cpp.

```

#include "ping.h"
#include <netinet/in.h>
#include <netinet/ip_icmp.h>
#include <netdb.h>
#include <stdlib.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <iostream>
#include <sys/select.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <map>

#define DEFDATALEN      56
#define MAXIPLLEN      60
#define MAXICMPLEN     76

//! Вычисление контрольной суммы пакета
static int in_cksum( unsigned short *buf, int sz ){
    int nleft = sz;
    int sum = 0;
    unsigned short *w = buf;
    unsigned short ans = 0;

    while (nleft > 1) {
        sum += *w++;
        nleft -= 2;
    }

    if (nleft == 1) {
        *(unsigned char *) (&ans) = *(unsigned char *)
w;

        sum += ans;
    }
}

```

```

        sum = (sum >> 16) + (sum & 0xFFFF);
        sum += (sum >> 16);
        ans = ~sum;

        return (ans);
};

//! Выполнение операции пинг
/*! @param host[i][0] - список хостов для проверки
    @param time[i]      - ограничение по времени для
select
    @return ret          0 - успех
                        1 - ошибка прав доступа
*/
static int sub_ping( PingDataType &_host, int time )
{
    struct hostent *h;

    int pingsock;

    struct icmp *pkt;
    char packet[DEFDATALEN + MAXIPLen + MAXICMPLEN];
    pkt = (struct icmp *) packet;
    memset( pkt, 0, sizeof(packet) );
    pkt->icmp_type = ICMP_ECHO;
    pkt->icmp_cksum = in_cksum((unsigned short *) pkt,
sizeof(packet));
    pingsock = socket( AF_INET, SOCK_RAW, IPPROTO_ICMP
);
    // Обычно если недостаточно прав (нужен рут)
    if( pingsock < 0 ){
        return 1;
    }

    struct sockaddr_in pingaddr;
    memset( &pingaddr, 0, sizeof(struct sockaddr_in)
);
    pingaddr.sin_family = AF_INET;

    int count = 0;
    for( PingDataType::iterator it = _host.begin(); it
!= _host.end(); it++)
    {
        if( it->second != PR_AVALIBLE ){
            // Если не получается получить IP адрес
из имени
            h = gethostbyname( (it->first).c_str() );
            if( NULL == h ){

```

```

        it->second = PR_ERROR;
        continue;
    };

    // Формируем запрос
    memcpy(&pingaddr.sin_addr,      h->h_addr,
sizeof(pingaddr.sin_addr));

    // Отправляем запрос
    sendto( pingsock, packet, sizeof(packet),
0,      (struct sockaddr *) &pingaddr,      sizeof(struct
sockaddr_in) );
        count++;
    }
}

// Обрабатываем ответы
timeval timeout;
timeout.tv_sec = time;
timeout.tv_usec = 0;

fd_set readset;
int checked = 0;
while(1){
    FD_ZERO(&readset);
    FD_SET(pingsock, &readset);

    int res = select(pingsock + 1, &readset, NULL,
NULL, &timeout);

    if ( res == -1 )
    {
        std::cout << "error select" << std::endl;
        break;
    } else if ( res == 0 ){
        break;
    } else {

        if( FD_ISSET(pingsock, &readset) )
        {
            int c;
            struct sockaddr_in from;
            socklen_t fromlen = sizeof(from);

            memset( packet, 0, sizeof(packet) );
            c = recvfrom( pingsock, packet,
sizeof(packet), 0, (struct sockaddr *) &from, &fromlen);

```

```

/* ip + icmp */
if( c >= 76 ){
    struct iphdr *iphdr = (struct
iphdr *) packet;

    pkt = (struct icmp *) (packet +
(iphdr->ihl << 2)); /* skip ip hdr */
    // Если хост доступен
    if( pkt->icmp_type ==
ICMP_ECHOREPLY ){
        PingDataType::iterator it;
        if( (it =
_host.find(inet_ntoa(from.sin_addr)) ) != _host.end() )
        {
            it->second =
PR_AVAILABLE;

            checked++;
        }
        if( checked == count )
        {
            break;
        }
    }
};
}

// Закрываем сокет
close( pingsock );
return 0;
}

//! Опрос доступности указанных ip адресов
/*! @param host[i][o] - список хостов для проверки
    @return ret      0 - успех
                    1 - ошибка доступа
*/
int ping( PingDataType &_host ){
    int ret;
    // В случае одиночной отправки пакета,
    // есть вероятность, что пакет потеряется,
    // поэтому отправляем пакет несколько раз
    // и обновляем список _host
    ret = sub_ping( _host, 2 );
    if( ret == 0 ){
        sub_ping( _host, 2 );
        sub_ping( _host, 1 );
    }
}

```

```
    }  
    return ret;  
}
```